BACHELOR OF COMPUTER APPLICATIONS

SEMESTER 4

DCA2201

COMPUTER NETWORKING

Unit 10

Transport Layer – Resource Allocation and Quality of Service

Table of Contents

1. INTRODUCTION

In the previous unit, we discussed congestion control in the transport layer. The presence of many packets in the network causes packet delay and loss that degrades performance, this situation is known as congestion. In this unit, we will discuss transport layer resource allocation and quality of service. Congestion control and resource allocation are two sides of the same coin. We can see that congestion-control mechanisms have some sort of resource allocation built into them. If the network takes an active role in allocating resources properly, then congestion may be avoided.

In this unit, we will start with a discussion on issues in resource allocation. In this unit, we will then discuss the different queuing disciplines and will discuss the quality of service for the transport of data.

1.1 Objectives:

After studying this unit, you should be able to:

- ❖ *Describe issues in resource allocation*
- ❖ *Explain queuing*
- ❖ *Describe traffic shaping*
- ❖ *Explain packet scheduling*
- ❖ *Describe admission control*

2. ISSUES IN RESOURCE ALLOCATION

Resource allocation is partially implemented in the routers, switches, and links inside the network and partially in the transport protocol running on the end hosts. End systems may use signaling protocols to convey their resource requirements to network nodes, which respond with information about resource availability. Resource allocation is the process by which network elements try to meet the competing demands that applications have for network resources such as primarily link bandwidth and buffer space in routers or switches.

2.1 Network Model

In this section, we will see the resource allocation in different network models.

Packet-switched network

Packet switching is a digital networking communications method that groups all transmitted data into suitably sized blocks, called **packets** that are transmitted through a communication network. First, we consider the resource allocation in a packet-switched network consisting of multiple links and routers. In such an environment, a given source may have more than enough capacity on the immediate outgoing link to send a packet, but somewhere in the middle of a network its packets encounter a link that is being used by many different traffic sources.

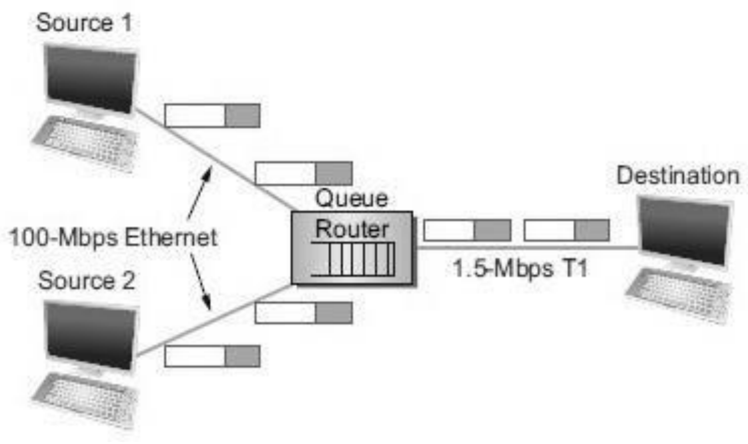


Fig 10.1: A packet switched network

Figure 10.1 illustrates a packet switched network where two high-speed links are feeding a low-speed link. This is in contrast to shared-access networks like Ethernet and wireless networks, where the source can directly observe the traffic on the network and decide accordingly whether or not to send a packet.

Connectionless flows

We assume that the network is essentially connectionless, with any connection-oriented service implemented in the transport protocol that is running on the end hosts. This is one of the model of the internet where IP provides a connectionless datagram delivery service and TCP implements an end-to-end connection abstraction. All datagrams are completely independent in a connectionless network because datagrams are certainly switched independently. But it is usually the case that a stream of datagrams between a particular pair of hosts flows through a particular set of routers. This idea of a *flow,* that is, a sequence of packets sent between a source/destination pair and following the same route through the network is an important abstraction in the context of resource allocation.

One of the powers of flow abstraction is that flows can be defined at different granularities. It can be host-to-host (i.e., have the same source/destination host addresses) or process-to-process (i.e., have the same source/destination host/port pairs). Figure 10.2 illustrates several flows passing through a series of routers.

