Noname manuscript No.
(will be inserted by the editor)

Station Assignment with Reallocation

Austin Halper · Miguel A. Mosteiro · Yulia Rossikova · Prudence W.H. Wong

Received: date / Accepted: date

Abstract We study a dynamic allocation problem that arises in various scenarios where mobile clients joining and leaving the system 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. We study the problem of assigning clients to stations so that every client transmits to some station, satisfying 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 proportional

A preliminary version of this work appeared in SEA 2015 [34].

The authors appear in alphabetical order.

Austin Halper
Pace University
Computer Science Dept.
New York, NY, USA
E-mail: ah17939n@pace.edu

Miguel A. Mosteiro
Pace University
Computer Science Dept.
New York, NY, USA
E-mail: mmosteiro@pace.edu

Yulia Rossikova
Kean University
Computer Science Dept.
Union, NJ, USA
E-mail: rossikoy@kean.edu

Prudence W.H. Wong
University of Liverpool
Dept. of Computer Science
Liverpool, UK
E-mail: pwong@liverpool.ac.uk

to the reciprocal of the client's laxity. We present negative results for previous related protocols that motivate the study; we introduce new protocols that expound trade-offs between station usage and reallocation cost; we determine experimentally a classification of the clients attempting to balance those opposite goals; we prove theoretically bounds on our performance metrics; and we show through simulations that, for realistic scenarios, our protocols behave much better than our theoretical guarantees.

Keywords Base Station Assignment · Reallocation Algorithms · Competitive Analysis · Radio Networks

1 Introduction

We study a dynamic allocation problem that arises in various scenarios where data on mobile devices has to be gathered and uploaded periodically to one of the many static access points available [1]. Examples include *wearable health-monitoring systems*, where ambulatory patients carry physiological sensors and the data gathered must be periodically uploaded, and *participatory sensing* [33, 35], where communities of mobile device users upload periodically information about their environment. For example, in the SPA system [38], sensors are attached to participants periodically sampling the heart rate, blood pressure, movement etc.; while in the MobGeoSen application [32], mobile phones update periodically their geo-location and associated environment. Depending on individuals the frequency different participants need to communicate may differ, e.g., depending on the health conditions.

Mobile devices, called *clients*, join and leave the system continuously, and they communicate with the static access points, called *stations*, via radio transmissions. The ephemeral nature of the clients is modeled by characterizing each client with a *life interval* (from its arrival time to departure time), during which the client has to communicate with some station periodically. The need of 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 *station bandwidth* shared among its connected clients, by a *client bandwidth* required for each transmission, and by the client laxity governing how often it must connect to some stations.

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*. As to be further elaborated in the next paragraph, reallocation has been considered in a similar context in

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

the Windows Scheduling problem [23], where the cost of reallocation is proportional to the number of clients reallocated. While counting the number of clients reallocated ensures that we do not reallocate too much, this may not be a fair cost and it is typical in scheduling to consider reallocation (or migration) in terms of the sizes of the jobs instead of the number, e.g., [37]. 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 [2]. Reallocation usually involves handover from one station to another incurring a cost that is time related and also signal related [20].

We aim to reduce the number of active stations (a station is active if it has at least one client allocated to it to transmit) 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 quantify the trade-off between both performance metrics: number of active stations and reallocation cost. We call this problem *Station Assignment Problem with Reallocation* (SA).

Previous work. To the best of our knowledge, the closest work to the present paper is [23], where reallocation algorithms were presented for Windows Scheduling (WS). The WS problem [10,16,11,23] 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. WS has applications to various areas such as communication networks, supply chain, job scheduling, media on demand systems, etc. In [23], 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 showed 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 [10,16,11] was first studied without reallocation and the objective was mainly to minimize the number of channels. As pointed out in [29], the WS problem can be shown to be NP-hard by assembling results available in literature [9,11,28]. For the static case [10,11] where a client never departs, we can have online algorithm whose number of channels is only an additive of $O(\sqrt{H})$ from the optimal H, where H is the sum of reciprocal laxities of all clients [11]. For the dynamic case [16] where a client may depart, the maximum number of channels used over time by the online algorithm is at most a constant times that of the optimal [16]. This means that the comparison is against peak load which may occur at different time in the online algorithm and the optimal offline algorithm. In [23] and this work, we compare against current load.

