Journal of Computer Science 10 (9): 1743-1751, 2014
ISSN: 1549-3636

doi:10.3844/jcssp.2014.1743.1751 Published Online 10 (9) 2014 (http://www.thescipub.com/jcs.toc)

IMPLEMENTATION OF CENTRAL QUEUE BASED REALTIME SCHEDULER FOR MULTIPLE SOURCE DATA STREAMING

[1]Kaviha, V., [2]V. Kannan and [3]S. Ravi

[1,3]Department of ECE, Dr.M.G.R Educational and Research Institute University, Chennai, India
[2]Jeppiaar Institute of Technology, Chennai, India

Received 2014-02-11; Revised 2014-02-15; Accepted 2014-04-22

ABSTRACT

Real-time data packet sources are required to remain robust against different security threats. This study proposes a real-time secure scheduling strategy for data transmission to enhance the communication throughput and reduce the overheads. The proposed system combines real-time scheduling with security service enhancement, error detection and realtime scheduler based on EDF algorithm using uc/os-II real time operating system, implemented on cortex M3 processor. The scheduling unit uses central queue management model and the security enhancement scheme adopts a blowfish encryption mechanism.

Keywords: Blowfish, Central Queue, EDF, Priority Scheduling, Micro/Os-II

1. INTRODUCTION

Realtime operating systems perform scheduling of tasks using “priority-based preemptive scheduling.” Each task in a software application is assigned a priority, with higher priority values representing the need for quicker responsiveness. “Preemptive” means that the scheduler is allowed to stop any task at any point in its execution, if it determines that another task needs to run immediately. In modern RTOS’s, multitasking is a technique used for enabling multiple tasks to share a single processor. It is simply the ability to run two or more independent tasks on one CPU what appears to be at the same time and not actually running concurrently. A realtime kernel like uc/os-II supports multitasking. It is a priority-based pre-emptive real-time multitasking operating system kernel for processors, written mainly in the C programming language (Abt and Thomas, 2013). The adoption of uc/os-II allows to quickly create a system that can do many things at the same time. It has the provision to automatically adjust the priority of a task during its runtime for inter task communication using kernel provided calls and creating true realtime responsive system.This is desirable feature to have realtime for avoiding riority inversion. To overcome priority inversion, uc/os-II supports priority ceiling and semaphore protocol mechanisms. It also provides priority based scheduler to improve throughput, enhancing speed and a queue based scheduler as a compile time option. When a task is considered, the key parameters include deadline, memory space required, waiting time, process time, turn-around time (Keerthika and Kasthuri, 2012).

In realtime applications for data communication, Priority-based task scheduling strategy which is designed to avoid important task to be lost in system, divides tasks into three types: Sending data packet, transmitting data packet and sensing local data according to the functions of different tasks in network. Therefore, it guarantees the more important task to be run in a priority way. Thus, throughput of the system is improved. The other important point to mention is that applying appropriate method of scheduling causes significant enhancement of fairness in task scheduling (Nojabaei *et al*., 2012).

Preemptive EDF strategy widely used in real-time system that is most optimal and dynamic scheduling for single processor. Undesirable deadline interchanges may occur with EDF scheduling. When a shared resource is accessed by tasks using critical sections (to prevent it

Corresponding Author: Kaviha, V., Department of ECE, Dr.M.G.R Educational and Research Institute University, Chennai, India

from being pre-empted by another task with an earlier deadline waiting for access to the same shared resource), it becomes important for the scheduler to temporarily assign the earliest deadline from amongst the other tasks waiting for the resource, to the task while it is within its critical section to prevent the task with earlier deadlines miss their respective deadline, especially if the task within its critical section has a much longer time to complete and its exit from its critical section and subsequent release of the shared resource may be delayed.

For avoiding this situation, priority ceiling protocol is implemented in which a task owning the resource lock running at a higher priority than any other task that may acquire the resource. Each shared resource is initialized to a priority ceiling and whenever a task locks the resource, the priority of the task is raised to the priority ceiling. It works as long as the priority ceiling is greater than the priorities of any another tasks that may lock the resource. These resources are implemented using semaphores. Semaphores are added into the resource structure along with other information like priority ceiling of the resource and the link to the task that was currently holding the resource. Basically, a semaphore is a protocol mechanism for task communication. If a data item is shared by number tasks, race conditions could occur if the shared item is not protected properly. The easiest protection mechanism is a lock. **Figure 1** shows the state diagram of mutex referred to as a mutex for mutual exclusion. For every task, before it accesses the set of data items, it acquires the lock. Once the lock is successfully acquired, the task becomes the owner of that lock and the lock is locked. Then, the owner can access the protected items. After this, the owner must release the lock and the lock becomes unlocked. It is possible that while the owner is accessing one of the protected data items and another task comes.

