Improved round robin queue management algorithm for elastic and inelastic traffic flows

S. Nandhini

School of Advanced Sciences,
VIT University,
Vellore, India
Email: nandhini.s@vit.ac.in
Email: nandhuraja@hotmail.com

Abstract: In current scenario, network traffic flows need a start-time fair queuing algorithm which is computationally efficient and also which can achieve maximum fairness regardless of variation in a network capacity. To enhance the situation of congestion, an improved round robin (IRR) queue management algorithm for elastic and inelastic traffic flows is proposed. In this approach, the traffic flows are categorised into elastic traffic flows and in-elastic traffic flows. The scheduling process in the inelastic flows is handled by the BRR scheduler algorithm since they have large capacity requirements and delay constraints and elastic flows will be scheduled using DRR-SFF. The results are simulated with NS-2 and they show consistent improvement in the performance of the network.

Keywords: real time traffic flow; non-real time traffic flow; inelastic flows; elastic flows; round robin queue; DRR-SFF scheduler; BRR scheduler; NS2; performance enhancement in networks; advanced queue models.

Reference to this paper should be made as follows: Nandhini, S. (2015) ‘Improved round robin queue management algorithm for elastic and inelastic traffic flows’, *Int. J. Mobile Network Design and Innovation*, Vol. 6, No. 2, pp.108–113.

Biographical notes: S. Nandhini is currently an Assistant Professor in the School of Advanced Sciences at VIT University, Vellore, India. She received her PhD in Science and Humanities from Anna University, Chennai, India. Her main areas of research interest are queuing theory particularly in communication networks, graph theory and fuzzy logic.

1 Introduction

Today, the traditional end-to-end network protocols cannot guarantee the fair allocation of network resources in current high bandwidth-delay-product networks. To support this issue, Fair queuing algorithm, which is a scheduling algorithm used in telecommunication networks to allow multiple packet flows to share the link capacity fairly and to ensure minimum throughput for users or flows sharing a wire line link (Khawam and Kofman, 2006) is implemented. When VBR video sources and data sources of integrated service networks coexist, the bandwidth for data applications may vary with time. Hence, fairness property must ignore variation in server capacity (Goyal et al., 1996). Unfair scheduling algorithms penalise channels for the use of idle bandwidth and do not provide any QoS guarantee in the presence of congestion. Fair scheduling algorithms ensure allocation of bandwidth fairly regardless of prior usage or congestion to enable throughput-intensive, flow-controlled applications in heterogeneous, large-scale, decentralised networks. Hence, fair scheduling algorithms are desirable for video applications (Goyal et al., 1996; Lin and Hamdi, 2010). A weighted version of fair queuing is called weighted fair queuing (WFQ). WFQ allocates an equal share of bandwidth to each flow. The variations of fair queuing includes, class-based weighted fair queuing (CBWFQ) and low latency queuing (LLQ) which are queuing methods based on WFQ (Balchunas, 2010).

2 Proposed queuing model

2.1 Overview

Deficit round robin with short flow first (DRR-SFF) (Sun et al., 2007) is a novel scheduling mechanism used to enhance the performance of short flows with limited penalising long flows. DRR-SFF uses weighed DRR for scheduling short and long flows respectively and treats long flows more fairly. The mean transmission time and loss rate of short flows under DRR-SFF are significantly reduced comparing to FIFO scheduling using drop tail is revealed through trace-driven simulation. Flows are organised as two groups, prioritised group (PQ) and best-effort group (BQ).

In Lin and Hamdi (2010), a two-stage FQ algorithm called ‘budget round robin’ (BRR) is proposed. It works in two steps. First, a high-bandwidth, high-storage buffer is designed using multiple DRAMs. It further divides the entire storage space into separated blocks, each block is of megabytes in size and these blocks are organised as circular linked list. Second, newly arrived packets are pushed into these blocks under the control of BRR while the outputs are popped from DRAMs continuously.

In our case, after categorising the flows as elastic (non-real time) and inelastic (Nandhini and Palaniammal, 2013a, 2014), the inelastic flows having large capacity requirements and delay constraints will be scheduled using BRR and the elastic flows will be scheduled using DRR-SFF.

