Lecture Scribe Notes

Prashanth Palanthandalam

The first part of the notes correspond to discussion of the papers *On the Computational Complexity of Maintaining GPS Clock in Packet Scheduling* and *Worst Case Weighted Fair Queuing*.

While simulating the GPS Clock, if the naïve algorithm is used, only a heap data structure is needed to keep track of the GPS Clock. When the new flow arrives at time t, we need to find the value of V_t.

The heap needs to support two operations:

1. ExtractMin
2. Insert

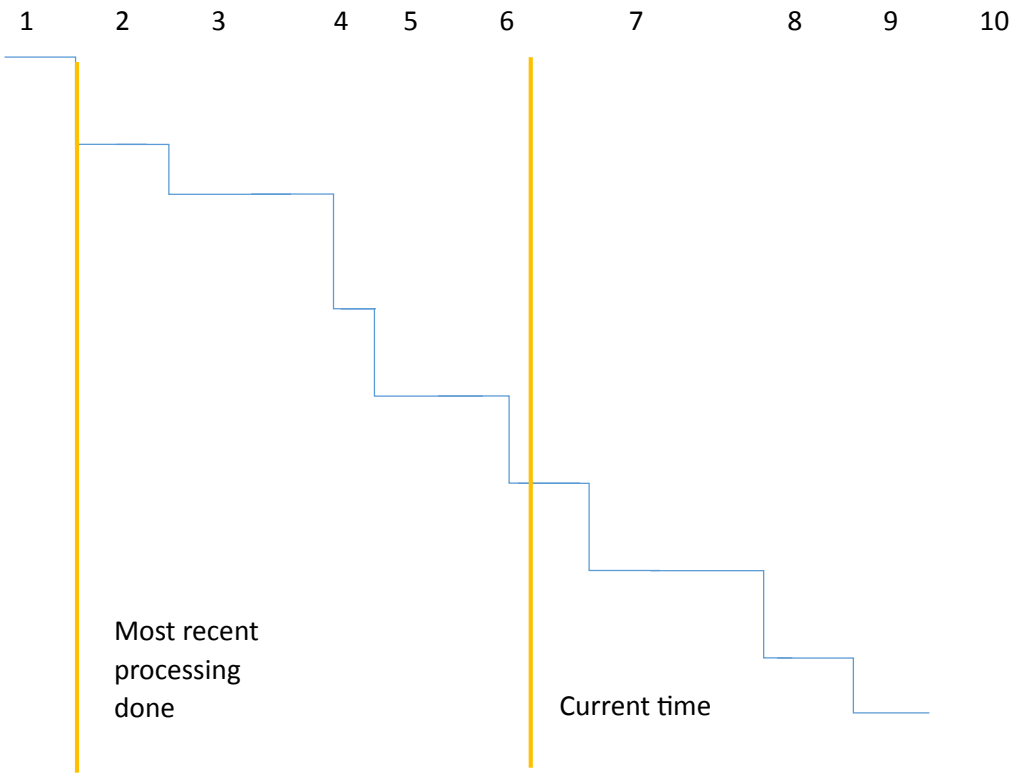
The heap can be replaced by any Abstract Data Type.

Weighted Fair Queuing (WFQ) always selects the path with the lowest GPS finish time.

In the naïve GPS simulation algorithm, only one heap is needed, which contains Head of Line (HOL) packets.

We need a separate heap in order to support ExtractMin, Insert and Delete functions. One heap is required to ensure packets depart in the order of their GPS finish time. This is because, while a packet is being served, a bunch of other packets may arrive.

Advanced Algorithm To Simulate GPS Clock



In the naïve algorithm, if the current time was at the position denoted in the diagram, the processing would have to be done at 6. The algorithm will not be able to keep pace with the current time.

With the advanced algorithm, processing is allowed to lag behind. To keep track of the time lag, a data structure is required. In order to find out the location of current time,

(Current time – Time when last processing was done) = Area under the graph between the two yellow markers.

Therefore, when a new arrival occurs, the position at the current time is known.

This reduces the number of steps from n in the naïve algorithm to logn. However, a heap is not sufficient for this algorithm. An AVL Tree or a Red Black Tree is required.

This data structure augments the existing data structure.

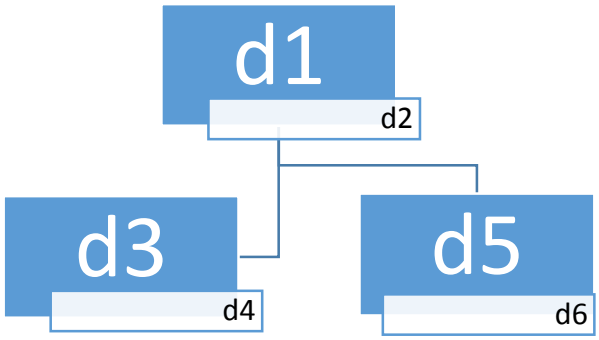
(Augmenting Data Structures, Chapter 14, Introduction to Algorithms by Cormen, Leiserson, Rivest and Stein)

Augmented Data Structures consist of **Data** and an **Invariant**.

For a heap, the invariant is the fact that a parent has a higher index than its children.

Consider an Augmented Data Structure. Let there be a field d_1, which has an invariant inv_1.

d_1 has to satisfy invariant inv_1. In an augmented data structure, an extra field is added to every node.



d_2 has to satisfy its invariant inv_2.

There are three routines, Insert, ExtractMin and Delete. When we execute the routines for d_1, we need to ensure that the invariant for d_2 is not violated as well. In order to satisfy the extra invariant, the old routine might have to be rewritten.

Inorder traversal of the tree results in sorted order of GPS finish time.

When there is a new arrival, the packet is pushed forward, as a result of which re-sorting has to be done to re-form the graph (ladder). As a result the root value might change.