Station Assignment with Reallocation

Miguel A. Mosteiro[1], Yulia Rossikova[1], and Prudence W.H. Wong[2(✉)]

[1] Department of Computer Science, Kean University, Union, NJ, USA
{mmosteir,rossikoy}@kean.edu
[2] Department of Computer Science, University of Liverpool, Liverpool, UK
pwong@liverpool.ac.uk

Abstract. We study a dynamic allocation problem that arises in various scenarios where mobile clients joining and leaving have to communicate with static stations via radio transmissions. Restrictions are a maximum delay, or laxity, between consecutive client transmissions and a maximum bandwidth that a station can share among its clients. Clients are assigned to stations so that every client transmits under those restrictions. We consider reallocation algorithms, where clients are revealed at its arrival time, the departure time is unknown until they leave, and clients may be reallocated to another station, but at a cost determined by the laxity We present negative results for related previous protocols that motivate the study; we introduce new protocols that expound trade-offs between station usage and reallocation cost; we prove theoretically bounds on our performance metrics; and we show through simulations that, for realistic scenarios, our protocols behave even better than our theoretical guarantees.

1 Introduction

We study a dynamic allocation problem in scenarios where data on mobile devices has to be gathered and uploaded periodically to one of the static access points available[1]. Examples include *wearable health-monitoring systems*, where data gathered via physiological sensors on ambulatory patients must be periodically uploaded, and *participatory sensing*, where mobile device users upload periodically environment information.

Mobile devices, called *clients*, join and leave continuously, and they communicate with the static access points, called *stations*, The clients' ephemeral nature is modeled by the *life interval* of each client (from its arrival to departure), during which the client has to communicate with some station *periodically*. Periodic communication is modeled by the client's *laxity*, which bounds the maximum duration a client is not transmitting to some stations. The intrinsically shared nature of the access to stations is modeled by a maximum shared *station bandwidth*, by a *client bandwidth*, and by the client laxity.

[1] We consider an upstream model, but the same results apply to downstream communication.

E. Bampis (Ed.): SEA 2015, LNCS 9125, pp. 151–164, 2015.
DOI: 10.1007/978-3-319-20086-6_12

Based on the above model, we study the problem of assigning clients to stations so that every client transmits to some stations satisfying the laxity and bandwidth constraints. We consider settings where clients are revealed at its arrival time and their departure time is only revealed when they depart (as in online algorithms). Clients may be reassigned from one station to another and we call such reassignment *reallocation*. Intuitively reallocation causes more disturbance to a client with small laxity. Therefore, we assume reallocation incurs a cost inversely proportional to a client's laxity. We aim to reduce the number of active stations and reduce the reallocation cost. However, these two goals are orthogonal, e.g., we can reallocate the clients every time a client arrives/departs so that the number of active stations is minimized while incurring a very high reallocation cost; alternatively we can keep the reallocation cost to zero but we may use many active stations after a sequence of client departures. In this paper, we aim to obtain a balance between the two performance metric. We call this problem *Station Assignment Problem with Reallocation* (SA).

Previous Work. The closest work to the present paper is [12], where reallocation algorithms were presented for Windows Scheduling (WS). The WS problem [6,7,11,12] is a particular case of SA where the bandwidth requirement of each client is the same and each channel (a.k.a. station in our case) can only serve one client at a time. In [12], a unit cost is incurred for each client reallocated and the objective is to minimize an aggregate sum reflecting the amortized reallocation cost and the number of channels used. A protocol called Classified Reallocation is shown to guarantee an amortized constant number of reallocations. This protocol is also evaluated experimentally together with two other protocols Preemptive Reallocation and Lazy Reallocation.

WS [6,7,11] was first studied without reallocation and the objective function was the number of channels. Both static case where clients never depart [6,7] and dynamic case where clients may depart [11] have been studied. For the dynamic case, the comparison is against peak load which may occur at different time in the online algorithm and the optimal offline algorithm. In [12] and this work, we compare against current load. Clients with same laxity were considered in [14]. We also extend the objective function in [12] such that the number of reallocated clients is weighted inversely by their laxity, and we provide trade-off between reallocation cost and number of stations.

SA and Other Assignment Problems. Our problem differs from various scheduling problems. The load balancing problem [4] also assigns tasks of different load to servers, yet does not consider periodic tasks and disallow reallocation. Interval coloring [1] concerns the number of machines used but not periodic tasks. Periodic appearance of tasks in real time scheduling [8] is determined by the input but not by the algorithm to satisfy laxity constraint. The SA problem is also related to b-matching [15], fractional matching [5], and adwords [13]. Among other details, the objective function is different.

There are two typical approaches of handling orthogonal objectives: to minimize the summation of two costs, e.g., energy efficient flow time scheduling minimizes the sum of energy usage and total flow time of the tasks [2]; and to

formulate two approximation ratios, e.g., energy efficient throughput scheduling algorithm is t-throughput-competitive and e-energy-competitive [10]. We adopt the later approach.

