ECE 642- Design and Analysis of Computer Networks

Lecture 15: WFQ and DRR

Prof. Bijan Jabbari, PhD
Dept. of Electrical and Computer Eng.
George Mason University
Fairfax, VA 22030-4444, USA

bjabbari@gmu.edu
http://cnl.gmu.edu/bjabbari

December 2007

WFQ (Weighted Fair Queueing)

- A scheduling method implemented at the server to schedule packets on the output link of a router or switch in a store-and-forward packet network
- Provides a fair method to allocate bandwidth and buffer space to different users
- The ideal way to guarantee a fair allocation of bandwidth is to use the bit-by-bit Round Robin which will assure each non-empty queue will get a turn of service to transfer one bit at each round

Important Relations for WFQ

For each flow α:

P_i^{α} : the length of packet i (or how many clock ticks it takes to transmit packet i)

$R(t)$: the number of rounds in the round-robin service within time t

S_i^{α} : the value of R(t) when packet i started service

F_i^{α} : the value of R(t) when packet i finished service

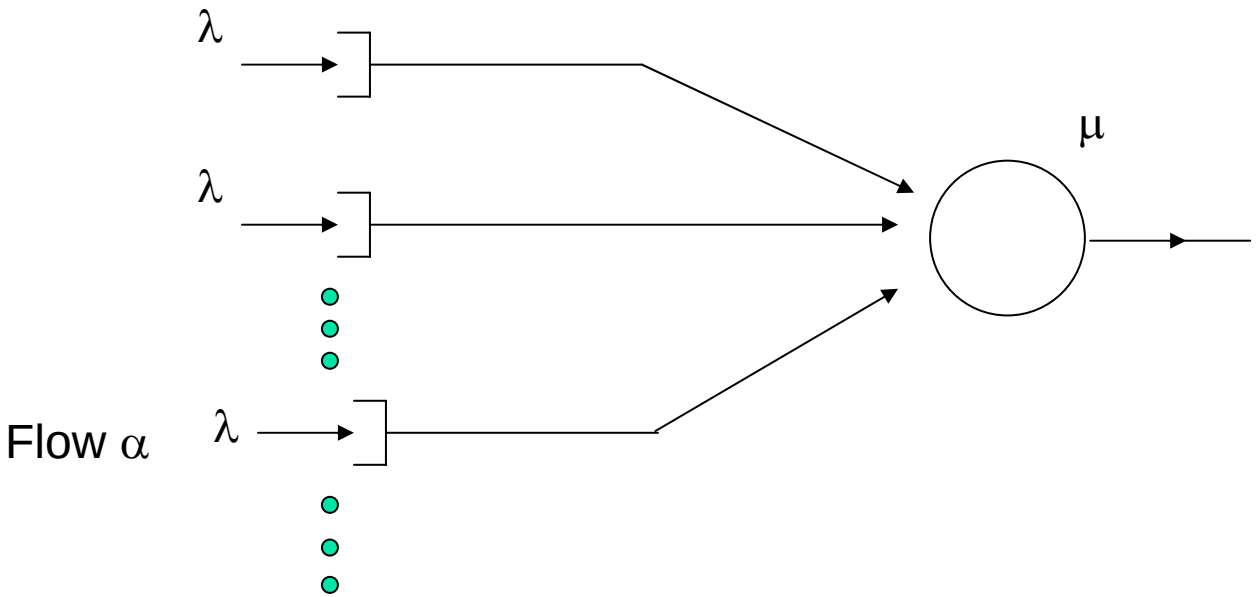
$N_{ac}(t)$: the number of flows active

L : line speed of the output link (assume equal to 1)

$$\frac{\partial R}{\partial t} = \frac{L}{N_{ac}(t)} \qquad S_i^{\alpha} = \max\left(F_{i-1}^{\alpha}, R(t_i^{\alpha})\right) \qquad F_i^{\alpha} = S_i^{\alpha} + P_i^{\alpha}$$