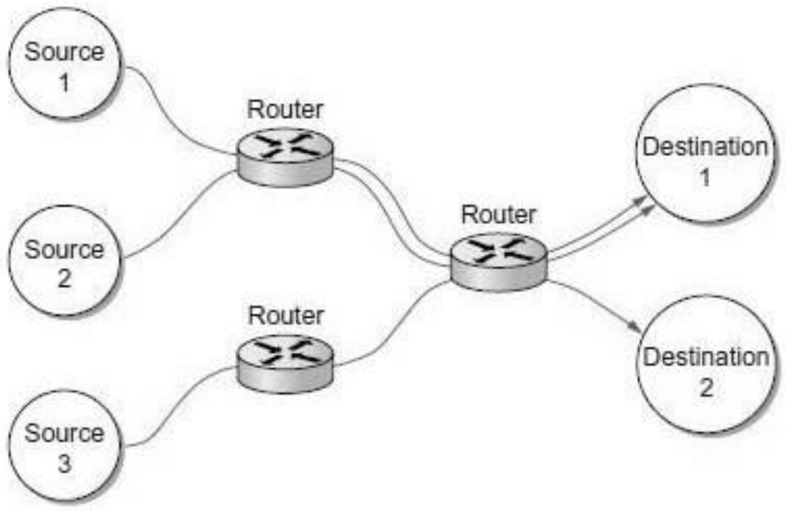


Fig 10.2: Multiple flows passing through a set of routers

Because multiple related packets flow through each router, it sometimes makes sense to maintain some state information for each flow. Information that can be used to make resource allocation decisions about the packets that belong to the flow. This state is sometimes called soft state; the main difference between *soft* state and hard state is that soft state need not always be explicitly created and removed by signaling. Soft state represents a middle ground between a purely connectionless network that maintains *no* state at the routers and a purely connection-oriented network that maintains hard state at the routers. In general, the correct operation of the network does not depend on soft state being present, but when a packet happens to belong to a flow for which the router is currently maintaining soft state, then the router is better able to handle the packet.

Flow can be either implicitly defined or explicitly established. In the former case, each router watches for packets that happen to be traveling between the same source/destination pair. The router does this by inspecting the addresses in the header and treats these packets as belonging to the same flow for the purpose of congestion control. In the latter case, the source sends a flow setup message across the network, declaring that a flow of packets is about to start. Even when explicitly established, a flow does not imply any end-to-end semantics and, in particular, does not imply the reliable and ordered delivery of a virtual circuit. It simply exists for the purpose of resource allocation.

2.2 Taxonomy

There are different ways in which resource allocation mechanisms differ, so creating a thorough taxonomy for resource allocation is a difficult proposition. Here, we will discuss three dimensions along which resource allocation mechanisms can be characterized.

Router-centric versus host-centric

Resource allocation mechanisms can be classified into two broad groups. They are those that address the problem from inside the network (i.e., at the routers) and those that address it from the edges of the network (i.e., in the hosts). In a router-centric design, each router takes responsibility for deciding when packets are forwarded and selecting which packets are to be dropped, as well as for informing the hosts that are generating the network traffic how many packets they are allowed to send. In a host-centric design, the end hosts observe the network conditions and adjust their behavior accordingly.

Reservation-based versus feedback-based

A second way that resource allocation mechanisms are sometimes classified is according to whether they use reservations or *feedback.* In a reservation-based system, some entity (e.g., the end host) asks the network for a certain amount of capacity to be allocated for a flow. Each router then allocates enough resources to satisfy this request. If the request cannot be satisfied at some router, because doing so would overcommit its resources, then the router rejects the reservation. This is analogous to getting a busy signal when trying to make a phone call. In a feedback-based approach, the end hosts begin sending data without first reserving any capacity and then adjust their sending rate according to the feedback they receive. This feedback can be either explicit (i.e., a congested router sends a "please slow down" message to the host) or implicit (i.e., the end host adjusts its sending rate according to the externally observable behavior of the network, such as packet losses).

Window based versus rate based

A third way to characterize resource allocation mechanisms is according to whether they are window based or rate based. We have already seen window-based transport protocols, such as TCP, in which the receiver advertises a window to the sender. This window corresponds to how much buffer space the receiver has, and it limits how much data the sender can transmit; that is, it supports flow control. A similar mechanism known as window *advertisement* can be used within the network to reserve buffer space.

It is also possible to control a sender's behavior using a rate, that is, how many bits per second the receiver or network is able to absorb. Rate-based control makes sense for many multimedia applications, which tend to generate data at some average rate and which need at least some minimum throughput to be useful.