Reallocation has been considered in scheduling [3,9,16]. In [9], a distinction is made between reassignment within server (reschedule) and between servers (migration). Here, we assume rescheduling within a station is free and we use "reallocation" to refer to reassignment to other stations. It is often that the number/size of jobs reallocated is bounded, e.g., by a function of the number of jobs in the system [9], the size of the arriving job [16] or the number of machines [3]. In our problem, we bound the reallocation by the weight (cumulative inverse laxity) of the clients departed.

2 Our Results

In this paper, we study reallocation algorithms for SA assuming that clients have arbitrary laxities and bandwidth requirements, that clients depart from the system at arbitrary times, and that they may be reallocated, but at some cost proportional to the resources needed. Specifically, our contributions are the following.

- We define a characterization of SA reallocation algorithms, which we call (α, β)- approximation, as a combination of the competitive ratio on station usage (α) and the cost of reallocations contrasted with the resources released by departures (β).
- We show a sequence of negative results proving that worst-case guarantees cannot be provided by previous protocols Classified Reallocation and Preemptive Reallocation [12], even if they are modified to our reallocation cost function.
- We present a novel SA protocol called Classified Preemptive Reallocation (CPR) where clients are *classified* according to laxity and bandwidth requirements, and upon departures the remaining clients are *preemptively* reallocated to minimize station usage, but only within their class. The protocol presented includes a range of classifications that exposes trade-offs between reallocation cost and station usage. In fact, we found first experimentally what is the classification function that better balances these goals, and then we provided theoretical guarantees for all functions.
- In our main theorem, we prove bounds on both of our performance metrics, and we instantiate those bounds into three classifications and for specific scenarios in two corollaries (refer to Section 5 for the specific bounds.)
- Finally, we present the results of our extensive simulations that allowed us to find the function that best balances station usage and reallocation cost. Additionally, our simulations show that, for a variety of realistic scenarios, CPR performs better than expected by the worst-case theoretical analysis, and close to optimal on average.

3 Definitions

Model. We consider a set S of stations and a set C of clients. Each client must transmit packets to some station. Time is slotted so that each time slot is long enough to transmit one packet. A client can be assigned to transmit to only one station in any given time slot. Starting from some initial time slot 1, we refer to the infinite sequence of time slots $1, 2, 3, \ldots$ as ***global time***. Each client $c \in C$ is characterized by an ***arrival time*** a_c and a ***departure time*** d_c, that define a ***life interval*** $\tau_c = [a_c, d_c]$ in which c is ***active***. That is, client c is active from the beginning of time slot a_c up to the end of time slot d_c. We define $C(t) \subseteq C$ to be the set of clients that are active during time slot t. With respect to resources required, each client c is characterized by a ***bandwidth*** requirement b_c, and a ***laxity*** $0 < w_c \leq |\tau_c|$, such that c must transmit to some station in S at least one packet within each w_c consecutive time slots in τ_c[2]. On the other hand, each station $s \in S$ is characterized by a ***station bandwidth*** or ***capacity*** B, which is the maximum aggregated bandwidth of clients that *may* transmit to s in each time slot.

Notation. Let the ***schedule*** of a client c be an infinite sequence σ_c of values from the alphabet $\{0\} \cup S$. Let $\sigma_c(t)$ be the t^{th} value of σ_c. A ***station assignment*** is a set σ of schedules that models the transmissions from clients to stations. That is, for each client $c \in C$ and time slot t, it is $\sigma_c(t) = s$ if c is scheduled to transmit to station $s \in S$ in time slot t, and $\sigma_c(t) = 0$ if c does not transmit in time slot t. If a client c is scheduled to transmit to a station s we say that c is ***assigned*** to station s. We say that a station that has clients assigned is ***active***, and ***inactive*** or ***empty*** otherwise.

Problem. The ***Station Assignment problem (SA)*** is defined as follows. For a given set of stations and set of clients, obtain a station assignment such that (i) each client transmits to some station at least once within each period of length its laxity during its life interval, (ii) in each time slot, no station receives from clients whose aggregated bandwidth is more than the station capacity. Notice that, for any finite set of stations, there are sets of clients such that the SA problem is not solvable. We assume in this work that S is infinite and what we want to minimize is the number of *active* stations.

Algorithms. We study ***reallocation algorithms*** for SA. That is, the parameters w_c and b_c needed to assign the client to some station are revealed at time a_c, but the departure time d_c is unknown to the algorithm until the client actually leaves the system (as in online algorithms). Then, at the beginning of time slot t, an SA reallocation algorithm returns the transmission schedules of all clients that are active in time slot t, possibly reassigning some clients from one station to another. (I.e., the schedules of clients that were already active may be changed from one time slot to another.) We refer to the reassignment of one client as

[2] To maintain low station usage, we will assume that the laxity can be relaxed during reallocation.

a ***reallocation***, whereas all the reassignments that happen at the beginning of the same time slot are called a ***reallocation event***.