Figure 1 Architectural diagram

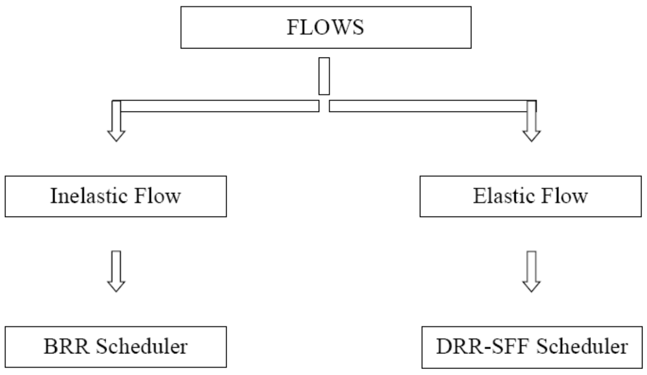


Figure 1 illustrates the overall process of our proposed queuing model, where the flows are differentiated into two types, elastic and inelastic flows. Based on the differentiated flows, the queuing algorithms BRR and DRR-SFF are implemented. The inelastic flows are scheduled with BRR scheduler and the elastic flows with DRR-SFF algorithm.

2.2 Flow classification – a remainder

The flow classifier identifies the ingress traffic flow as inelastic or elastic based on the estimated delay and loss as in Nandhini and Palaniammal (2013b).

The egress updates the average packet delay, PD_{avi} for delay sample $D_i(t)$ at time t using an exponential weighted moving average (EWMA).

$$PD_{avi}(t) = \mu * PD_{avi}(t-1) + (1-\mu) * D_i(t) \tag{1}$$

where μ is a small fraction $0 \le \mu \le 1$ to emphasise recent history rather than the current sample alone.

At egress router, the difference in loss ratios can be then estimated as,

$$D = L^T_{act} - L^T \tag{2}$$

where (L^T_{act}) is the actual loss ratio and L^T is the measured loss ratio at the interval T.

If the value of loss ratio [as per equation (2)] exceeds to a threshold T_1 and if the delay [as per equation (1)] exceeds a threshold T_2, then the flows are marked as inelastic flows by the egress node, otherwise they are considered as elastic traffic.

2.3 Deficit round robin with short flow first

DRR-SFF is a fair scheduling scheme which is widely deployed. DRR-SFF can be considered as the round robin scheduling for variable-length packets. DRR-SFF inherits the characteristics of DRR and may be simply enforced by hardware or software system. Flow size with a less threshold T is known as short flows and otherwise they are known as long flows. (ie)

$$\text{Flow} = \begin{cases} \text{Short}, & FS < T \\ \text{Long}, & \text{Otherwise} \end{cases} \tag{3}$$

where FS is the flow size.

Here, FS represents the total data packets in a flow. The network divides the queues into two groups, prioritised queue group (PQ) and best-effort queue group (BQ). Each flow is first put n to PQ and then moved to BQ after T^{th} byte have been scheduled. As PQ is assigned higher priority than BQ, flows with flow size less than T^{th} bytes are favoured, while long flows in BQ would not be starved since PQ does not have strict priority over BQ. Both these PQ and BQ serve flows using a DRR discipline. Every group in the network is offered services proportional to its allocated weight. For each group, a deficit counter (DC) is associated. This DC is incremented by a quantum for every round of scheduling which represents the weight of the group.

$$\text{(ie)} \quad DC_i = DC_i + Q_i \tag{4}$$

where DC_i is the DC of flow i and Q_i is the quantum of flow i given by

$$Q_i = (R_i / C) \times F \tag{5}$$

Here, R_i is the rate of the flow i, C is the service rate of the link and F is the frame size given by

$$F = \sum_{i=1}^{n} Q_i \tag{6}$$

The group in the network is been served as long as the DC is greater than the zero. The DC is decremented by the number of bytes served and which is carried over to succeeding spherical.

$$DC = DC - NBS \tag{7}$$

where NBS is the number of bytes served.

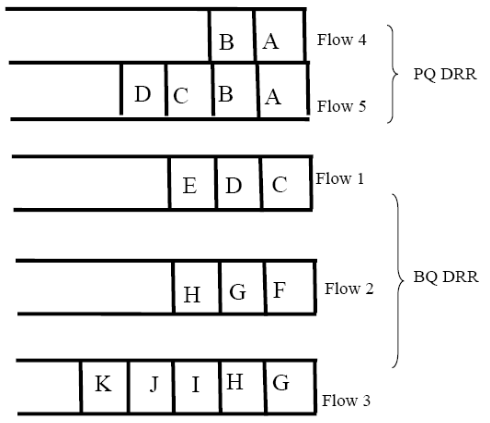
The DC is immediately reset to zero when the group becomes empty.

$$\text{(ie) } DC = 0, \text{ if } PQ = \phi \text{ or } BQ = \phi \tag{8}$$

By setting their quanta the priority of the two groups is determined through this weighted DRR.