2.3 Evaluation Criteria

The final issue is one of knowing whether a resource allocation mechanism is good or not. Different types of evaluation schemes can be used, which are given below.

Effective Resource Allocation

A good starting point for evaluating the effectiveness of a resource allocation scheme is to consider the two principal metrics of networking: throughput and delay. Clearly, we want as much throughput and as little delay as possible.

One sure way for a resource allocation algorithm to increase throughput is to allow as many packets into the network as possible, so as to drive the utilization of all the links up to 100%. We would do this to avoid the possibility of a link becoming idle because an idle link necessarily hurts throughput. The problem with this strategy is that increasing the number of packets in the network also increases the length of the queues at each router. Longer queues, in turn, mean packets are delayed longer in the network. The ratio of throughput to delay can be used as a metric for evaluating the effectiveness of a resource allocation scheme. This ratio is sometimes referred to as the *power* of the network. The power is given by:

Power = Throughput/Delay

The objective is to maximize this ratio, which is a function of how much load to be placed on the network. The load, in turn, is set by the resource allocation mechanism. Figure 10.3 shows a representative power curve, in which, ideally, the resource allocation mechanism would operate at the peak of this curve.

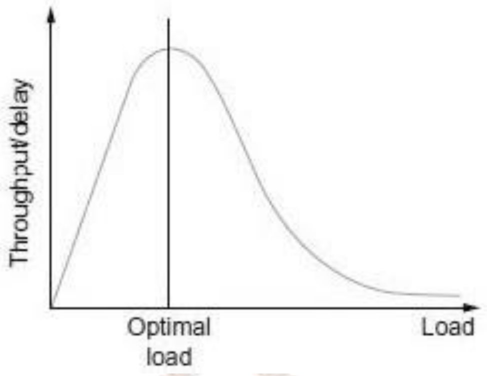


Fig 10.3: Ratio of throughput to delay as a function of load.

To the left of the peak, the mechanism is being too conservative. That is, it is not allowing enough packets to be sent to keep the links busy. To the right of the peak, so many packets are being allowed into the network that increases in delay due to queuing are starting to dominate any small gains in throughput.

Fair resource allocation

We must also consider the issue of fairness for judging a resource allocation scheme. We can't define what exactly constitutes fair resource allocation. Consider for example, a reservation-based resource allocation scheme provides an explicit way to create controlled unfairness. With such a scheme, we might use reservations to enable a video stream to receive 1 Mbps across some link while a file transfer receives only 10 kbps over the same link.

In the absence of explicit information to the contrary, when several flows share a particular link, we would like for each flow to receive an equal share of the bandwidth. This definition assumes that a *fair* share of bandwidth means an equal share of bandwidth. But, even in the absence of reservations, e*qual* shares may not equate to fair shares if we consider length of the paths being compared into account.

SELF-ASSESSMENT QUESTIONS – 1

1. End systems may use _____________ to convey their resource requirements to network nodes.
2. Because multiple related packets flow through each router, it sometimes makes sense to maintain some state information for each flow, this state is sometimes called _______________.
3. Flow can be either implicitly defined or explicitly established. (True/False)
4. In a ______________ design, the end hosts observe the network conditions and adjust their behavior accordingly.
5. Which of the following system, some entity asks the network for a certain amount of capacity to be allocated for a flow?
   a) Feed-back based
   b) reservation-based
   c) rate-based
   d) window-based
6. is an example for window-based transport protocols.
7. In order to evaluate the effectiveness of a resource allocation scheme, we have to consider the two principal metrics of networking. They are ___________and ________________.

3. QUEUING

Queuing is a technique used in internetwork devices such as routers or switches during periods of congestion. Packets are held in the queues for subsequent processing. After being processed by the router, the packets are then sent to their destination based on priority. Irrespective of how simple or how advanced the rest of the resource allocation mechanism is, each router must implement some queuing discipline that governs how packets are buffered while waiting to be transmitted. The queuing algorithm can be thought of as allocating both bandwidth and buffer space. In this session, we will discuss two common queuing algorithms. They are, first-in, first-out (FIFO) and fair queuing (FQ).