Performance Metric. Previous work [12] has considered the number of clients reallocated as the reallocation cost. In the present work, we consider a different scenario where the cost of reallocating a client is proportional to resources requested by that client. Specifically, we assume a cost for the reallocation of each client c of ρ/w_c, where $\rho > 0$ is a parameter. Then, letting $\mathcal{R}(ALG, t)$ be the cost of the reallocation event incurred by algorithm ALG at time t, and $R(ALG, t)$ be the set of clients being reallocated, the overall cost is the following.

$$\mathcal{R}(ALG, t) = \rho \sum_{c \in R(ALG,t)} \frac{1}{w_c}. \tag{1}$$

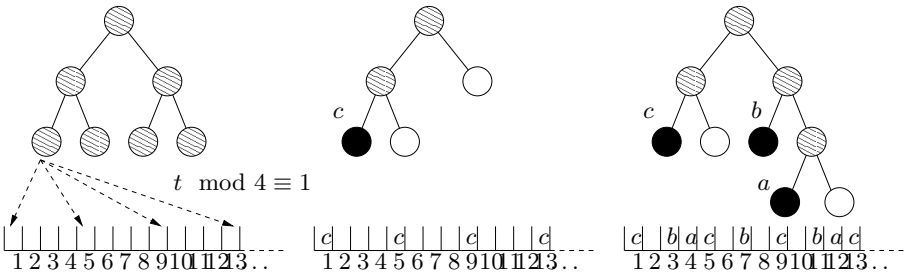
We will drop the specification of the algorithm whenever clear from the context.

With respect to performance, we aim for algorithms with low reallocation cost and small number of active stations. Unfortunately, these are contradictory goals. Indeed, the reallocation cost could be zero if no client is reallocated (online algorithm), but the number of active stations could be as big as the number of active clients (e.g. initially multiple clients assigned to each station, and then all but one client from each active station depart). On the other hand, the number of active stations could be minimized applying an offline algorithm on each time slot, but the reallocation cost could be large. Thus, we characterize algorithms with both metrics as follows.

For any SA algorithm ALG, let $S(ALG, t)$ be the number of active stations at time t in the schedule, let $D(ALG, t)$ be the set of clients departed since the last reallocation up to time t. Denoting $\sum_{c \in C'} 1/w_c$ as the ***weight*** of the clients in $C' \subseteq C$, let $\mathcal{D}(ALG, t)$ be the weight of the clients departed since the last reallocation up to time t, that is, $\mathcal{D}(ALG, t) = \sum_{c \in D(ALG,t)} 1/w_c$. Also, we denote the minimum number of active stations needed at time t as $S(OPT, t)$. Throughout, we will drop the specification of the algorithm whenever it is clear from the context. Then, we say that an SA reallocation algorithm ALG achieves an (α, β)***-approximation*** if the following holds for any input.

$$\max_t \frac{S(ALG, t)}{S(OPT, t)} \leq \alpha$$
$$\max_t \frac{\mathcal{R}(ALG, t)}{\mathcal{D}(ALG, t)} \leq \beta.$$

In words, the overhead on the number of stations used by ALG is never more than a multiplicative factor α over the optimal, and the reallocation cost, amortized on the "space" left available by departing clients is never more than β. The latter is well defined since reallocations only occur after departures. Notice that these ratios are strong guarantees, in the sense that they are the maximum of the ratios instead of the ratio of the maxima. (This distinction was called previously in the literature *against current load* versus *against peak load* respectively.) Moreover, the reallocation ratio computed as the maximum *over reallocation events* is also stronger than the ratio of cumulative weights since the system started.



(a) Mapping node - timeslot. (b) First client assigned. (c) Some clients assigned.

Fig. 1. Illustration of a binary broadcast tree. (a) A depth-2 tree corresponds to periodic broadcast of period 2^2. (b) Clients are assigned to leaves, e.g., client c with laxity 4 is assigned the black node meaning timeslot $1, 5, 9$, etc. are reserved for it. (c) Open leaf (white node) corresponds to available slot.

4 Algorithms

Broadcast Trees. A common theme in WS algorithms with *periodic* transmission schedules is to represent those schedules with *Broadcast Trees* [6,11,12]. See Figure 1 for illustration. Throughout the paper, we refer to a set of broadcast trees as the ***forest***, and to the distance in edges from a node to the root as the ***depth***. Generalizing, the 2^d nodes at depth d in a complete binary tree represent the time slots t mod 2^d (see Figure 1(a)). Then, to indicate that some (periodic) time slot has been reserved for a client c to transmit to a given station s, we say informally that c is assigned to the corresponding node in the broadcast tree of s. Throughout the rest of the paper, we use both indistinctively. Refer to [6,11] for further details on broadcast trees.