Figure 2 illustrates DRR-SFF working in the network. Here, the DRR-SFF uses T = 2 (threshold is two packets) during the scheduling in the network. Let us assume that the external buffer contains five flows. The flows 1, 2 and 3 are served in BQ since they are short parts of flow. In the PQ, the news flows 4 and 5 are served. The flow 4 will be emptied after the first two packets of flow 4 are served. The emptied flow 4 will be removed and flow 5 would be moved to BQ.

Figure 2 DRR-SFF



Algorithm of enqueueing and dequeueing module of DRR-SFF

```
Initiation: on arrival of packet p
high_low_turn = 1;
FreeBuffer();                  //if no free buffers left
i = ExtractFlow(p);
i→pkts ++;
if(i→pkts == 1)                //new flow
i→DC = Q;
// Q is the Quantum
if(i→bytes_scheduled < T)      //active in PQ group
i←InsertActivelist_PQ;
Else // active in BQ group
i←InsertActvielist_BQ;
Enqueue(i,p);                  //enqueue packet p to queue i
```

In the enqueuing algorithm, if the buffer is indicated as full and when a new data packet has arrived, the network drops the data packet using buffer stealing. Buffer stealing drops the first data packet from BQ, then from PQ. A flow ‘i’ is introduced at the end of the active list of PQ, when the flow has only one packet and its bytes scheduled is below the threshold T. If in case bytes scheduled are not below the threshold T then the flow with a single data packet is added at the end of BQ.

To implement the dequeueing algorithm the PQ or BQ should be null.

```
if(high_low_turn && AL_PQ) ≠ null;    //AL is the activelist
if(high_DC <= 0)
high_DC += Q_high_;
if(i→DC <= 0) i->DC += Q;             //Remove the head of
                                      AL_PQ(i)
pkt=Dequeue(i);
Packetsize = Length(pkt);
high_DC-= Packetsize;
i→DC-= Packetsize;
if(high_DC <= 0) high_low_turn = 0;
i→bytes_scheduled += Packetsize;
if(i→bytes_scheduled >= T) {
if(empty i)
eliminate i from AL_PQ
else transfer i to AL_BQ
}else if(i→DC <=0 )
transfer i to AL_PQ tail
else if(!high_low_turn && AL_BQ) ≠ null;
if(low_DC <= 0) low_DC+=Q_low;
if(i→DC <= 0) i→DC += Q;              //Remove the head of
                                      AL_BQ(i)
pkt=Dequeue(i);
Packetsize = Length(pkt);
low_DC-= Packetsize;
i→DC-= Packetsize;
if(low_DC <= 0) high_low_turn = 1;
if(empty i)
eliminate i from AL_BQ
if(i→DC <=0)
transfer i to AL_BQ tail
```

In the dequeuing algorithm, DC is added by PQ’s quantum when the DC is not more than zero. By subtracting the bytes of packet p, from flow ‘i’s’ DC, PQ active list is dequeued. If in case flow i’s DC is not more than zero, the entry head will be changed to the following entry of flow ‘i’ in the PQ active list. If DC of PQ is not more than zero, then the next turn is BQ else data packets will be scheduled in the PQ. During PQ’s scheduling, the scheduled flow bytes is not less than the threshold T and not empty then the flow is moved to BQ. The scheduling in BQ is same as that in PQ.

2.4 Budget round robin

BRR technique tries to save the received packets in their output series. BRR keeps the track of the buffered packets and records the connections which are in active and then the BRR estimates the storage quota for each connection which is active. When a new data packet is received, the BRR assumes the number active connections are stable and target block number is calculated. Thus, once the storage quota for the present block is exhausted, the available space of succeeding block is allotted before hand. BRR omits the trivial connections by setting up a threshold value in order to maintain the accuracy of such estimation. The connections with larger buffered data size than threshold will be considered as active connection. Moreover, BRR additionally prevents any connection from assembling an unnecessary quantum. Quantum which is maximal accumulated has been strictly restricted, so the algorithm can be precise the inaccuracy as soon as the connections are updated which are active.