3.1 FIFO

The idea of FIFO queuing, also called first-come, first-served (FCFS) queuing, is that *the first packet that arrives at a router is the first packet to be transmitted*. Figure 10.4 (a) shows a FIFO with "slots" to hold up to eight packets. Given that the amount of buffer space at each router is finite, if a packet arrives and the queue (buffer space) is full, then the router discards that packet, as shown in figure 10.4 (b).

This is done without regard to which flow the packet belongs to or how important the packet is. This is sometimes called tail drop, since packets that arrive at the tail end of the FIFO are dropped. Note that tail drop and FIFO are two separable ideas. FIFO is a *scheduling discipline*. It determines the order in which packets are transmitted. Tail drop is a *drop policy*, which determines which packets get dropped. Because FIFO and tail drop are the simplest instances of scheduling discipline and drop policy, respectively, they are sometimes viewed as a bundle known as the vanilla queuing implementation which is often referred to simply as *FIFO queuing.*

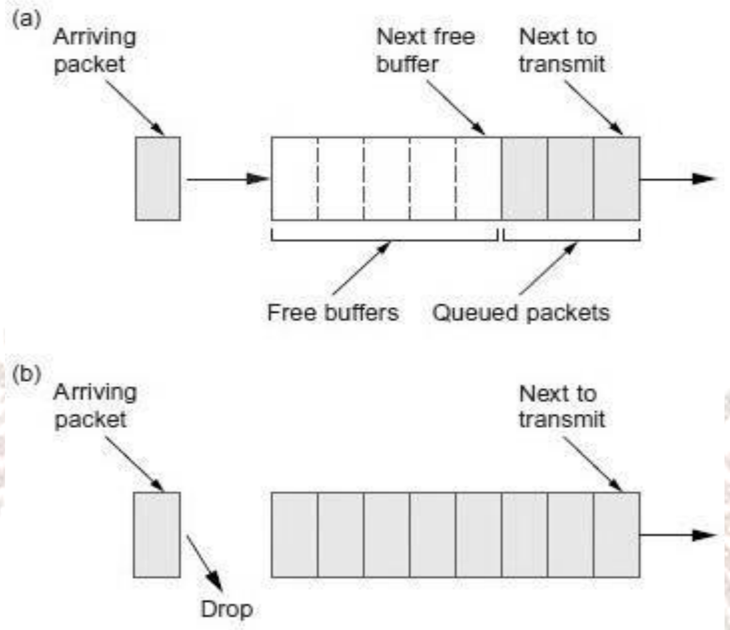


Fig 10.4: (a) FIFO queuing (b) tail drop at a FIFO queue

This simple approach to queuing pushes all responsibility for congestion control and resource allocation out to the edges of the network. A simple variation on basic FIFO queuing is priority queuing. The idea is to mark each packet with a priority. The routers then implement multiple FIFO queues, one for each priority class. The router always transmits packets out of the highest-priority queue if that queue is nonempty before moving on to the next priority queue. Within each priority, packets are still managed in a FIFO manner.

3.2 Fair Queuing

The main problem with FIFO queuing is that it does not discriminate between different traffic sources or it does not separate packets according to the flow to which they belong. This is a problem at two different levels. At one level, it is not clear that any congestion-control algorithm implemented entirely at the source will be able to adequately control congestion with so little help from the routers. At another level, because the entire congestion-control mechanism is implemented at the sources and FIFO queuing does not provide a means to inspect how well the sources adhere to this mechanism. Fair queuing (FQ) is an algorithm that has been proposed to address this problem.

The idea of FQ is to maintain a separate queue for each flow currently being handled by the router. The router then services these queues in a sort of round-robin manner as shown in figure 10.5.

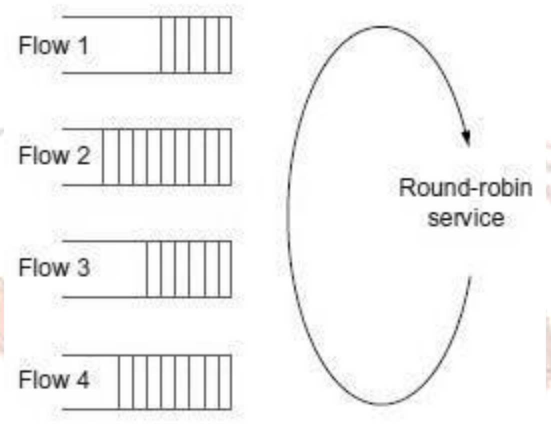


Fig 10.5: Round-robin service of four flows at a router

When a flow sends packets too quickly, then its queue fills up. When a queue reaches a particular length, additional packets belonging to that flow's queue are discarded. In this way, a given source cannot arbitrarily increase its share of the network's capacity at the expense of other flows.