WS Algorithms. Chan et al. [11] presented a WS algorithm preserving the following invariant. For each station, the broadcast tree has at most one available leaf at each depth. In order to preserve this invariant, when a client departs, the remaining clients in the same tree are rearranged. If reallocations among trees are possible, the algorithm *Preemptive Reallocation* (PR) [12] extended the same idea to all trees, maintaining the invariant that *throughout all trees* there is at most one available leaf at each depth. For laxities that are powers of 2, PR achieves an optimal station usage. However, we show in Lemma 1 (1) and (2) that simple modification to PR leads to negative results.

A WS algorithm with provable bounded reallocation cost guarantees was shown also in [12]. The protocol, called *Classified Reallocation* (CR), guarantees that all clients assigned to the same station have the same laxity, except for one distinguished station that handles all laxities linear and above. To attain constant amortized reallocation cost, clients are moved to/from the distinguished station only after the number of clients has halved/doubled. However, for the reallocation cost function in Equation 1, CR has an arbitrarily bad reallocation cost ratio, as we show in Lemma 1 (3).

Algorithm 1. Classified Preemptive Reallocation. $\lfloor\lfloor x \rfloor\rfloor$ is the largest power of 2 that is not larger than x. We represent the transmission schedules with broadcast trees. A node with both children available becomes an available leaf. A station with no client assigned becomes non-active. $\langle w_{low}, w_{high} \rangle$ are the boundaries of the class of the input client.

```
1  Algorithm
2      upon arrival or departure of a client c do
3          if arrival then allocate(c, ⟨w_low, w_high⟩)
4          else consolidate(c, ⟨w_low, w_high⟩)
5  Procedure allocate(c, ⟨w_low, w_high⟩)
6      for each depth i = ⌊log w_c⌋ − ⌈log w_low⌉ down to 0 do
7          for each active station s of class ⟨w_low, w_high, 1/⌊⌊B/b_c⌋⌋⟩ do
8              if there is a leaf ℓ available at depth i in the broadcast tree of s then
9                  allocate to ℓ a new subtree with client c assigned at depth
                   ⌊log w_c⌋ − i − ⌈log w_low⌉ of the broadcast subtree
10                 return
11     activate a new station s in class ⟨w_low, w_high, 1/⌊⌊B/b_c⌋⌋⟩
12     choose one of the leaves ℓ at depth 0 of the broadcast subtrees of s
13     allocate to ℓ a new subtree with client c assigned at depth
       ⌊log w_c⌋ − ⌈log w_low⌉ of the broadcast subtree
14 Procedure consolidate(c, ⟨w_low, w_high⟩)
15     for each depth i = ⌊log w_c⌋ − ⌈log w_low⌉ down to 1 do
16         if there are two active stations of class ⟨w_low, w_high, 1/⌊⌊B/b_c⌋⌋⟩ both
           with a leaf at depth i available then reallocate sibling subtree of
           smaller weight
17         else return
       // reallocations cleared a whole broadcast subtree
18     if there are two active stations of class ⟨w_low, w_high, 1/⌊⌊B/b_c⌋⌋⟩ with
       empty broadcast subtrees then reallocate a subtree from the station with at
       least one empty subtree to the station with exactly one empty subtree
```

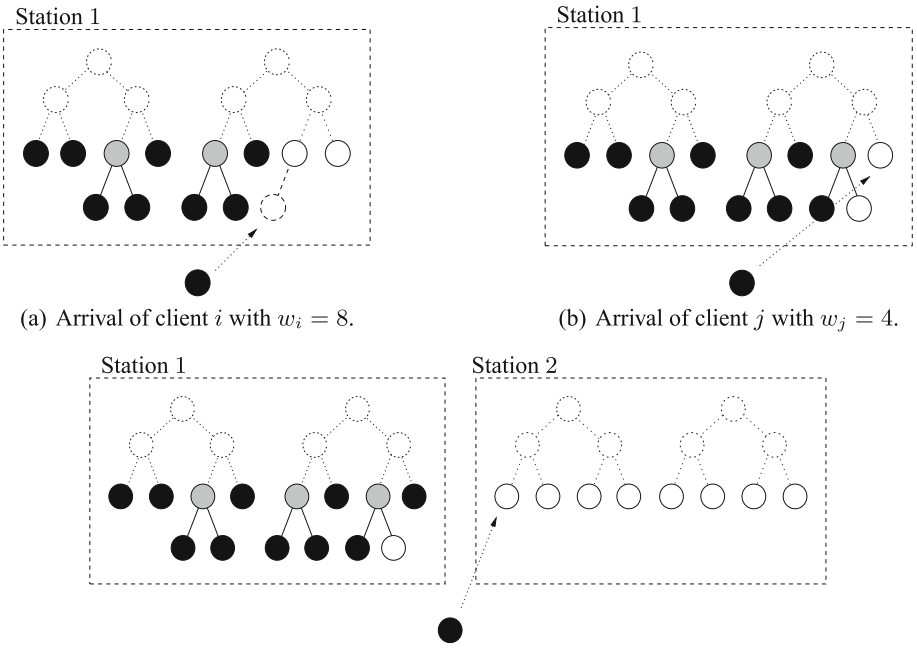
Classified Preemptive Reallocation. The negative results in Lemma 1 apply to WS. Given that WS is a particular case of SA fixing $b_c = B$ for all clients, the same negative results apply to SA. Thus, should the reallocation cost be maintained low, a new approach is needed. We present now an online SA protocol (Algorithm 1), which we call *Classified Preemptive Reallocation* (CPR), that provides guarantees in channel-station usage and reallocation cost. The protocol may be summarized as follows. Clients are classified according to laxity and bandwidth requirements. Upon arrival, a client is allocated to a station within its corresponding class to guarantee a usage excess (with respect to optimal) of at most one station per class plus one station throughout all classes. Upon departure of a client, if necessary to maintain the above-mentioned guarantee, clients are reallocated, but only within the corresponding class. The protocol includes

