Research Paper Summary and Proposed Changes

Attached research paper schedules the streams in the form of priority buckets(see the below figure).

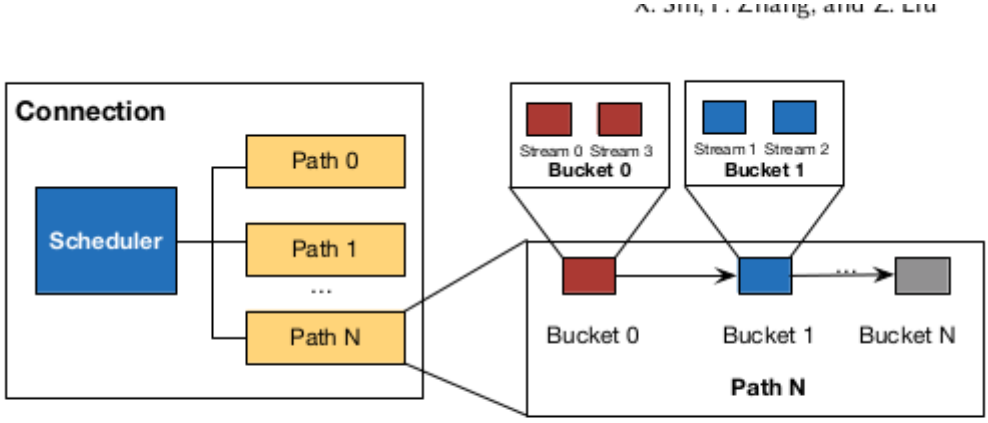


Figure 1: A structure of PriorityBucket design.

- Within a connection there can be multiple paths
- Path may have zero or more streams.
- Global scheduler allocates path to streams based on the proposed scheduling policy.

Stream and Bucket

- stream
  - lighweight bi-directional byte stream abstraction with unique id
  - they can be opened closed or cancelled without tearing down the entire connection.
  - can be scheduled on one or more paths for calculated portion of data
- To prioritize the critical streams on a path we distinguish streams with different priority buckets.
- Streams of equal priority value are divided into the same bucket.(as shown in fig)
- Inside the same bucket transmission follows FCFS.
- Bucket with low priortiy depends on most recent pending stream in higher priority bucket.
- Low priority bucket transmissions can be preempted by the critical or higher priority bucket transmission.

Scheduling Algorithm

Table 1: Scheduling Parameters

Symbol	Meaning
D	total data volume of stream
h_i	total data volume of streams with higher priority
e_i	total data volume of streams with equal priority
T	overall completion time of stream
i	path i
n	number of paths
o_i	estimated one-way delay of the path i
b_i	estimated bandwidth of the path i
d_i	fraction of data assigned to path i
t_i	completion time of stream s on path i

Algorithm 1 PriorityBucket algorithm

```
1:  procedure ScheduleStream
2:      sortedList ← sort paths by ascending order of r_i
3:      initialize d_i = 0, i ∈ [1, 2, ..., n]
4:      D' ← D
5:      for path i in sortedList[1, ..., n − 1] do
6:          r_gap ← r_{i+1} − r_i
7:          if r_gap ≠ 0 and D' > 0 then
8:              if D' >= r_gap × Σ_{k=1}^{i} b_k then
9:                  for j ← 1 to i do
10:                     inc ← r_gap × b_j
11:                     d_j ← d_j + inc
12:                     D' ← D' − inc
13:                 end for
14:             else
15:                 for j ← 1 to i do
16:                     inc ← D' × (b_j / Σ_{k=1}^{i} b_k)
17:                     d_j ← d_j + inc
18:                 end for
19:                 D' ← 0
20:                 break
21:             end if
22:         end if
23:     end for
24:     if D' > 0 then
25:         for path i in sortedList do
26:             d_i ← d_i + D' × (b_i / Σ_{k=1}^{n} b_k)
27:         end for
28:     end if
29: end procedure
```

The paper presents Algorithm1 for scheduling of the streams.
This algorithm can be viewed as the optimization problem where we want to get the lowest value of parameter T.

Algorithm introduces one more parameter r_i.

$r_i = \frac{h_i + e_i}{d_i} + o_i$

$\therefore r_i$ = time required for transmission of equal and higher priority streams + one time delay.

i.e the time required for completion if we do not send any data over path i.

It can be divided in the below steps:

Step 1:

Sort the paths in ascending order of r_i.

Step 2:

The fillgap process schedules data to balance the completion time gap of multiple paths.

It fetches two paths with smallest r_i and will try to schedule balance out the time.

e.g. At first the algorithm will consider path1 and path2 with smallest r_i i.e. r_1 and r_2.