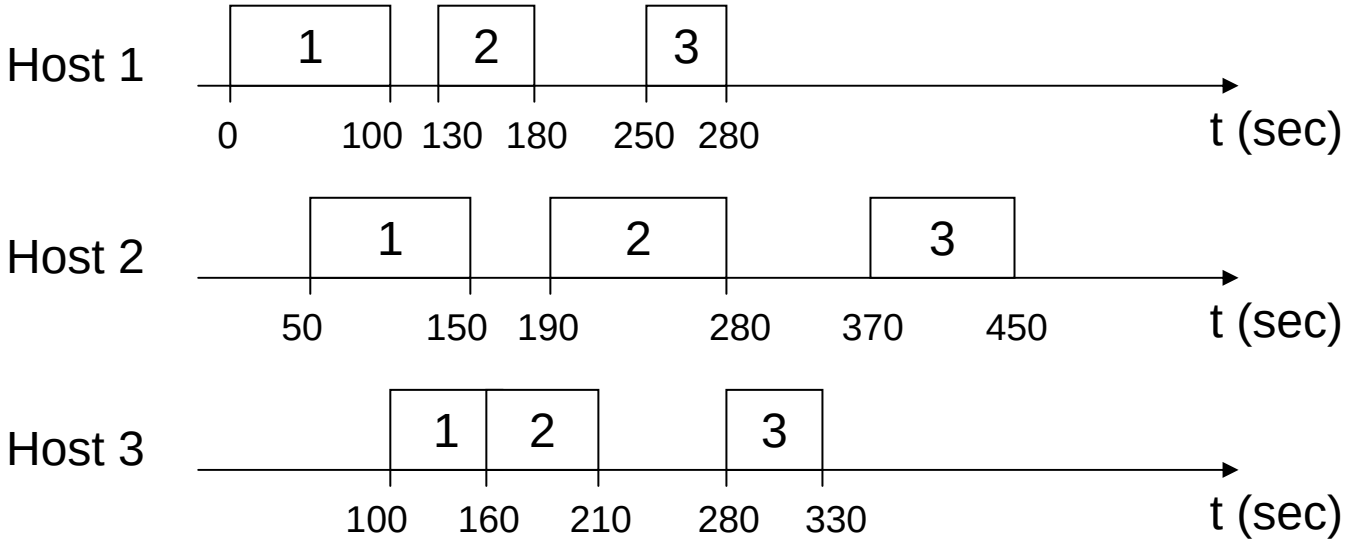
WFQ

- The server schedules transfer of packets for each flow



Implementation of WFQ

- In reality, the implementation of the fair queueing algorithm leads to finding the F_i^{α} of each packet i belonging to flow α which defines the sending order of the packets
- Consider the packet flow arrivals as in the example below:



Description of Implementation

- Starting with first packet in flow 1 we have

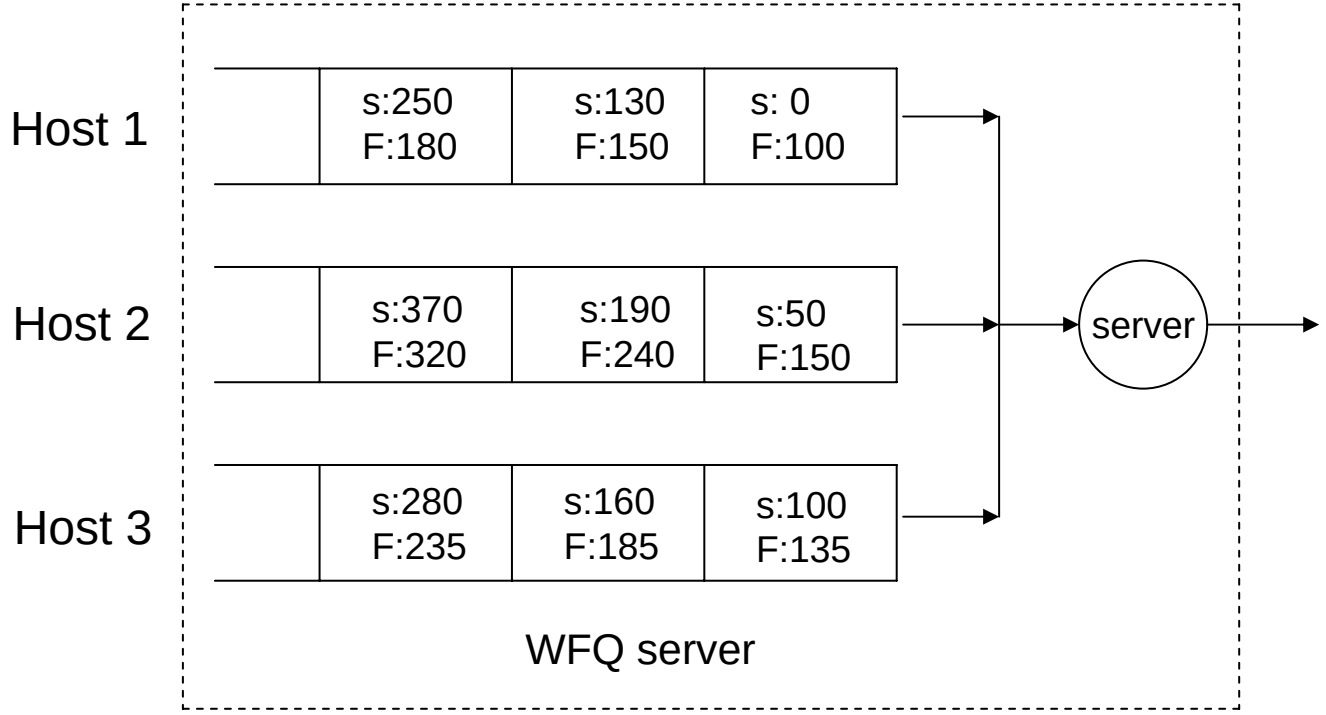
R(0)=0, then S_1^1 =R(0)=0; P_1^1 =100 and $F_1^1 = S_1^1 + P_1^1$ =100

- In flow 2, for first packet R(50)=50, S_1^2 = R(50)= 50; P_1^2 =100 then, finish round number for it will be F_1^2=150
- In flow 3, for first packet R(100)= 75,

S_1^3 =R(100)=75 (one active flow between 0 and 50 and two active flows between 50 and 100); P_1^3 =60 then, finish round number for it will be F_1^3=75+60=135

Since F_1^3 (=135)< F_1^2 (=150), then packet 1 of flow 3 will be sent before packet 1 of flow 2

WFQ (example)



s: arrival time of the packet (in sec)
F: round number when finish serving the packet (in round)