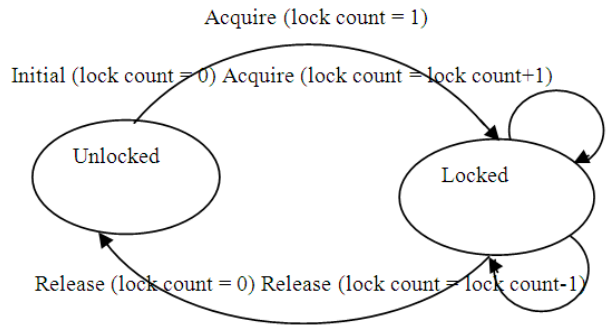

Fig. 1. State diagram of mutex

Of course, this second task must acquire that lock. However, since the lock is locked, this request is unsuccessful and the requesting task will be suspended and queued at the lock. When the lock is released by its owner, one of the waiting tasks will be allowed to continue and locks the lock.

2. RELATED WORKS

Jiang (2012) implemented a FCFS scheduler to schedule best-effort traffic on a dynamic computing system. For asynchronous best-effort networks, a scheduler was proposed based on FCFS and a combined strategy of backfilling and prediction for grid computing. When different types of data traffics with different QoS requirements share and congest a single network, Weighted Fair-Queue (WFQ) scheduler was implemented to solve the starvation problem. Different models of WFQ were implemented for networks of different types. The Generalized Processor-Sharing (GPS) model was adopted for clustered networks, where data units are in the forms of divisible tasks (sub-tasks). For packet switched networks, Packet Weighted-Fair Queue (PWFQ) scheduler was implemented that does not terminate the traffic session until it finishes the current packet. However, it may exceed the allowable bandwidth of a session. In order to deal with this problem, the worst-case fair-Weighted Fair-Queuing (WF2Q) scheduler was implemented, where each packet is checked whether it can be scheduled within the session's time slice. The Standard EDF (SEDF) scheduler was implemented to serve real-time data flows in an integrated network (Jiang, 2012). It has optimal efficiency when dealing with similar data traffics. For data streams with different QoS requirements, a modified version of the SEDF with live monitoring strategy was developed. For heavily loaded traffic, an EDF scheduler was implemented that has a pre-negotiation phase between the system and the data generators. Jagbeer Singh attempted uniform multiprocessor machine characterized by a speed or computing capacity with the interpretation that a job executing on a processor with speed s for t time units completes (s*t) units of execution. The Earliest-Deadline First (EDF) scheduling of real-time systems upon uniform multiprocessor machines is considered. It is known that online algorithms tend to perform very poorly in scheduling. Such real-time systems on multiprocessors; resource-augmentation techniques are presented here that permit online algorithms in general (EDF in particular) to perform better than may be expected given these inherent limitations.

Pratap Chandra mantel has shown the superiority of Blowfish algorithm with others in terms of the throughput, processing time and power consumption. More the throughput, more the speed of the algorithm and less will be the power consumption. Secondly, AES has advantage over the other 3DES and DES in terms of throughput and decryption time (Mandal, 2012). Third point is that 3DES has the least performance among all the algorithms mentioned here. Finally we can conclude that Blowfish is the best of all. In future we can perform same experiments on image, audio and video and developing a stronger encryption algorithm with high speed and minimum energy consumption.