FQ does not involve the router telling the traffic sources anything about the state of the router or in any way limiting how quickly a given source sends packets. In other words, FQ is still designed to be used in conjunction with an end-to-end congestion-control mechanism. It simply segregates traffic so that ill-behaved traffic sources do not interfere with those that are faithfully implementing the end-to-end algorithm. FQ also enforces fairness among a collection of flows managed by a well-behaved congestion-control algorithm.

There are two things to notice about fair queuing. First, the link is never left idle as long as there is at least one packet in the queue. Any queuing scheme with this characteristic is said to be work conserving. One effect of being work *conserving* is that if we are sharing a link with a lot of flows that are not sending any data then; we can use the full link capacity for our flow. As soon as the other flows start sending, however, they will start to use their share and the capacity available to our flow will drop. The second thing to notice is that if the link is fully loaded and there are n-flows sending data, we cannot use more than 1/nth of the link bandwidth. If we try to send more than that, our packets will be assigned increasingly large timestamps, causing them to sit in the queue longer awaiting transmission.

It is possible to implement a variation of FQ, called weighted fair queuing (WFQ) that allows a weight to be assigned to each flow (queue). This weight logically specifies how many bits to transmit each time the router services that queue, which effectively controls the percentage of the link's bandwidth that that flow will get. Simple FQ gives each queue a weight of 1, which means that logically only 1 bit is transmitted from each queue each time around. This results in each flow getting 1/nth of the bandwidth when there are n flows. With WFQ, if, one queue might have a weight of 2, a second queue might have a weight of 1, and a third queue might have a weight of 3. Assuming that each queue always contains a packet waiting to be transmitted, the first flow will get one-third of the available bandwidth, the second will get one-sixth of the available bandwidth, and the third will get one-half of the available bandwidth.

SELF-ASSESSMENT QUESTIONS – 2

8. The idea of ______________ is that the first packet that arrives at a router is the first packet to be transmitted.
9. In_ _ _ _ _ _ _ _ _ _ _ queuing, the idea is to mark each packet with a priority.
10. In __________________queuing, the idea is to maintain a separate queue for each flow currently being handled by the router.

4. QUALITY OF SERVICE

Quality of service (QoS) refers to a network's ability to achieve maximum bandwidth and deal with other network performance elements like latency, error rate and uptime. Quality of service also involves controlling and managing network resources by setting priorities for specific types of data on the network. A network that can provide different levels of services such as real time applications and non-real time applications are often said to support quality of service (QoS). An easy solution to provide good quality of service is to build a network with enough capacity for whatever traffic will be thrown at it. This is known as *overprovisioning.* The trouble with this solution is that it is expensive.

4.1 Traffic Shaping

Before the network can make QoS guarantees, it must know what traffic is being guaranteed. *Traffic shaping* is a technique for regulating the average rate and burstiness of a flow of data that enters the network. When a flow is set up, the user and the network agree on a certain traffic pattern for that flow. This agreement is called an SLA (Service Level Agreement). Traffic shaping reduces congestion. Monitoring traffic flow is called *traffic policing*. Shaping and policing are important while transmitting real time data such as audio or video which have rigorous quality-of-service requirements. One more general way to characterize traffic is with the *leaky bucket* and *token bucket* algorithms. Consider a bucket with a small hole in the bottom as shown in figure 10.6 (b). Whatever the rate at which water enters the bucket, the outflow is at a constant rate, *R*, when there is any water in the bucket and zero when the bucket is empty. Also, once the bucket is full to capacity *B*, any additional water entering it spills over the sides and is lost.

