Chapter 6 Congestion Control and Resource Allocation

Question: How to *effectively* and *fairly* allocate resources among a collection of competing users?

Introduction to Congestion Control and Resource Allocation

- Resources
  - Bandwidth of the links
  - Buffers on the routers and switches
- Packets contend at a router for the use of a link, with each contending packet placed in a queue waiting for its turn to be transmitted over the link
- When too many packets are contending for the same link
  - The queue overflows and packets get dropped
  - When drops become common events, the network is said to be *congested.* Network should provide a congestion control mechanism to deal with such a situation
- Congestion control and resource allocation are two sides of the same coin
  - If the network takes an active role in allocating resources, then congestion may be avoided → No need for congestion control
    - But allocating resources with any precision is difficult because resources are distributed throughout the network
  - On the other hand, we can always let the sources send as much data as they want and then recover from the congestion when it occurs
    - Easier approach but can be disruptive because many packets may be discarded by the network before congestion can be controlled

- Congestion control and resource allocation involve both hosts and network elements such as routers
    - In network elements various queuing disciplines can be used to control the order in which packets get transmitted and which packets get dropped
    - At the end hosts, the congestion control mechanism paces how fast sources are allowed to send packets

Issues in Resource Allocation

- Network Model
    - We consider resource allocation in a packet-switched network (or internet) consisting of multiple links and switches (or routers).
        - 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
    - 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.
        - The datagrams are 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
        - A sequence of packets sent between a source/destination pair and following the same route through the network is called a *flow*
        - Flows can be defined at different granularities. For example, a flow can be host-to-host or process-to-process.

- Taxonomy
  - Router-Centric versus Host-Centric
    - 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 (e.g., how many packets they are successfully getting through the network) and adjust their behavior accordingly.
    - Note that these two groups are not mutually exclusive
  - Reservation-Based versus Feedback-Based
    - 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 (buffers and/or percentage of the link's bandwidth) to satisfy this request. If the request cannot be satisfied at some router, then the router rejects the reservation.
    - 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 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
  - Window-based mechanism uses a window to reserve buffer space (i.e., to support resource allocation) within the network
  - Rate-based mechanism controls sender's behavior using a rate, that is, how many bits per second the network is able to absorb

- Evaluation Criteria
  - Effective resource allocation
    - A good starting point is to consider the two principal metrics of networking: throughput and delay.
    - We want as much throughput and as little delay as possible. Unfortunately, these two goals are often somewhat at odds with each other.
    - Some network designers have proposed using the ratio of throughput to delay as a metric for evaluating the effectiveness of a resource allocation scheme. This ratio is referred to as the *power* of the network.
      - Power = Throughput/Delay
    - Power is a function of the network load. The load, in turn, is set by the resource allocation mechanism
  - Fair resource allocation
    - Assuming that fair implies equal share of bandwidth and that all paths are of equal length, Raj Jain proposed a metric that can be used to quantify the fairness of a congestion-control mechanism.
    - Given a set of flow throughputs (x_1, x_2, . . . , x_n), Jain's fairness index is defined as

$$f(x_1, x_2, \ldots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}$$

        - The fairness index always results in a number between 0 and 1, with 1 representing greatest fairness.

Queuing Disciplines

- Each router must implement some queuing discipline that consists of a *scheduling discipline* and a *drop policy*
    - The scheduling discipline determines the order in which packets are transmitted (i.e., allocates link bandwidth).
    - The drop policy determines which packets get discarded (i.e., allocates buffer space).
- FIFO (First-Come, First-Served) Queuing
    - The first packet that arrives at a router is the first packet to be transmitted
    - If a packet arrives and the queue (buffer space) is full, then the router discards that packet. This is called *tail drop*.
    - FIFO with tail drop is the most widely used in Internet routers
        - The prevalent form of congestion control in the Internet assumes no help from the routers
- Priority Queuing - a simple variation on FIFO queuing
    - Each packet is marked with a priority.
    - The routers implement multiple FIFO queues, one for each priority class.
    - Queues are serviced in strict order of queue priority. Within each priority, packets are managed in a FIFO manner.
    - Problem: the high-priority queue can starve out all the other queues
        - There need to be limits on how much high-priority traffic is inserted in the queue

    - Priority queuing is used in the Internet: there is a special queue for routing updates
- Fair Queuing (FQ)
    - The main problem with FIFO queuing is that it does not separate packets according to the flow to which they belong.
        - It is possible for an ill-behaved source (flow) to capture an arbitrarily large fraction of the network capacity
    - FQ addresses this problem by maintaining a separate queue for each flow currently being handled by the router. Queues are serviced in round-robin.
        - FQ segregates traffic so that an ill-behaved source does not interfere with well-behaved sources.
    - Packets being processed at a router are not necessarily the same length ➔ To truly allocate the link bandwidth in a fair manner, it is necessary to take packet length into consideration
    - What we really want is bit-by-bit round-robin; FQ simulates this behavior by first determining when a given packet would finish being transmitted if it were being sent using bit-by-bit round-robin, and then using this finishing time to sequence the packets for transmission.
    - The algorithm:
        - Imagine a clock that ticks once each time a bit is transmitted from all of the active flows.
        - For a single flow
            - Let P_i denote the length of packet i.
            - Let S_i denote the time when the router starts to transmit packet i.

- Let F_i denote the time when the router finishes transmitting packet i.
- Let A_i denote the time when packet i arrives at the router.
- $F_i = S_i + P_i$ and $S_i = MAX (F_{i-1} , A_i)$, so $F_i = MAX (F_{i-1} , A_i) + P_i$.

  - For multiple flows
    - For each flow, calculate F_i for each packet.
    - Treat all F_i's as timestamps, next packet to transmit is one with lowest timestamp.
  - The algorithm does not exactly simulate bit-by-bit round robin because a newly arriving packet can't preempt a packet that is currently being transmitted.
- Properties of FQ
  - Work-conserving: the link is never left idle as long as there is at least one packet in the queue.
  - Fair bandwidth allocation: when there are n active flows, each flow gets $1/n^{th}$ of the link bandwidth.

- Weighted Fair Queuing (WFQ) - a variation of FQ
  - A weight is assigned to each flow (queue)
    - The weight specifies how many bits to transmit each time the router services that queue
  - Bandwidth received by a flow depends on its weight and the number of flows that are sharing the link.
  - If packet i belongs to a flow with weight w, then $F_i = MAX (F_{i-1} , A_i) + P_i/w$