Computing and communication on a cloud are intimately related. Therefore, it should be no surprise

that the first algorithm we discuss can be used for scheduling packet transmission as well as threads.

Interconnection networks allow cloud servers to communicate with one another and with users. These

networks consist of communication links of limited bandwidth and switches/routers/gateways of limited

capacity. When the load exceeds its capacity, a switch starts dropping packets because it has limited

input buffers for the switching fabric and for the outgoing links, as well as limited CPU cycles.

A switch must handle multiple flows and pairs of source-destination endpoints of the traffic. Thus, a

scheduling algorithm has to manage several quantities at the same time: the bandwidth, the amount of

data each flow is allowed to transport; the timing when the packets of individual flows are transmitted;

and the buffer space allocated to each flow. A first strategy to avoid network congestion is to use a FCFS

scheduling algorithm. The advantage of the FCFS algorithm is a simple management of the three quantities:

bandwidth, timing, and buffer space. Nevertheless, the FCFS algorithm does not guarantee fairness;

greedy flow sources can transmit at a higher rate and benefit from a larger share of the bandwidth.

To address this problem, a fair queuing algorithm proposed in [252] requires that separate queues, one

per flow, be maintained by a switch and that the queues be serviced in a round-robin manner. This algorithm

guarantees the fairness of buffer space management, but does not guarantee fairness of bandwidth

allocation. Indeed, a flow transporting large packets will benefit from a larger bandwidth (see Figure 6.8).

The fair queuing (FQ) algorithm in [102] proposes a solution to this problem. First, it introduces

a bit-by-bit round-robin (BR) strategy; as the name implies, in this rather impractical scheme a single

bit from each queue is transmitted and the queues are visited in a round-robin fashion. Let R(t) be the

number of rounds of the BR algorithm up to time t and Nactive(t) be the number of active flows through

the switch. Call ta

i the time when the packet i of flow a, of size Pa

i bits arrives, and call Sa

i and Fa

i the

values of R(t) when the first and the last bit, respectively, of the packet i of flow a are transmitted. Then,

Fa

i

= Sa

i

+ Pa

i and Sa

i

= max

Fa

i−1, R

ta

i

. (6.28)

The quantities R(t), Nactive(t), Sa

i , and Fa

i depend only on the arrival time of the packets, ta

i , and not

on their transmission time, provided that a flow a is active as long as

R(t)   Fa

i when i = max

j |ta

i   t

. (6.29)

The authors of [102] use for packet-by-packet transmission time the following nonpreemptive scheduling

rule, which emulates the BR strategy: The next packet to be transmitted is the one with the smallest

Fa

i . A preemptive version of the algorithm requires that the transmission of the current packet be

interrupted as soon as one with a shorter finishing time, Fa

i , arrives.

A fair allocation of the bandwidth does not have an effect on the timing of the transmission. A possible

strategy is to allow less delay for the flows using less than their fair share of the bandwidth. The same

paper [102] proposes the introduction of a quantity called the bid, Ba

i , and scheduling the packet

transmission based on its value. The bid is defined as

Ba

i

= Pa

i

+ max

Fa

i−1,

R

ta

i

− δ

, (6.30)

with δ a nonnegative parameter. The properties of the FQ algorithm, as well as the implementation of

a nonpreemptive version of the algorithms, are analyzed in [102].