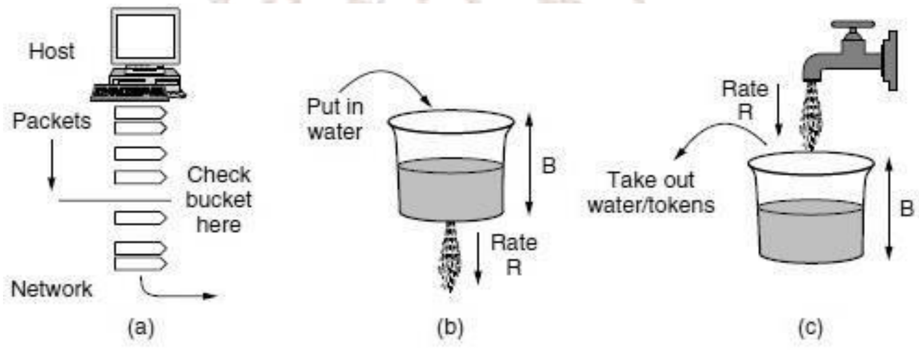


Fig 10.6: (a) Shaping packets (b) A Leaky bucket (c) A token bucket.

This bucket can be used to shape or police packets entering the network, as shown in figure 10.6 (a). Conceptually, each host is connected to the network by an interface containing a leaky bucket. To send a packet into the network, it must be possible to put more water into the bucket. If a packet arrives when the bucket is full, the packet must either be queued until enough water leaks out to hold it or be discarded. This technique was proposed by Turner in 1986 and is called the *leaky bucket algorithm.*

A different but equivalent formulation is to imagine the network interface as a bucket that is being filled as shown in figure 10.6 (c). Here, the tap is running at rate R and the bucket has a capacity of B, as before. Now, to send a packet we must be able to take water, or tokens, as the contents are commonly called, out of the bucket. No more than a fixed number of tokens, B, can accumulate in the bucket, and if the bucket is empty, we must wait until more tokens arrive before we can send another packet. This algorithm is called the *token bucket algorithm.*

4.2 Packet Scheduling

Algorithms that allocate router resources among the packets of a flow and between competing flows are called packet *scheduling algorithms.* Three different kinds of resources can potentially be reserved for different flows. They are bandwidth, bufferspace and CPU cycles. Packet scheduling algorithms allocate bandwidth and other router resources by determining which of the buffered packets to send on the output line next. Each router buffers packets in a queue for each output line until they can be sent, and they are sent in the same order that they arrived. This algorithm is known as *FIFO (First-In First-Out), or equivalently FCFS (First-Come First-Serve).*

FIFO scheduling is simple to implement, but it is not suited to providing good quality of service because when there are multiple flows, one flow can easily affect the performance of the other flows. Many packet scheduling algorithms have been invented that provide stronger isolation between flows and prevent attempts at interference. One of the first ones was the *fair queueing* algorithm developed by Nagle in 1987. The essence of this algorithm is that routers have separate queues, one for each flow for a given output line. When the line becomes idle, the router scans the queues in a round-robin manner, as shown in figure 10.7.

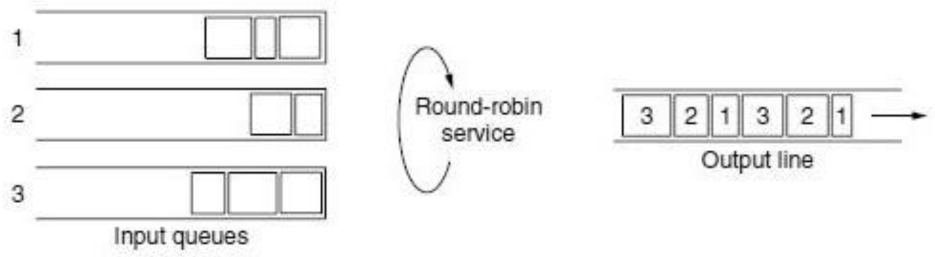
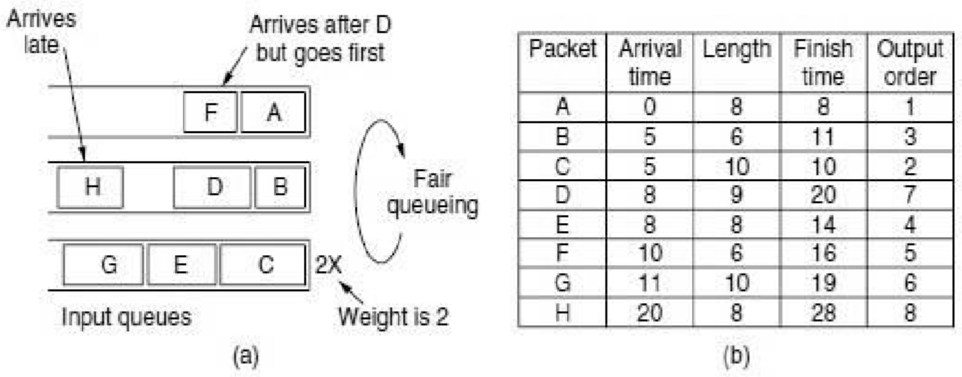


Fig 10.7: Round-robin fair queueing

It then takes the first packet to the next queue. In this way, with n hosts competing for the output line, each host gets to send one out of every n packet. It is fair in the sense that all flows get to send packets at the same rate. Sending more packets will not improve this rate.