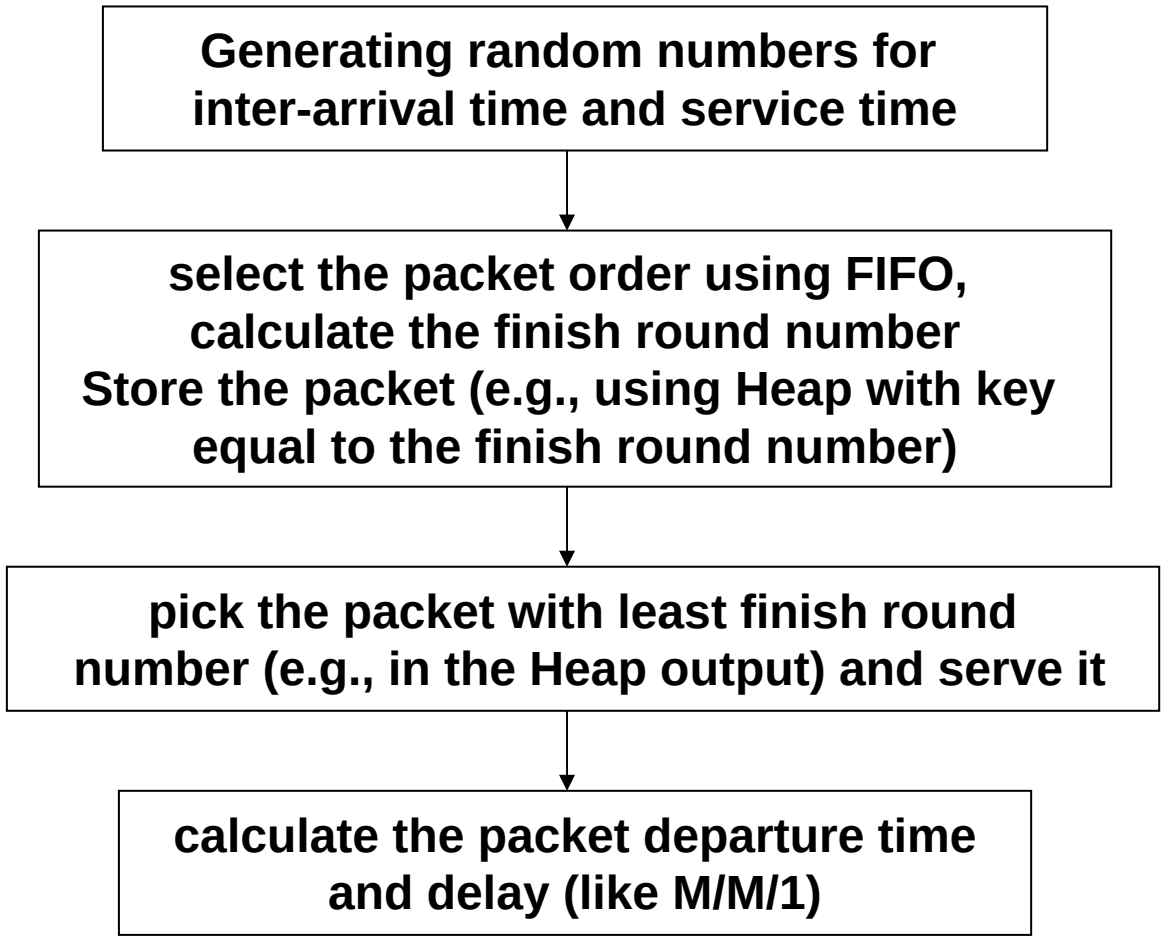
Transmission using WFQ (example)

Sending order of packets:

	H:2 PN:3	H:2 PN:2	H:3 PN:3	H:3 PN:2	H:1 PN:3	H:1 PN:2	H:2 PN:1	H:3 PN:1	H:1 PN:1

H: Host ID
PN: packet ID

WFQ (implementation flow chart)



To compute finish round number

/* *F*, D and *N* are temporary variables */
$N = N_{ac}(t_{chk})$;
do {
    $F = MIN(F_a \mid a\ is\ active)$;
    $D = t - t_{chk}$;
    if ($F <= R_{chk} + D * L / N$) {
        declare the conversation with $F_a = F$ inactive;
        $t_{chk} = t_{chk} + (F - R_{chk}) * N / L$;
        $R_{chk} = F$;
        $N = N - 1$;
    }
    else {
        $R(t) = R_{chk} + D * L / N$;
        $R_{chk} = R(t)$;
        $t_{chk} = t$;
        $N_{ac}(t) = N$
        exit;
    }
}

DRR (Deficit Round Robin)

- The difference between the traditional Round Robin and deficit Round Robin are as follows
  - If a queue is not able to send a packet in the previous round because its packet size is too large will get bigger quantum for sending packet in next round.
  - The quantum is equal to the remainder of the previous round added to the quantum of the next round which in this case will be at least two times the quantum of each round.
  - If a queue is sending a packet in the previous round whose packet size is smaller than given quantum, the remainder of this round will be added to the quantum of the next round.