We want to find d_1 such that

$r_1 + \frac{d_1}{b_1} = r_2$ (here b_1 is the bandwidth of path1)

On solving,

$d_1 = (r_2 - r_1) \times b_1$

Similarly this process will be repeated until all the data is allocated or completion time of the paths becomes equivalent.

Step 3:

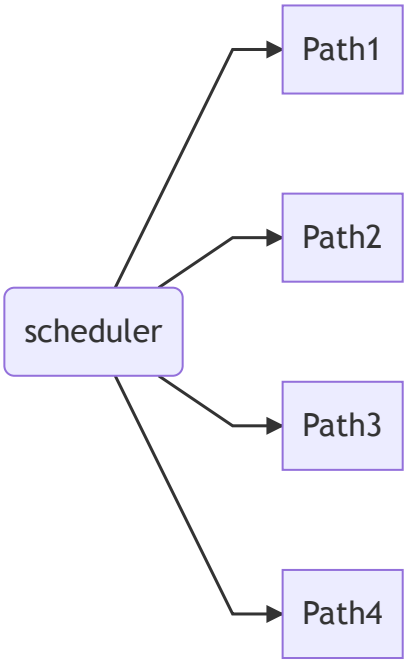
If the data is still remaining then it will be scheduled in the proportion of the path's bandwidth.

The complexity of this algorithm is $O(n^2)$.

Proposed Algorithm Explanation

- The complexity of presented algorithm is $O(n^2)$ but a better algorithm with O(n) time complexity can be derived.
- If the data available to send for the stream is less then the amount being computed by the paths then there will be uneven completion times for the paths.

e.g. Consider the below figure.



Consider that there are 4 paths and data being scheduled computed by above algorithm is d_1, d_2, d_3, d_4 and total amount of data is D.

if D < d_1 + d_2 + d_3 + d_4

for the explanation and understanding purpose assume (D = d_1 + d_2)

Thus, there will be no data being sent over the path 3 and path 4.

Now, when we schedule the data on stream 1 the completion time of path 1 (from now on c_i = completion time of path i), will be equivalent to r_2 and c_2 = r_3.

Since, we are not scheduling any data on the path 3 and path 4 completion time will be 0 for them.

so now the overall completion time will be c_2 = r_3 and c_1 < c_2. Thus if part of the data is sent via path1 it will definitely improve the time and it should be less than r_3.

If before scheduling we compute the data required to be sent across the paths we might be able to improve the time complexity.

Prorposed Algorithm steps

Step 1:
Compute the r_i for all the paths and sort them in ascending order.
We want to compute d_i such that completion time is equal for all the paths.

Thus,

$r_1 + \frac{d_1}{b_1} = r_2 + \frac{d_2}{b_2} = r_3 + \frac{d_3}{b_3} = r_n + \frac{d_n}{b_n}$

Now,

We know that r_n has maximum value. Thus we will assume that we are not sending any data over last path.

then, on solving we get

$d_i = (r_n - r_i) \times b_i$

Step 2:
(D represents the amount of data present in the stream)

If $\sum d_i > D$,
perform step1 without including the path with highest r_i value.
else go to step 3.

This step will ensure that we are using the paths in optimal manner and will avoid unnecessary delay.

Step 3:

If any data is remaining schedule it in proportion with the bandwith of the paths included for comptation of d_i.

Now consider the above scenario,
$D = d_1 + d_2$ computed at first iteration for the previous algorithm.

Now in the new algorithm,

- First we will consider the 4 paths and will try to equalize the delay for all equal to r_4. Hence data values computed for this will be greater than the required data. Hence step1 will be performed again.
- Now, this time we will not consider path 4 and try to equalize delays equal to r_3. Let us assume data amount values computed for this step d_1' and d_2'.

Here, clearly d_2' = d_2 and d_1' > d_1.

$\therefore D < d_2' + d_1'$.

Thus we will again perform to step1 without considering path3.

- Now, in this step we will only compute the value of data to be scheduled on path 1. We will try to equalize delay to r_2.
  Assume data computed for path1 in this step be d_1''.

Clearly, $d_1'' < D$.

Hence we will move to step 3 in this case. In the step 3 we will divide the remaining data between path 1 and path 2. Hence the total completion time is clearly less than that of r_3.

Complexity Analysis

For the proposed new algorithm, step 1 requires O(1) execution time. Depending on the results of step 2 step 1 might needed to be performed O(n) times. In the step 3, we are scheduling the remaining data in proportion to the bandwidth of the paths available. Hence the complexity of step 3 is as well O(n).

Thus overall time complexity is O(n).