three different classifications providing different trade-offs between reallocation cost and station usage. We recreate the idea of broadcast trees, but now we have multiple trees representing the schedule of each station. On one hand, we use broadcast trees with depth bounded by the class laxities. We call them ***broadcast subtrees*** to reflect that they are only part of a regular broadcast tree. On the other hand, we have the multiplicity yielded by the shared station capacity B. An example of broadcast subtrees can be seen in Figure 2. Further details follow.

The mechanism to allocate an arriving client can be described as follows. Upon arrival, a client c is classified according to its laxity and bandwidth requirement. Specifically, c is assigned to a class for clients with bandwidth requirement $B/\lfloor\lfloor B/b_c\rfloor\rfloor$ and laxity in $[w_{low}, w_{high})$, for some w_{low} and w_{high} that depend on the classification chosen. Notice that each station has up to $\lfloor\lfloor B/b_c\rfloor\rfloor \cdot \lceil\lceil w_{low}\rceil\rceil$ subtrees. That is, $\lfloor\lfloor B/b_c\rfloor\rfloor$ ways to share its capacity B and $\lceil\lceil w_{low}\rceil\rceil$ ways to share its schedule (see Figure 2). Within its class, we assign c to an available leaf at depth $\lfloor \log w_c\rfloor - \lceil \log w_{low}\rceil$ in any subtree in the forest (see Figure 2(b)). If there is no such leaf available, we look at smaller depths up in the forest one by one. If we find an available leaf at depth $\lceil \log w_{low}\rceil \leq i < \lfloor \log w_c\rfloor - \lceil \log w_{low}\rceil$, we allocate to that leaf a new subtree with c assigned at depth $\lfloor \log w_c\rfloor - i$ with respect to the root of the broadcast subtree (see Figures 2(a) and 2(c)). If no such leaf is available at any depth, a new broadcast subtree T is created with c assigned at depth $\lfloor \log w_c\rfloor - \lceil \log w_{low}\rceil$, and T is assigned to a newly activated station. Refer to Algorithm 1 for further details.

The above allocation mechanism maintains the following invariant: (1) there is at most one leaf available at any depth larger than $\lceil \log w_{low}\rceil$ of the forest, and (2) there is at most one station with leaves available at depth $\lceil \log w_{low}\rceil$ (an empty broadcast subtree). When a client departs, this invariant is re-established through reallocations as follows. When a client c departs, if $\lfloor \log w_c\rfloor > \lceil \log w_{low}\rceil$, we check if there was already a leaf ℓ available at depth $\lfloor \log w_c\rfloor - \lceil \log w_{low}\rceil$. If there was one, either the sibling of c or the sibling of ℓ has to be reallocated to re-establish the invariant. We greedily choose to reallocate whichever sibling has smaller weight of the two (see Figure 3(a)). The process does not necessarily stop here because, if $\lfloor \log w_c\rfloor - 1 > \lceil \log w_{low}\rceil$ and there was a leaf already available at depth $\lfloor \log w_c\rfloor - 1 - \lceil \log w_{low}\rceil$, together with the newly available leaf at depth $\lfloor \log w_c\rfloor - 1 - \lceil \log w_{low}\rceil$ due to the reallocation at depth $\lfloor \log w_c\rfloor - \lceil \log w_{low}\rceil$, it yields two leaves available at depth $\lfloor \log w_c\rfloor - 1 - \lceil \log w_{low}\rceil$. Hence, again one of the sibling subtrees has to be reallocated (see Figure 3(b)). This transitive reallocations upwards the forest may continue until a depth where no reallocation is needed or until the depth $\lceil \log w_{low}\rceil + 1$ is reached, when the reallocation leaves a broadcast subtree empty. In the latter case, we reallocate a whole broadcast subtree so that only one station has empty subtrees and the invariant is re-established. Refer to Algorithm 1 for further details.

Notice that when a client is reallocated (even within a station) its laxity may be violated once. Consider for instance the schedule in Figure 1(c). Let $w_a = 4$, that is, a is transmitting at its lowest possible frequency. If at the end of time



(a) Arrival of client i with $w_i = 8$.

(b) Arrival of client j with $w_j = 4$.

