Fair Queuing Algorithm

Reviewed by Ankur Gupta (04010605)
As a part of MA402 course under Dr. N. Selvaraju.

Introduction:

We are going to discuss a fair queuing algorithm over here. This algorithm finds its role in controlling congestion in datagram networks and is based on an earlier suggestion be Nagel. It was found out that fair queuing provides several important advantages over the usual first-come-first-serve queuing algorithm which we are going to find out in the paper.

Congestion in the datagram networks can be controlled in two ways:

1. At the source, where flow control algorithms vary the rate at which source sends packets.
2. At the gateway, where routing and queuing algorithms control the congestion.

We are going to assume that we are using the best flow control algorithms and hence in this paper we are going to discuss only about queuing algorithms.

Queuing algorithms can be thought of as allocating three nearly independent quantities: bandwidth (which packets get transmitted), promptness (when do those packets get transmitted), and buffer space (which packets are discarded by the gateway). Until the time when this paper was published first-come-first-serve (FCFS) was the most common algorithm. But this algorithm was fraught with malfunctions such as allocation of higher bandwidth to a source, sending packets at sufficiently high speed. So Nagle proposed a fair queuing algorithm in which gateways maintain separate queues for packets from each individual source and serviced them in a round-robin manner.

In circuit switched networks where there is explicit buffer reservation and uniform packet size, it has been established that round robin service disciplines allocate bandwidth fairly. But in case of other networks, a source using long packets gets more bandwidth than one using short bandwidth, so bandwidth is not allocated fairly.

What is a fair allocation?

Let's consider allocation of a single resource among N users as an example. Assume there is an amount μ_{total} of this resource and each user requests an amount ρ_i and, under a particular allocation, receives an amount μ_i. The max-min fairness criterion states that an allocation is fair if (1) no user receives more than it requests, (2) no other allocation scheme satisfying condition 1 has a higher minimum allocation, and condition 2 remains recursively true as we remove the minimal user and reduce the total resource accordingly, $\mu_{total} \leftarrow \mu_{total} - \mu_{min}$. This condition reduces to $\mu_i = MIN(\mu_{fair}, \rho_i)$ in the simple example, with μ_{fair}, the *fair share*, being set so that $\mu_{total} = \sum_{i=1}^{N} \mu_i$. Allocation on the basis of source-destination pairs, or *conversations*, will constitute a user.

Definition of Algorithm

Owing to varying packet-sizes, fair allocation of bandwidths is not an easy task. To see how this unfairness can be avoided, a hypothetical service discipline where transmission occurs in a bit-by-bit round robin fashion. In this the bandwidth is allocated fairly since at every instant in time, each conversation is receiving its fair share. Let R(t) denote the number of rounds made in the round-robin service discipline up to time t (R(t) is a continuous function, with the fractional part indicating partially completed rounds). Let $N_{ac}(t)$ denote the number of active conversations, i.e. those that have bits in their queue at time t. Then, $\frac{\partial R}{\partial t} = \frac{\mu}{N_{ac}(t)}$, where μ is the linespeed of the gateway's outgoing line (we will, for convenience, work in units such that μ= 1). A packet of size P whose first bit gets serviced at time t_o will have its last bit serviced P rounds later, at time t such that $R(t)=R(t_o)+P$. Let t_i^α be the time that packet i belonging to conversation α arrives at the gateway, and define the numbers S_i^α and F_i^α as the values of R(t) when the packet started and finished service. With P_i^α denoting the size of the packet, the following relations hold: $F_i^\alpha = S_i^\alpha + P_i^\alpha$ and $S_i^\alpha = MAX(F_{i-1}^\alpha, R(t_i^\alpha))$.

Sending packets in a bit-by-bit round robin fashion while satisfying our requirements for an adequate queuing algorithm, is obviously unrealistic. This impractical algorithm is tried to be emulated in a practical packet-by-packet transmission scheme. A natural way to emulate the bit-by-bit round-robin algorithm is to let the quantities F_i^α define the sending order of the packets. Our packet-by-packet transmission algorithm is simply defined by the rule that, whenever a packet finishes transmission, the next packet sent is the one with the smallest value of F_i^α. In a preemptive version of this algorithm, newly arriving packets whose finishing number F_i^α is smaller than that of the packet currently in transmission preempt the transmitting packet.

Promptness allocation must be based solely on data already available at the gateway. One such allocation strategy is to give more promptness (less delay) to users who utilize less than their fair share of bandwidth. Separating the promptness allocation from the bandwidth allocation can be accomplished by introducing a nonnegative parameter δ, and defining a new quantity, the bid B_i^α, via $B_i^\alpha = P_i^\alpha+MAX(F_{i-1}^\alpha, R(t_i^\alpha)-\delta)$. The quantities $R(t)$, $N_{ac}(t)$, S_i^α and F_i^α remain as before, but now the sending order is determined by the B's, not the F's. The asymptotic bandwidth allocation is independent of δ, since the F's control the bandwidth allocation, but the algorithm gives slightly faster service to packets that arrive at an inactive conversation.

Properties of Fair Queuing:

We have not been able to characterize the promptness allocation for an arbitrary arrival stream of packets. This is explained properly, with the help of an elaborate example, in the research paper attached. And I feel that since the whole section needs to be read to understand it

completely, so it's not worth to copy and paste the whole section here. Instead interested students can read the section 2.3 from the paper attached.

Bibliography:

1. Demers A, Keshav S, Shenker S. Analysis and Simulation of a Fair Queuing Algorithm. *ACM*, 1989.