3. IDENTIFIED ENVIRONMENT DESCRIPTION

In this study, for realtime implementation, ARM cortex M3 based LPC 1788 processor is chosen as it has multi-parameter acquisition, multi-level monitoring and supports networking (Jiang, 2012). It is a general purpose 32 bit processor which offers high performance and very low power consumption. The software coding for the hardware functionality is written in embedded C language. Features of LPC 1788 include:

- Running at frequencies of up to 100 MHz
- Memory Protection Unit (MPU)
- Nested Vectored Interrupt Controller (NVIC), Non-Maskable Interrupt (NMI) input
- Wakeup Interrupt Controller (WIC)
- Up to 96 kB on-chip SRAM, Up to 4 kB on-chip EEPROM
- External Memory Controller (EMC)
- DMA controller (GPDMA)
- JTAG interface, Serial Wire Debug and Serial Wire Trace Port options
- Four reduced power modes: Sleep, deep-sleep and power-down, deep power-down
- Clocks: On-chip crystal oscillator (operating range of 1 MHz to 25), 12 MHz Internal RC oscillator (IRC)

Uc/os-II is a hard realtime kernel of an open source code that has stability, reliability and the selected software build environment is keil µvision. Features of uc/os-II include. Very small realtime kernel (Kolhari and Nithin, 2012):

- Memory footprint is about 20KB for a fully functional kernel
- Highly portable, ROM able, scalable, preemptive realtime and deterministic kernel
- Connectivity with uc/GUI platform and uc/file system
- Supports all types of processors from 8bit to 64 bit

4. PROPOSED METHOD

4.1. Data Streaming with Queue Scheduler

This study provides the implementation of central queue based on EDF priority scheduler for data packet communication. Earliest Deadline First (EDF) or least time to go is a dynamic scheduling algorithm used in real-time operating systems to place tasks in a queue. As shown in **Fig. 2**. The i/p streamer consists of an input packet handler that can accept packets of variable size from multiple sources into a queue. A data stream is a sequence of digitally encoded coherent signals (packets of data or data packets) used to transmit or receive information that is in the process of being transmitted and Central queue based EDF scheduler is implemented for receiving and servicing data packets available in the FIFO queue.

4.2. Central Queue Based EDF Algorithm

Queuing is a fundamental consequence of the statistical sharing that occurs in packet networks. One way to reduce jitter might be to eliminate the statistical behavior of the sources. The central queue algorithm is such one that supports true priority scheduling on a system-wide basis. By definition, it is the only algorithm to provide such support. The other algorithms only implement priority scheduling within separate queues and not on a system-wide basis. The primary benefit of using Central Queue scheduling is its adherence to pure priority scheduling, i.e., EDF algorithm, a feature unique to the algorithm. It is not surprising that the algorithm provides the best service for high priority tasks, since the Central Queue algorithm is the only algorithm that employs system-wide priority scheduling. However, its handling of low priority tasks can be poor under high loads, when most of the migration overhead is passed on to the low priority tasks. **Figure 3** shows the model of data packet networking arriving into the queue implemented in this study. Here, $src_1^{(1)}, src_2^{(1)}$ are the data packets from source1 and source 2 respectively arriving into the queue in the FIFO manner. In packet-switched networks, the notion of a scheduling algorithm is used as an alternative to first-come first-served queuing (Abhijit and Apte, 2012). In this implementation, Task 1 is the queue filling rate and data packets are arriving into the queue and task 2 is the servicing rate of the queue. **Figure 4** Shows the state diagram of central queue model.

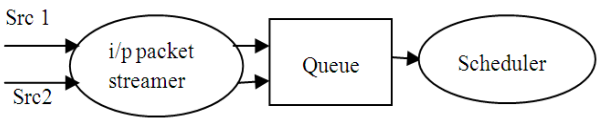

Fig. 2 I/P streamer model

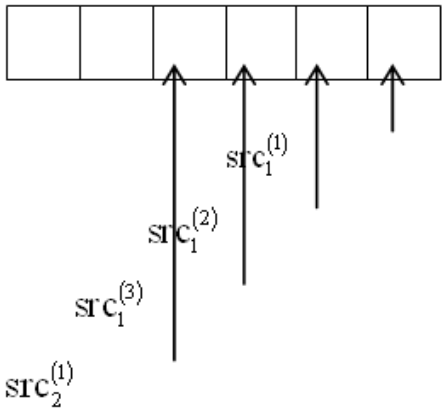

Fig. 3 Queue model

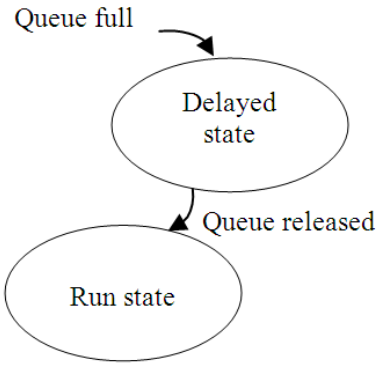

Fig. 4 State diagram

Whenever the queue is full, it is indicated by the event flag and task 1 will be in the delayed state. Now data packets are ready to be serviced, task 2 is processed and whenever the queue is released data packets are arriving, i.e., task 1 is running.