Important parameters for DRR

- For each flow i:
  - Q_i : bits allocated in each round
  - DC_i: deficit counter maintaining the remainder from the previous round
  - $bytes_i(k)$: the number of bytes sent out for queue i in round k

$$bytes_i(k) \le Q_i$$

For the initial round: assume

- if (queue i is empty after this round) $DC_i = 0$
- else $DC_i = Q_i - bytes_i(k)$

DRR (Example)

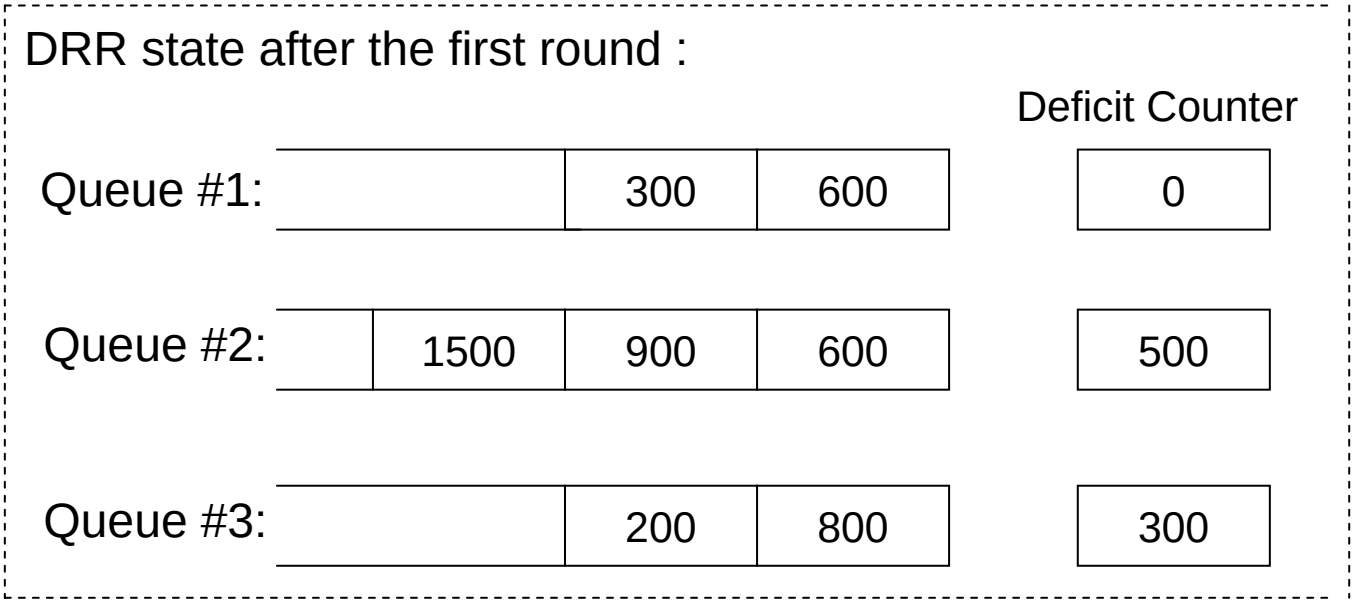
- Consider the example shown below:

Initial state :

					Deficit Counter
Queue #1:		300	600	500	0
Queue #2:		1500	900	600	0
Queue #3:		200	800	200	0

DRR(Example)

DRR state after the first round :



DRR(Example)

DRR state after the second round :

				Deficit Counter
Queue #1:		300	600	500
Queue #2:		1500	900	400
Queue #3:			200	0

References

- “Analysis and Simulation of a Fair Queueing Algorithm”, Demers, A., Keshav, S. and Shenker, S., 1989
- “On the efficient implementation of Fair Queueing”, Srinivasan Keshav, 1991
- “Efficient Fair Queueing Using Deficit Round-Robin”, M. Shreedhar and George Varghese, 1996