Q_i is the current quantum, $L_R(i)$ is the last paid round, $T_B(i)$ is the total number of bytes and MAQ is the maximal accumulated quantum.

```
Enqueuing:
Initialization:
Assume ($Q_i$, $L_R(i)$, $T_B(i)$) to zeros for all i.
Define NewArrivalPacket as p; i = GetConnectionID(p);
UpdataConnection(p, i);
Qi = (($Q_i$ + ($T_B(i)$ – $L_R(i)$) * QU) > MAQ)? MAQ: ($Q_i$ + ($T_B(i)$ – $L_R(i)$) * QU);
$L_R(i)$= $T_B(i)$;
if($Q_i \geq$ GetPacketSize(pkt)) then
$b_n$ = ($T_B(i)$> INPUTPointer)? $T_B(i)$: INPUTPointer;      //bn is the blocknum
else $b_n$ = $T_B(i)$ + Ceil((GetPacketSize(pkt) – $Q_i$) / QU);
SaveToBlock(pkt, $b_n$); $T_B(i)$ = $b_n$;
if the INPUTPointer-th block is full, then
update INPUTPointer to the next non-full block in ascending order;
```

```
Dequeuing:
Initialization:
ActiveConnection=0; INPUTPointer=0; OUTPUTPointer=0;
Get one packet pkt from the OUTPUTPointer-th block;
i = GetConnectionID(pkt);
Updata ActiveConnection (pkt, i);
if (OUTPUTPointer == (INPUTPointer-1)) then
update INPUTPointer to the next non-full block in ascending order;
if (OUTPUTPointer-th block is null)
OUTPUTPointer++;
```

3 Simulation results

3.1 Simulation model and parameters

In this section, we examine the performance of our improved round robin (IRR) technique with an extensive simulation study based on network simulator (NS-2). We compare the results with the drop tail queue technique. The topology used in the simulation is shown in Figure 3.

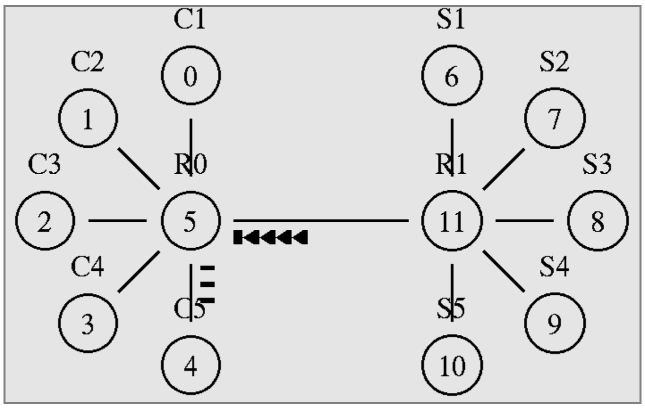
3.2 Performance metrics

In the simulation experiments, for elastic (non-real-time), TCP traffic is used. For inelastic (real-time), VoIP and video traffic are used. The traffic rate is varied from 100 Kb to 500 Kb. We measure the following metrics.

- throughput
- delay.

The results are described in the next section.

Figure 3 Simulation topology



3.3 Results

3.3.1 Results based on VoIP traffic

In this experiment, two set of TCP and three set of VoIP flows are used.

Figure 4 Rate vs. delay (VOIP) (see online version for colours)

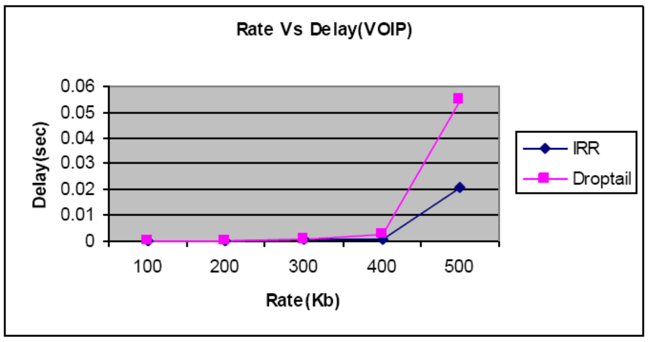
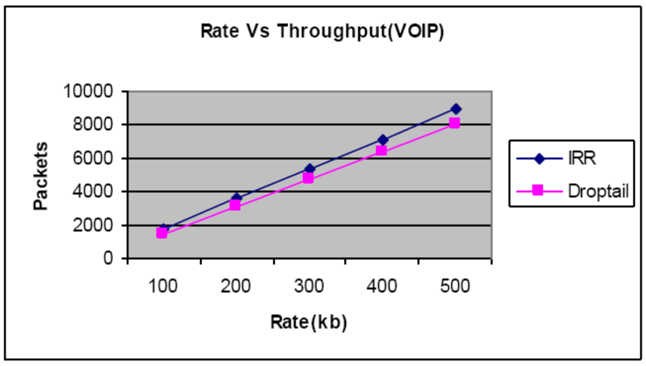