Demers et al. in 1990 suggested an improvement that has been done in round-robin in such a way as to implement a byte-byte round-robin instead of a packet-by-packet round-robin. The idea is to compute a virtual time, that is the number of the round at which each packet would finish being sent. Each round drains a byte from all of the queues that have data to send. The packets are then sorted in order of their finishing times and sent in that order. This algorithm and an example of finish times for packets arriving in three flows are illustrated in figure 10.8.



Packet	Arrival time	Length	Finish time	Output order
A	0	8	8	1
B	5	6	11	3
C	5	10	10	2
D	8	9	20	7
E	8	8	14	4
F	10	6	16	5
G	11	10	19	6
H	20	8	28	8

Fig 10.8: (a) Weighted fair queueing. (b) Finishing times for the packets

If a packet has length *L*, the round at which it will finish is simply *L* rounds after the start time. The start time is either the finish time of the previous packet, or the arrival time of the packet, if the queue is empty when it arrives. In figure 10.8 (b), when looking only at the first two packets in the top two queues, packets arrive in the order *A, B, D,* and *F*. Packet *A* arrives at round 0 and is 8 bytes long, so its finish time is round 8. Similarly, the finish time for packet

B is 11. Packet D arrives while B is being sent. Its finish time is 9 byte-rounds after it starts when B finishes, or 20. Similarly, the finish time for F is 16.

One shortcoming of this algorithm in practice is that it gives all hosts the same priority. Another modified algorithm is called *WFQ (Weighted Fair Queueing).* Letting the number of bytes per round be the weight of a flow, W. As a final example of a scheduler, packets might carry timestamps and be sent in timestamp order. Sending packets in order of their timestamps has the beneficial effect of speeding up slow packets while at the same time slowing down fast packets. The result is that all packets are delivered by the network with a more consistent delay.

4.3 Admission Control

Quality of Service (QoS) guarantees are established through the process of admission control. Admission control is used to control congestion, which is a performance guarantee. The user offers a flow of packets with an accompanying QoS requirement to the network. The network then decides whether to accept or reject the flow based on its capacity and the commitments it has made to other flows. If it accepts, the network reserves capacity in advance at routers to guarantee QoS when traffic is sent on the new flow.

Many routing algorithms find the single best path between each source and each destination and send all traffic over the best path. This may cause some flows to be rejected if there is not enough spare capacity along the best path. QoS guarantees for new flows may still be kept by choosing a different route for the flow that has excess capacity. This is called QoS routing.

The idea behind admission control is simple: When some new flow of packets wants to receive a particular level of service, admission control looks at the flow's traffic characteristics (called the *TSpec*) and the service requested from the network (called the *RSpec*) of the flow and tries to decide if the desired service can be provided to that amount of traffic, given the currently available resources, without causing any previously admitted flow to receive worse service than it had requested. If it can provide the service, the flow is admitted; if not, then it is denied. The hard part is figuring out when to say yes and when to say no. Admission control is very dependent on the type of requested service and on the queuing, discipline employed in the routers.

Admission control should not be confused with policing. Admission control is a per-flow decision to admit a new flow or not. Policing is a function applied on a per-packet basis to make sure that a flow conforms to the TSpec that was used to make the reservation. If a flow does not conform to its TSpec for example, because it is sending twice as many bytes per second as it said it would then it is likely to interfere with the service provided to other flows, and some corrective action must be taken. There are several options, the evident one being to drop offending packets. However, another option would be to check if the packets really are interfering with the service of other flows. If they are not interfering, the packets could be sent on after being marked with a tag that says, in effect, "This is a nonconforming packet. Drop it first if you need to drop any packets."