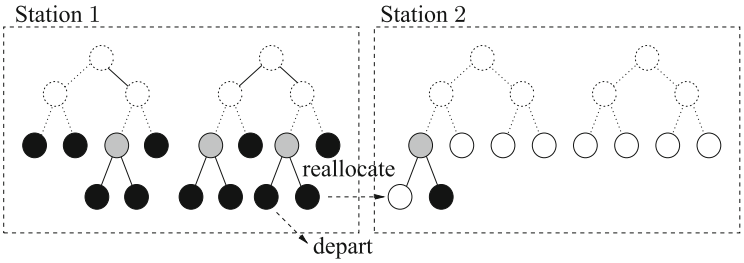
(c) Arrival of client k with $w_k = 4$.

Fig. 2. Illustration of allocation mechanism. Class: laxities $[4, 16)$, bandwidth $1/2$. Subtrees are depicted connected to a broadcast tree to reflect their location in the station schedule.

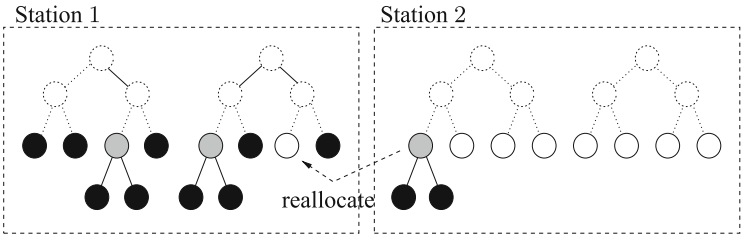
slot 7 client b departs, at the beginning of time slot 8 client a will be reallocated to the slot of client b, that is, to transmit next in slot 11. This new schedule violates w_a because the previous slot when a transmitted was 5. For WS, in [11] the issue is approached making a client transmit once more within the original schedule. As the authors say, this approach introduces a transition delay. In their model, there is no impact on station usage because their ratio is against peak load. However, for a ratio against current load such as our model, reserving a slot for a client in more than one station implies an overhead on channel usage. Indeed, for any given allocation/reallocation policy, an adversarial input can be shown so that either the laxity is stretched or the channel usage is not optimal. Hence, in our model we assume that when a client is reallocated the laxity may be stretched, folding the cost in the reallocation cost.

5 Analysis

We start with negative results in Lemma 1, which apply to WS, and to SA fixing $b_c = B$ for all clients. The proofs, left to the full paper, are all based on showing an adversarial client set for which the claim holds.



(a) Departure of client j with $w_j = 4$.



(b) Upwards reallocation of sibling with smaller weight.

Fig. 3. Illustration of reallocation mechanism. Class: laxities $[4, 16)$, bandwidth $1/2$. After the second reallocation Station 2 is left empty and, hence, deactivated. Subtrees are depicted connected to a broadcast tree to reflect their location in the station schedule.

Lemma 1. *1. There exists a client arrival/departure schedule such that, in Preemptive Reallocation [12], the ratio of number of clients reallocated against the number of arrivals plus departures is unbounded.*

2. For Preemptive Reallocation [12], modified so that the sibling subtree of smaller **weight** *is reallocated to restore the invariant, rather than the subtree with less clients, the following holds. For any $d > 0$, there exists a client arrival/departure schedule such that it is $\max_t \mathcal{R}(t)/\mathcal{D}(t) \geq \rho(2^d - 1)^2/2^d$.*

3. For any integer $x > 0$ and any $w \geq 2^{x+5}$ arbitrarily big such that w is a power of 2, there exists a client arrival/departure schedule such that, in Classified Reallocation [12], it is $\max_t \mathcal{R}(t)/\mathcal{D}(t) \geq \frac{\rho/4}{7 \cdot 2^x} w$.

The above lemma shows that the application of previous WS reallocation algorithms to SA is not feasible. Theorem 1 gives guarantees on station usage and reallocation cost for CPR. The proof, left to the full paper, shows that the invariant is re-established after each arrival or departure. Then, competitiveness on station usage is derived from the invariant properties. Finally, to bound β, a worst case scenario minimizing the weight of departed clients and maximizing the reallocated weight is shown. To provide intuition and comparison for the simulations, we instantiate Theorem 1 on a setting where all laxities are powers of 2 and all bandwidth requirements are the full capacity of a station.

Theorem 1. *At any time slot t, CPR achieves an (α, β)-approximation as follows.*

$$\alpha = \max_t \frac{4(1 + \Gamma(ALG, t) + S(OPT, t))}{S(OPT, t)}$$
$$\beta = \max_t \rho(2\lfloor\lfloor w_{high_{\max}}(t)\rfloor\rfloor / \lceil\lceil w_{low_{\max}}(t)\rceil\rceil - 1).$$

Where $\Gamma(ALG, t)$ is the number of classes used by CPR at time t, and $w_{high_{\max}}(t)$ and $w_{low_{\max}}(t)$ are the maximum upper and lower limits of a class at time t.

