As mentioned repeatedly, cloud computing is intrinsically dependent on communication; thus, network

resource management is a very important aspect of the management of computer clouds. A critical aspect

of resource management in cloud computing is to guarantee the communication bandwidth required by

an application as specified by a service-level agreement (SLA). The solutions to this problem are based

on the strategies used for some time on the Internet to support the quality-of-service (QoS) requirements

of data streaming.

First, we discuss the stochastic fairness queuing (SFQ) algorithm [240], which takes into account

data packet sizes to ensure that each flow has the opportunity to transmit an equal amount of data.

SFQ is a simpler and less accurate implementation of fair queuing algorithms and thus requires fewer

calculations. The fair queuing (FQ) algorithm discussed in Section 6.9 ensures that a high-data-rate

flow cannot use more than its fair share of the link capacity. Packets are first classified into flows by the

system and then assigned to a queue dedicated to the flow; queues are serviced one packet at a time in

round-robin order, as shown in Figure 7.7. FQ’s objective is max–min fairness; this means that first, it

maximizes the minimum data rate of any of the data flows, then it maximizes the second minimum data

rate, etc. Starvation of expensive flows is avoided, but the throughput is lower.

Next we review a widely used strategy for link sharing, the class-based queuing (CBQ) method

proposed by Sally Floyd and Van Jacobson in 1995 [125]. The objective of CBQ is to support flexible

link sharing for applications that require bandwidth guarantees such as VoIP, video streaming, and audio

streaming. At the same time, CBQ supports some balance between short-lived network flows, such as

Web searches, and long-lived ones, such as video streaming or file transfers.

CBQ aggregates the connections and constructs a hierarchy of classes with different priorities and

throughput allocations. To accomplish link sharing, CBQ uses several functional units: (i) a classifier

that uses the information in the packet header to assign arriving packets to classes; (ii) an estimator of the short-term bandwidth for the class; (iii) a selector, or scheduler, which identifies the highest-priority

class to send next and, if multiple classes have the same priority, to schedule them on a round-robin

basis; and (iv) a delayer to compute the next time when a class that has exceeded its link allocation is

allowed to send.

The classes are organized in a tree-like hierarchy; for example, in Figure 7.8 we see two types of

traffic: group A corresponding to short-lived traffic and group B corresponding to long-lived traffic.

The leaves of the tree are considered Level 1 and in this example include six classes of traffic: real

time, Web, interactive, video streaming, audio streaming, and file transfer. At Level 2 there are the two

classes of traffic, A and B. The root, at Level 3, is the link itself.

The link-sharing policy aims to ensure that if sufficient demand exists, then, after some time intervals,

each interior or leaf class receives its allocated bandwidth. The distribution of the “excess” bandwidth

follows a set of guidelines but does not support mechanisms for congestion avoidance.

A class is overlimit if over a certain recent period it has used more than its bandwidth allocation

(in bytes per second), underlimit if it has used less, and atlimit if it has used exactly its allocation. A

leaf class is satisfied if it is underlimit and has a persistent backlog, and it is unsatisfied otherwise. A

nonleaf class is unsatisfied if it is underlimit and has some descendent class with a persistent backlog.

A precise definition of the term persistent backlog is part of a local policy. A class does not need to be

regulated if it is underlimit or if there are no unsatisfied classes. The class should be regulated if it is

overlimit and if some other class is unsatisfied, and this regulation should continue until the class is no

longer overlimit or until there are no unsatisfied classes (see Figure 7.9 for two examples).

The Linux kernel implements a link-sharing algorithm called hierarchical token buckets (HTB)

inspired by CBQ. In CBQ every class has an assured rate (AR); in addition to the AR every class in

HTB has also a ceil rate (CR) (see Figure 7.10). The main advantage of HTB over CBQ is that it allows

borrowing. If a class C needs a rate above its AR, it tries to borrow from its parent; then the parent examines its children and, if there are classes running at a rate lower than their AR, the parent can

borrow from them and reallocate it to class C.