The EDF scheduling unit uses the above central queue model in which the queue will be searched for the new task closest to its deadline whenever a task is finished or new task is released. This task is the next to be scheduled for execution. This algorithm is simple and proved to be optimal when the system is preemptive, under loaded and there is only one processor. Earliest Deadline First (EDF) scheduling is a dynamic priority assignment. The priority of each task is decided based on the value of its deadline. The task with nearest deadline is given highest priority and it is selected for execution. Now task instances always get assigned a priority inverse proportional to their absolute deadline i.e., the priority is as higher as the absolute deadline is shorter (ties are broken in favor of already running task instances). This means that whenever a task instance is released the priorities have to be re-calculated and the priority of a task (i.e., of its instances) may vary during runtime. At each instance of time this task instance that currently has the highest priority among all active task instances is executed. Therefore EDF is intrinsically preemptive.

5. PERFORMANCE METRICS

5.1. Through put

Throughput is the amount of data packets moved successfully from one place to another in a given time period.

5.2. Packet Loss

It is the fraction of packets not successfully received (i.e., passed CRC check) within some time window.

5.3. Mean Service Rate

It is the ratio between speed of the channel in bits per second to the mean packet length in bits.

5.4. Queuing Delay

It is the delay between the time the packet is assigned to the queue for transmission and the time it starts being transmitted.

5.5. Transmission Delay

It is the delay between the times that the first and last data bits of a packet are transmitted.

5.6. Packet Arrival Rate

Number of packets arriving into the queue per unit time.

6. IMPLEMENTATION

6.1. Block Diagram Description

In this study, central queue based EDF scheduler is implemented under realtime environment for receiving and servicing data packets in a FIFO queue.

Figure 5 show the set up implemented using security protocol (Blowfish) and error detection coding schemes. This realtime application has been developed and run on cortex M3 LPC 1788 processor using uc/os-II.

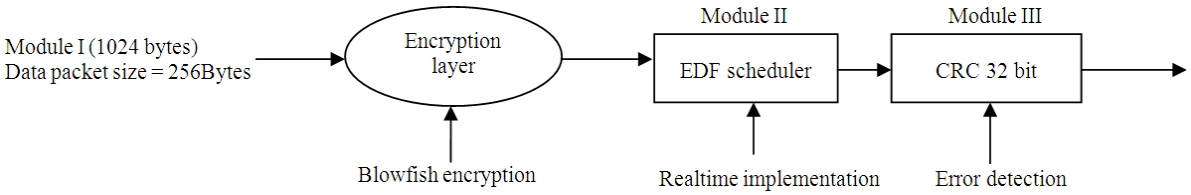

Fig. 5. Security and error detection concepts

6.2. Encryption Layer

Module I employs blowfish encryption algorithm, a symmetric block cipher that can be effectively used for encryption and Safeguarding of data. It is a 64-bit block cipher that takes a variable-length key from 32 bits to 448 bits making it ideal for securing data. It is the fast block cipher, except when changing keys. The algorithm consists of two parts: A key-expansion part and a data-encryption part. Key expansion converts a key of at most 448 bits into several sub key arrays totaling 4168 bytes. Data encryption occurs via a 16-round Feistel network. Each round consists of a key dependent permutation and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookups per round. The Feistel Network that makes up the body of Blowfish is designed to be as simple as possible, while still retaining the desirable cryptographic properties of the structure.

6.3. Scheduler Unit

The encrypted and secured data will be of 1024 bits and is processed in module II for assigning priority number which is a realtime central queue based EDF scheduler of uc/os-II.

6.4. Error Detection Module

Cyclic Redundancy Check (CRC) implemented in module III which is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. It is based on the theory of cyclic codes. The use of systematic cyclic codes, which encode messages by adding a fixed-length check value is for the purpose of error detection in communication networks. On retrieval, the calculation is repeated and corrective action can be taken against presumed data corruption if the check values do not match. In this way, all the data packets are manipulated using the above implementation (Saleh and Dong, 2013).