Corollary 1. *For a set of clients C such that, for all $c \in C$, it is $b_c = B$ and $w_c = 2^i$ for some $i \geq 0$, and for all t it is $w_{\max}(t) > w_{\min}(t) \geq 4$, the following holds. At any time slot t, CPR achieves an (α, β)-approximation as follows.*

1. *If the client classification boundaries are $[w_i, w_{i+1})$, where $w_1 = 1$, and $w_i = 2w_{i-1}$, for any $i > 1$, then*

$$\alpha = 1 + (2 + \log(w_{\max}(t)/w_{\min}(t)))\,/H(C(t))$$
$$\beta = 3\rho.$$

2. *If the client classification boundaries are $[w_i, w_{i+1})$, where $w_1 = 1, w_2 = 2, w_3 = 4$, and $w_i = w_{i-1} \log w_{i-1}$, for any $i > 3$, then*

$$\alpha = 1 + (2 + \log w_{\max}(t)/\log\log w_{\min}(t))\,/H(C(t))$$
$$\beta = \rho(2\log w_{\max}(t) - 1).$$

3. *If the client classification boundaries are $[w_i, w_{i+1})$, where $w_1 = 1, w_2 = 2$, and $w_i = w_{i-1}^2$, for any $i > 2$, then*

$$\alpha = 1 + (2 + \log(\log w_{\max}(t)/\log w_{\min}(t)))\,/H(C(t))$$
$$\beta = \rho\left(2\sqrt{w_{\max}(t)} - 1\right).$$

Where $H(C(t)) = \lceil\sum_{c \in C(t)} 1/w_c\rceil$, $w_{\max}(t) = \max_{c \in C(t)} w_c$, $w_{\min}(t) = \min_{c \in C(t)} w_c$, $b_{\max}(t) = \max_{c \in C(t)} b_c$, and $b_{\min}(t) = \min_{c \in C(t)} b_c$.

6 Simulations

In this section, we present the main experimental simulations results of the CPR algorithm. We highlight here that the classification factor (logarithmic) that balances station usage and reallocation cost was found through experimentation with various functions. For the specific cases presented (constant, logarithmic, and linear factors) we have focused on a scenario where $\forall c \in C, b_c = B$ and $w_c = 2^i, i \geq 0$ (as in Corollary 1). Simulations for arbitrary bandwidths and laxities are left to the full version of this paper.

To evaluate thoroughly the performance of our protocol, we have produced various sets of clients (recall that each client is characterized by arrival time,

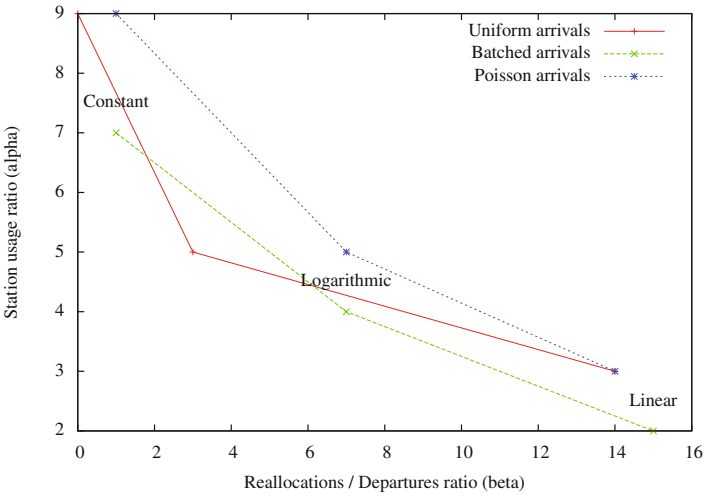


Fig. 4. Worst case α vs. β. $|C| = 4000$, $w_{\max} = 1024$, $w_{\min} = 1$, $\rho = 1$.

departure time, and laxity). The laxity of each client was chosen independently at random from $\{1, 2, 4, \ldots, 1024\}$, with a distribution biased towards large laxities. More precisely, for each client c, $w_c = 1$ with probability $1/1024$, or $w_c = 2^i$ with probability $2^i/2^{11}$, for $1 \leq i \leq 10$. For $n = 4000$ clients, time was discretized in $2n$ slots. The arrival time of each client was chosen: (a) uniformly at random within $[1, 2n]$; (b) in 3 batches of $n/3$ clients arriving at $t = 1$, $t = n/2$, and $t = n$; and (c) as a Poisson process with rate 0.7. For each client, the departure time was chosen uniformly at random from the interval $[t_a, 2n]$, where t_a is the time of arrival of such client. With respect to the protocol, three different classification factors: constant, logarithmic, and linear, were used.

For each of the nine scenarios arising from the combination of the variants, we evaluated experimentally the (α, β)-approximation of CPR. Our simulations showed that the performance in practical settings is as expected or better than the theoretical bounds. The reallocation vs. departures weight ratio (bounded by β) is around 1 most of the time for all three algorithms. On the other hand, after a period upon initial arrivals and a period before last departures, the station usage ratio against $H(C(t))$, which is only a lower bound of the optimal, (bounded by β) is most of the time below 2.