Figure 5 Rate vs. throughput (VOIP) (see online version for colours)



From Figures 4 and 6, we can see that the delay (in case of VoIP and TCP respectively) of IRR model is less than drop tail technique and from Figures 5 and 7, it is clear that the received bandwidth (VoIP and TCP, respectively) of IRR model is fairly higher than drop tail technique.

Figure 6 Rate vs. delay (TCP) (see online version for colours)

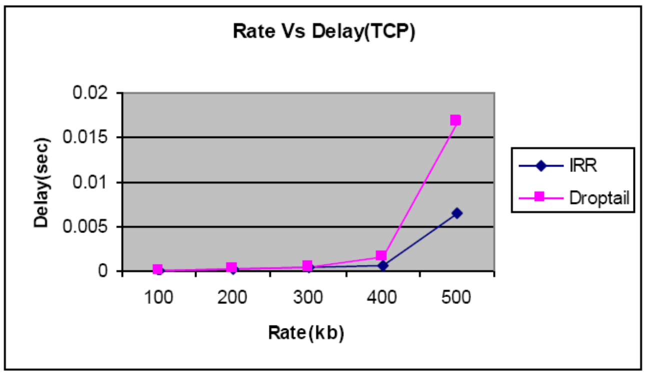


Figure 7 Rate vs. throughput (TCP) (see online version for colours)

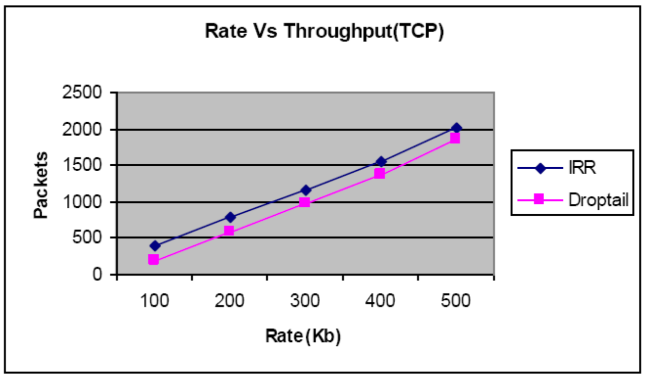


Figure 8 Rate vs. delay (video) (see online version for colours)

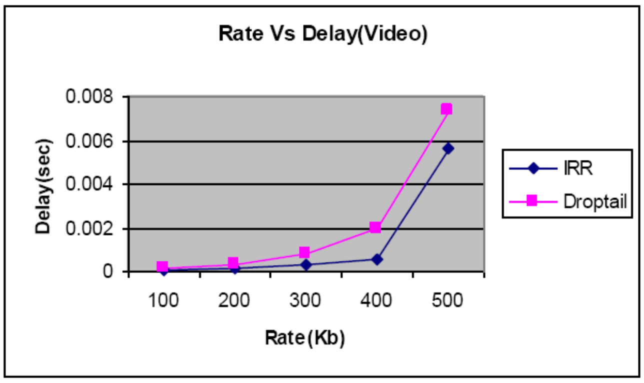
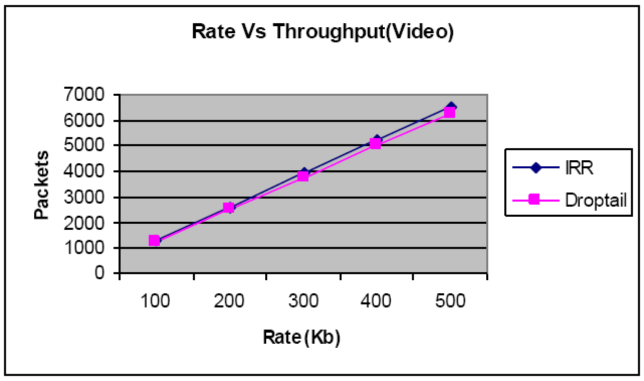


Figure 9 Rate vs. throughput (video) (see online version for colours)



3.3.2 Results based on video traffic

In this experiment, two set of TCP and three set of video flows are used.