By implementing security and error detection schemes, the data structure for packet 1 of source 1 is shown in **Fig. 6** as an example.

6.5. Queuing Implementation

6.5.1. Decomposing Data into Stream of Packets

Figure 7.1 to 7.6 show how user data is decomposed into stream of packets by undergoing several steps from non-realtime to realtime implementation.

6.6. Priority Assignment

Table 1 and **2** Provide the information of allocating priority for source 1, source 2 and its corresponding data packets.

6.7. Scheduler Implementation

Figure 8 implements EDF scheduling based central queue algorithm under realtime environment of uc/os-II.

Table 3 and **4** explain how the scheduler implements selection of data packets for a particular time period.

6.8. Graphical Implementation

The base values chosen for packet arrival rate and packet servicing rate in this study are listed in **Table 5**.

Let task1 corresponding to streaming of src 1 and src 2 packets. The Packet arrival rate is 5 ms and 20 ms respectively. Task2 be the servicing of all packets in the queue of size 100 slots. The first task scheduled by EDF is filling rate of queue because it has shortest (5 ms) period and therefore it has earliest deadline. When task1 is completed, task2 is scheduled next as it has the deadline of 8ms which is next to task1. **Figure 9.1 to 9.4** show the status of task1, task2, queue retaining packet status for servicing 25 packets in 200 ms.

Figure 10.1 to 10.4 show the status of task1, task2, queue retaining packet status for servicing 100 packets in 800 ms.

Data → Blowfish encryption (Presentation layer) → 32bit CRC add (Physical Layer)	
256 Bytes	1024 bytes + 32 bit CRC

$Src_1^{(1)}$

Fig. 6 Data structure of packet of source 1

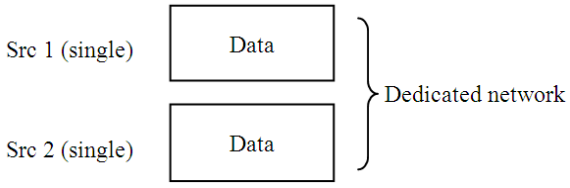

Fig 7.1. Separate channel

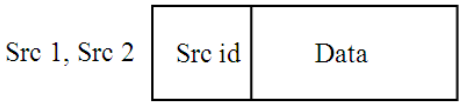

Fig .7.2. Common channel for src 1 and src 2

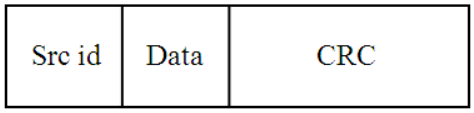

Fig. 7.3. Error detection added to data structure

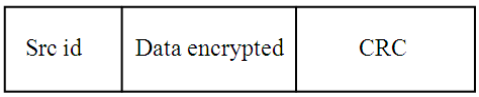

Fig. 7.4. Security and error detection

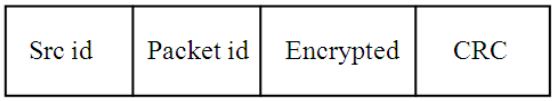

Fig .7.5. Packet id assignment (Logical channel) (To suit ATM signaling)

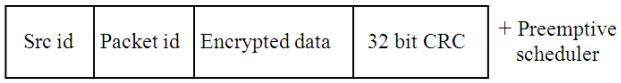

Fig .7.6. Realtime multiple source data structure

The status of serviced packets, queue slot status and different time instants are shown in **Table 6.**

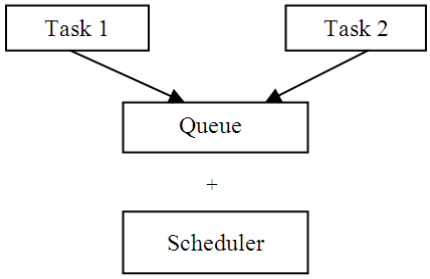

Fig. 8. Scheduler and queue realtime implementation

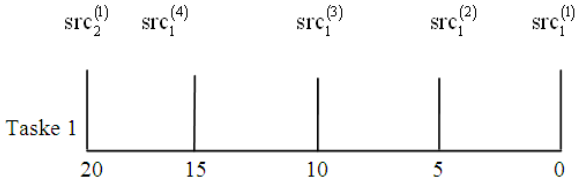

Fig. 9.1. Queue filling rate (ms)

