Classification And Policing. When a packet arrives, a router *classifies* the packet by assigning the packet a flow identifier. For a fine-grain system, the identifier specifies an individual connection; for a coarse-grain system, the identifier specifies a traffic class. Once an identifier has been assigned, the router performs *policing*, which means that the router verifies that the packet does not violate parameters for the flow. In particular, if a customer sends data faster than the maximum rate for which the customer is paying, a policer begins to discard packets. One technique used for policing is *Random Early Discard* (*RED*) in which packets on a given flow are dropped probabilistically. A queue is established for the flow, and the current size of the queue is used to determine the probability of drop. When the queue is less than half full, the probability is set to zero. When the queue is completely full, the probability is set to one. At queue sizes in between, the probability is linearly proportional to the number of packets in the queue. Using RED helps avoid a cyclic problem caused by *tail drop* in which all incoming packets are discarded once a queue fills, many TCP sessions each back off and begin slow start, traffic increases until the queue fills again, and the cycle repeats.

Forwarding Computation. When computing a next-hop, a router or switch can use the flow identifier. In some cases, the flow identifier determines the path to be followed (e.g., all voice traffic is sent out port 54 to a voice switch). In other cases, the flow identifier is ignored, and the destination address in each packet is used to select a next hop. The exact details of forwarding depend on the purpose of a particular switch or router and the manager's QoS policies.

Output Queuing. Most implementations of QoS create a set of queues for each output port. Once the forwarding computation selects an output port for the packet, the output queuing mechanism uses the flow identifier to place the packet in one of the queues associated with the port. A coarse-grain system typically uses one queue per class. Thus, if a manager establishes eight QoS classes, each output port will have eight queues. A fine-grain system often has one queue per connection, with queues arranged in a hierarchy. For example, one network processor chip provides 256,000 queues arranged in a multi-level hierarchy.

Traffic Scheduling. A *traffic scheduler* implements the QoS policies by selecting a packet to send whenever a port is idle. For example, a manager might specify that three customers each receive 25% of the capacity and all other customers share the remaining capacity. To implement such a policy, a traffic scheduler might use four queues and a *round-robin* approach to select packets. Thus, if all customers are sending data, the three designated customers will each receive one quarter of the capacity, as specified.

More sophisticated packet selection algorithms can be used to implement complex forms of proportional sharing. Complexity arises because a traffic scheduler must maintain long-term policies even though packets arrive in bursts. Thus, a traffic scheduler must adapt to situations where a given queue temporarily exceeds its allotted data rate provided the long-term average meets the specified bounds. Similarly, a traffic scheduler must adapt to a situation where one or more queues is temporarily empty by dividing the unused capacity among other queues.

Many traffic scheduling algorithms have been proposed and analyzed. It is not possible to create a practical algorithm that achieves perfection; each is a compromise