Admission control is closely related to the important issue of policy. For example, a network administrator might wish to allow reservations made by his company's CEO to be admitted while rejecting reservations made by more lowly employees. Of course, the CEO's reservation request might still fail if the requested resources aren't available, so we see that issues of policy and resource availability may both be addressed when admission control decisions are made.

SELF-ASSESSMENT QUESTIONS – 3

11. An easy solution to provide good quality of service is to build a network with enough capacity for whatever traffic will be thrown at it. This is known as ____________________.
12. is a technique for regulating the average rate and burstiness of a flow of data that enters the network.
13. Monitoring traffic flow is called ____________________.
14. Algorithms that allocate router resources among the packets of a flow and between competing flows are called _______________.

5. SUMMARY

Let us recapitulate the important concepts discussed in this unit:

- Resource allocation is partially implemented in the routers, switches, and links inside the network and partially in the transport protocol running on the end hosts.
- In a router-centric design, each router takes responsibility for deciding when packets are forwarded and selecting which packets are to be dropped.
- In a host-centric design, the end hosts observe the network conditions and adjust their behavior accordingly.
- In a reservation-based system, some entity asks the network for a certain amount of capacity to be allocated for a flow.
- In a feedback-based approach, the end hosts begin sending data without first reserving any capacity and then adjust their sending rate according to the feedback they receive.
- The idea of FIFO queuing, also called first-come, first-served (FCFS) queuing, is that the first packet that arrives at a router is the first packet to be transmitted.
- The idea of fair queuing (FQ) is to maintain a separate queue for each flow currently being handled by the router.
- Traffic shaping is a technique for regulating the average rate and burstiness of a flow of data that enters the network.

6. TERMINAL QUESTIONS

1. Describe resource allocation in different network models.
2. Describe different taxonomy in resource allocation.
3. Differentiate between FIFO queuing and fair queuing.
4. Explain Traffic Shaping.
5. Describe Packet Scheduling.

7. ANSWERS

Self-Assessment Questions

1. Signaling Protocols
2. Soft state
3. (a) True
4. Host-centric
5. (b) reservation- based
6. TCP
7. Throughput, delay
8. FIFO queuing
9. Priority
10. Fair queuing
11. Overprovisioning
12. Traffic shaping
13. Traffic policing
14. Packet scheduling algorithm

Terminal Questions

1. First, we consider the resource allocation in a packet-switched network (or internet) consisting of multiple links and routers. In such an environment, a given source may have more than enough capacity on the immediate outgoing link to send a packet, but somewhere in the middle of a network its packets encounter a link that is being used by many different traffic sources. (Refer section 2.1 for more details).
2. There are different ways in which resource allocation mechanisms differ, so creating a thorough taxonomy is a difficult proposition. Here, we will discuss three dimensions

along which resource allocation mechanisms can be characterized. Resource allocation mechanisms can be classified into two broad groups. They are those that address the problem from inside the network (i.e., at the routers) and those that address it from the edges of the network (Refer section 2.2 for more details).

3. The idea of FIFO queuing, also called first-come, first-served (FCFS) queuing, is that *the first packet that arrives at a router is the first packet to be transmitted*. The idea of Fair Queuing is to maintain a separate queue for each flow currently being handled by the router. The router then services these queues in a sort of round-robin manner. (Refer section 3.1 and 3.2 for more details).
4. *Traffic shaping* is a technique for regulating the average rate and burstiness of a flow of data that enters the network. When a flow is set up, the user and the network agree on a certain traffic pattern for that flow. This agreement is called an SLA (Service Level Agreement). Traffic shaping reduces congestion. (Refer section 4.1 for more details).
5. Algorithms that allocate router resources among the packets of a flow and between competing flows are called *packet scheduling algorithms.* Three different kinds of resources can potentially be reserved for different flows. They are bandwidth, buffer space and CPU cycles. (Refer section 4.2 for more details).

References:

1. Andrew S Tanenbaum, David J.Wetherall, *"Computer Networks,"* Fifth edition.
2. Larry L. Peterson, Bruce S. Davie, "Computer Networks- a Systems Approach," Fifth edition.
3. James F. Kurose, Keith W.Ross, "Computer *Networking-A top-down approach*," Sixth edition.
4. Behrouz A. Forouzan, Sophia Chung Fegan, *"Data Communication and Networking,"* Fourth edition.
5. William Stallings, *"Computer Networking With Internet Protocols and Technology,"* Third edition.