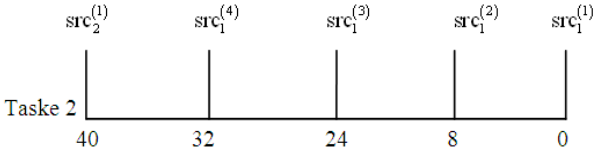

Fig. 9.2. Queue servicing rate (ms)

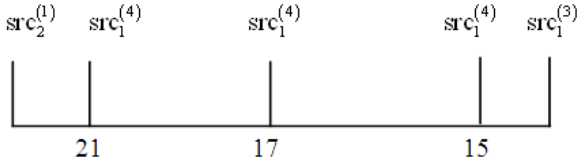

Fig. 9.3. Queue retention (ms)

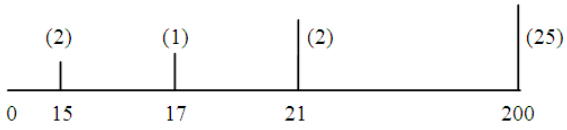

Fig .9.4. 25 packets in 200 ms

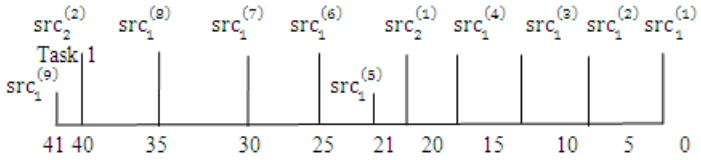

Fig. 10.1. Queue filling rate (ms)

Fig. 10.2. Queue servicing rate (ms)

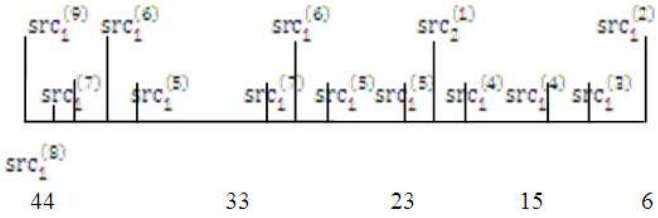

Fig. 10.3. Queue retention (ms)

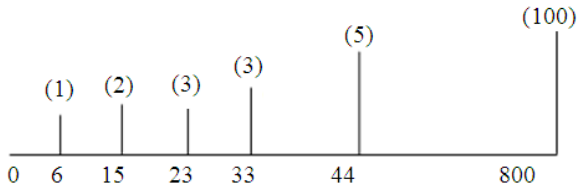

Fig. 10.4. 100 packets in 800ms (ms)

Table 1. Starting priority for multiple source packets

Sources	Priority Allocation (Initial)
Src_1	65535
Src_2	64511

Table 2. Priority of packets for source 1 and source 2

Src_1 packet ID	Priority number
1	65535
2	65534
3	65533
.	.
.	.
Src_2 packet ID	Priority number
1	64511
2	64510
3	64509
.	.
.	.

Table 3. Arrival of source 1 and source 2 packets into queue

Queue element	Packet number	Queue buffer ID	Priority number
Src_2	1	1	64511
Src_1	1	1	65535
Src_1	2	1	65534
Src_2	2	1	64510
Src_1	3	2	65533
Src_1	4	3	65532

Table 4. Queue state, packet selected Vs time

	Scheduler implementation	
Time (ms)	[Multiple packets]	Selected packet
16	Src_1^2	Src_1^2
24	Src_2^2	Src_1^3
	Src_1^3	
	Src_1^4	
32	Src_2^2	Src_1^4
	Src_1^4	
	Src_1^5	
40	Src_2^2	Src_1^5
	Src_1^5	
	Src_1^3	
	Src_1^6	

Table 5. Base values

Source	P_A (Packet arrival rate) in ms	P_{se} (Packet servicing rate) in ms	Buffer size 100 packets
$src_1^{(1)}$	5	8	
$src_1^{(2)}$	10	16	
$src_1^{(3)}$	15	24	
$src_1^{(1)}$	20	32	
$src_2^{(2)}$	40	40	

Table 6. Serviced packet status

Time (ms)	Serviced packets	Packets in queue (Dynamic) Src1	Src2
200 ms	25	40	10
1000 ms	125	200	50
800 ms	100	160	40(Queue full)

Table 7. Performance metrics

Delay (ms)	
Src1	23805
Src2	82240
Server	71000
Add queue	24900
Get queue	29740

Fig. 11. Transmitted image from Source 1

7. EXPERIMENTAL RESULTS

Figure 11 and **12** show source 1 and source 2 images transmitted as data packets as per the implemented Central queue based EDF scheduler.