Figures 8 and 10 show less delay if IRR model is implemented when compared with drop tail technique. From Figures 9 and 11, it is clear that the received bandwidth of IRR higher than drop tail technique in case of video flows.

Figure 10 Rate vs. delay (TCP) (see online version for colours)

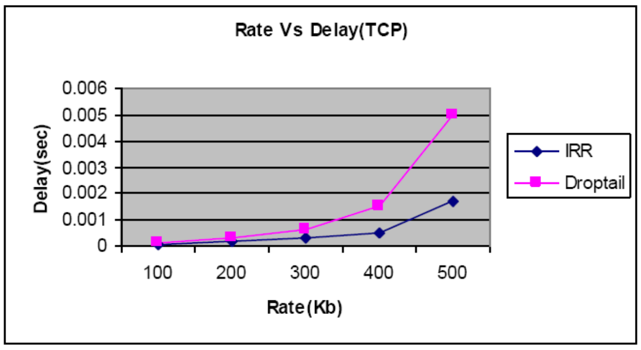
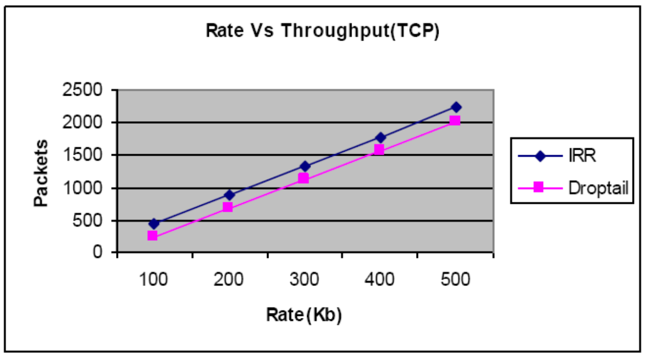


Figure 11 Rate vs. throughput (TCP) (see online version for colours)



4 Conclusions

In this paper, an IRR queue management algorithms for elastic and inelastic traffic flows has been proposed. This approach is proposed in order to achieve maximum fairness independent of deviation in network capacity. In this approach, the traffic flows are scheduled by the two kinds of scheduler algorithm which are BRR scheduler algorithm and DRR-SFF scheduler algorithm. BRR scheduler algorithm schedules the inelastic traffic flows and DRR-SFF scheduler algorithm schedules the elastic traffic flows. Through this approach, it is possible to achieve fairness in the network for throughput-intensive and also this approach is computationally efficient.

References

Balchunas, A. (2010) 'QoS and queuing', *Router Alley Description*, v1.31, ©2010.

Goyal, P., Vin, H.M. and Cheng, H. (1996) 'Start-time fair queuing: a scheduling algorithm for integrated services packet switching networks', in the *Proceedings of SIGCOMM 1996.*

Khawam, K. and Kofman, D. (2006) 'Opportunistic weighted fair queueing', *Vehicular Technology Conference, IEEE 64th*, Montreal, Que, pp.1–5.

Lin, D. and Hamdi, M. (2010) 'Two-stage fair queuing using budget round-robin', *Proceedings of IEEE, ICC 2010.*

Nandhini, S. and Palaniammal, S. (2013a) 'Fuzzy based congestion detection technique for queuing in IP networks', *International Review on Computers and Software (IRESCOS)*, April, Vol. 8, No. 4, pp.941–948,

Nandhini, S. and Palaniammal, S. (2013b) 'Stateless aggregate fair marking scheduler for differentiated service networks', *Journal of Computer Science*, Vol. 9, No. 1, pp.63–73.

Nandhini, S. and Palaniammal, S. (2014) 'Enhanced core stateless fair queuing with multiple queue priority scheduler', *International Arab Journal of Information Technology*, March, Vol. 11, No. 2, pp.159–167.

Sun, C., Shi, L., Hu, C. and Liu, B. (2007) 'DRR-SFF: a practical scheduling algorithm to improve the performance of short flows', *Proceeding of International Conference on Networking and Services (ICNS '07)*, Athens, Greece.