To evaluate the behavior of our algorithms in adverse conditions, we extended the number of cases considering $|C| = 4000, 8000$, and 16000 clients, and the range of laxities to $\{16, 32, 64, \ldots, w_{\max}\}$, for $w_{\max} = 1024, 4096$, and 16384. The laxities were drawn uniformly at random. These cases, combined with the arrival distributions and the classification factors, yielded 81 scenarios tested. We observed that the trade-offs between α and β according to the algorithm used apply to all these scenarios. Indeed, having more clients and setting higher $w_{\max}$ does not affect the trade-offs, only their magnitude as expected from the functions bounding α and β in Corollary 1. Should the reallocation ratio be

minimized, the constant factor classification achieves better performance at a higher station usage. On the other hand, if channel usage must be kept low, the linear factor classification performs better incurring in higher reallocation cost. The logarithmic factor balances both costs. Figure 4 illustrates these trade offs for one of the scenarios. In comparison with the bounds proved in Corollary 1, for the scenarios simulated CPR behaves better than expected.

Acknowledgments. Authors would like to thank Martín Farach-Colton for useful discussions. This work has been supported in part by the National Science Foundation (CCF-1114930); Kean University UFRI grant; U. of Liverpool Departmental Visiting Fellowship; U. of Liverpool Network Sciences & Technologies (NeST).

References

1. Adamy, U., Erlebach, T.: Online coloring of intervals with bandwidth. In: Solis-Oba, R., Jansen, K. (eds.) WAOA 2003. LNCS, vol. 2909, pp. 1–12. Springer, Heidelberg (2004)
2. Albers, S., Fujiwara, H.: Energy-efficient algorithms for flow time minimization. ACM Trans. on Algorithms **3**(4), 49 (2007)
3. Albers, S., Hellwig, M.: On the value of job migration in online makespan minimization. In: Epstein, L., Ferragina, P. (eds.) ESA 2012. LNCS, vol. 7501, pp. 84–95. Springer, Heidelberg (2012)
4. Azar, Y.: On-line load balancing. In: Fiat, A. (ed.) Online Algorithms 1996. LNCS, vol. 1442, pp. 178–195. Springer, Heidelberg (1998)
5. Azar, Y., Litichevskey, A.: Maximizing throughput in multi-queue switches. Algorithmica **45**, 69–90 (2006)
6. Bar-Noy, A., Ladner, R.E.: Windows scheduling problems for broadcast systems. SIAM Journal on Computing **32**(4), 1091–1113 (2003)
7. Bar-Noy, A., Ladner, R.E., Tamir, T.: Windows scheduling as a restricted version of bin packing. ACM Trans. on Algorithms **3**(3), 28 (2007)
8. Baruah, S., Goossens, J.: Scheduling real-time tasks: algorithms and complexity. In: Leung, J. (ed.) Handbook of Scheduling: Algorithms, Models and Performance Analysis, pp. 15-1–15-41. CRC Press (2004)
9. Bender, M.A., Farach-Colton, M., Fekete, S.P., Fineman, J.T., Gilbert, S.: Reallocation problems in scheduling. In: SPAA, pp. 271–279 (2013)
10. Chan, H.-L., Chan, J.W-T., Lam, T.W., Lee, L.-K., Mak, K.-S., Wong, P.W.H.: Optimizing throughput and energy in online deadline scheduling. ACM Trans. on Algorithms **6**(1) (2009)
11. Chan, W.-T., Wong, P.W.H.: On-line windows scheduling of temporary items. In: Fleischer, R., Trippen, G. (eds.) ISAAC 2004. LNCS, vol. 3341, pp. 259–270. Springer, Heidelberg (2004)
12. Farach-Colton, M., Leal, K., Mosteiro, M.A., Thraves, C.: Dynamic windows scheduling with reallocation. In: Gudmundsson, J., Katajainen, J. (eds.) SEA 2014. LNCS, vol. 8504, pp. 99–110. Springer, Heidelberg (2014)
13. Feldman, J., Mehta, A., Mirrokni, V., Muthukrishnan, S.: Online stochastic matching: beating 1–1/e. In: FOCS, pp. 117–126 (2009)

14. Fernández Anta, A., Kowalski, D.R., Mosteiro, M.A., Wong, P.W.H.: Station assignment with applications to sensing. In: Flocchini, P., Gao, J., Kranakis, E., der Heide, F.M. (eds.) ALGOSENSORS 2013. LNCS, vol. 8243, pp. 155–166. Springer, Heidelberg (2014)
15. Kalyanasundaram, B., Pruhs, K.: An optimal deterministic algorithm for online b-matching. Theoretical Computer Science **233**(1–2), 319–325 (2000)
16. Sanders, P., Sivadasan, N., Skutella, M.: Online scheduling with bounded migration. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 1111–1122. Springer, Heidelberg (2004)