Figure 13 and **14** show reconstructed packets in the receiver side after decomposing into data packets in the source side by implementing security and error detection protocols.

The implementation is tested with two TCP ports with one port as send packets and the other as receiver. The proposed algorithms are run in port 1 and the received packets along with the metrics such as throughput, queue related values and implementation/packet reception time is obtained.

Fig. 12. Transmitted image from Source 2

Fig. 13. Recovered image from source 1

Fig.14. Recovered image from source 2

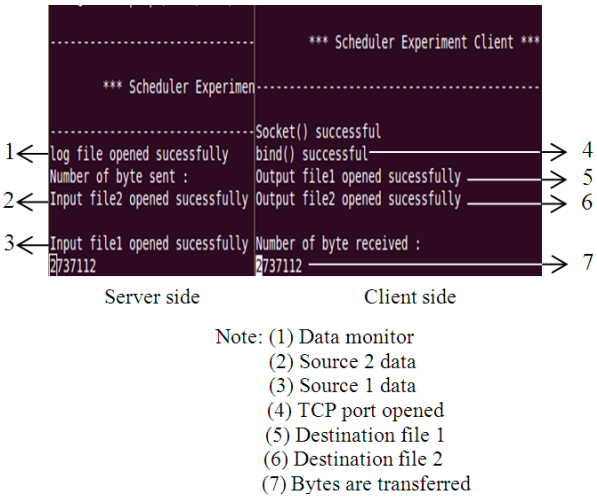

Fig. 15. Output implentation

The screen captured result is given in **Fig. 15** and the metric values are listed in **Table 7**.

8. CONCLUSION

In this study, central queue based EDF algorithm is implemented because of its optimality, i.e., the processor can be utilized fully and it has less context switches. At the same time this algorithm has less predictability and controllability. Control over the execution is very lesser and response time cannot be reduced. For future scope, by properly redesigning the hardware and enhancing features of scheduler, controllability over the execution can be easily achieved.

9. REFERENCES

Nojabaei, S., Z. Leman, S.H. Tang and S. Sulaiman, 2012. Development of priority oriented scheduling method to increase the reliability of manufacturing systems. Am. J Applied Sci., 9: 1435-1442. DOI: 10.3844/ajassp.2012.1435.1442

Keerthika, P. and N. Kasthuri, 2012. An efficient fault tolerant scheduling approach for computational grid. Am. J. Applied Sci., 9: 2046-2051. DOI: 10.3844/ajassp.2012.2046.2051

Abhijit, A. and S.S. Apte, 2012. A comparative performance analysis of load balancing algorithms in distributed systems using qualitative parameters. Int. J. Recent Techn. Eng., 1: 175-179.

Mandal, P.C., 2012. Superiority of blowfish algorithm. Int. J. Adv. Res. Comput. Sci. Software Eng., 2: 196-201.

Saleh, M. and L. Dong, 2013. Realtime scheduling with security enhancement for packet switched networks. IEEE Trans. Netw. Service Manage., 10: 271-285. DOI: 10.1109/TNSM.2013.071813.120299

Kolhari, N.R. and I.B. Nithin, 2012. Porting and implementation of features of uc/os II RTOS on ARM 7controller LPC 2148 with different IPC mechanisms. Int. J. Eng. Res. Techn., 1: 2278-0181.

Abt, A.R. and K. Thomas, 2013. ARM based embedded web servers for industrial applications. Proceeding of the International Conference on Computing and Control Engineering, Apr. 12-13. Coimbatore Institute of Information Technology.

Jiang, W., 2012. Resource Allocation of security-critical tasks with statistically guaranteed energy constraint embedded and realtime computing systems and applications. Proceeding of the 18th IEEE International Conference, pp: 330-339. DOI: 10.1109/RTCSA.2012.34