As noted in [11], WS is closely related to the classical bin packing problem [18,19,17]. In addition to this, introducing bandwidth in our model gives

[2] As a first step we consider a reallocation cost in terms of laxity. It is of interest to consider bandwidth in the cost and we leave this future work.

another perspective in relation to bin packing. If all clients have very large laxity (such that the laxity constraint does not restrict them from being assigned to the same station) and the only concern becomes the bandwidth, then the problem of minimizing the number of stations becomes the same as minimizing the number of bins. Therefore, lower bounds on the approximation ratios of bin packing, i.e., 1.54037 for asymptotic approximation ratio [7] and 1.5 for absolute approximation ratio [21], apply to the station usage ratio of our problem when reallocation is not allowed.

SA and other assignment problems. SA generalizes several problems. It generalizes the WS problem that considered periodic transmission to capture bandwidth sharing. Different objectives are considered, in [10,16,11] the goal is to minimize the number of channels used while in [23] the goal is to minimize a combined cost of the number of reallocated clients and number of channels. We extend the later cost function such that the number of reallocated clients is weighted inversely by the client laxity. The problem in [25] considers clients with the same laxity and characterizes adversarial arrivals that admit feasible solutions. This makes the problem substantially different from ours as the periodic transmission can be handled as if the bandwidth is shared equally among the clients. We generalize the study to allow different laxities, and provide trade-off between reallocation cost and number of stations.

Our problem differs from existing scheduling problems despite sharing similarities. SA shares the idea of assigning tasks of different bandwidth to stations as the load balancing problem [5] of assigning jobs of different loads to machines, yet the load balancing problem does not consider periodic transmission, does not allow reallocation, and the objective is to minimize the maximum load. Interval coloring [1,22] concerns the number of machines used but not periodic tasks. Periodic tasks have been considered in real time scheduling [12] but the periodic appearance of the tasks is determined by the input, while in our problem the periodic appearance is determined by the algorithm to satisfy the laxity constraint. The SA problem is also related to online assignment problems such as b-matching [31], fractional matching [6], and adwords [24]. Among other details, the objective function is different.

We consider two orthogonal objectives which is common in scheduling context. E.g., in energy efficient scheduling problems, one would minimize the use of energy to provide acceptable quality of service. There are two typical approaches of optimization: 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 performance ratios as we do in this work, e.g., energy efficient throughput scheduling derives online algorithm that is t-throughput-competitive and e-energy-competitive [15]. Moreover, jointly targeting high bandwidth and low delay is also quite common in practice. For instance, in [30], the authors present a greedy scheduling policy for wireless networks aimed to achieve provably good performance in terms of both, throughput and delay. The model is different from ours (multiple radio channels, which can be viewed as a discrete version of our continuous-bandwidth

allocation, but only one base station and only one packet per client), but the two-dimensional optimization is the same.

Our objective function takes into account the assignment cost, which is often the optimization criteria in scheduling and network design problems. A good example is energy efficient speed-scaling scheduling where the speed of a processor is scalable to a higher speed consuming more energy while more productive. In [8] the objective function is the energy usage (modeled as an arbitrary power function) plus fractional weighted flow time. This is generalized in [26] to parallel machines where the objective function is energy plus an arbitrary assignment cost. Similar cost functions have been considered for the minimum-cost network-design problem, where packets have to be routed through a network of speed scalable routers, and the goal is to minimize the aggregate cost of assigning a packet to a link and the energy consumption of supporting the current load on the router [4]. On the other hand, scheduling in wireless networks with reallocation of resources has also been considered [13] yet reallocation is assumed to incur no cost.

Reallocation has been considered in the context of scheduling [14,36,3]. In [14], 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, but by different quantities, e.g., by a function of the number of jobs in the system [14], the size of the arriving job [36] 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 laxity and bandwidth requirements (arbitrary for the analysis, set to specific values for experimental evaluation), 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)-performance, 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 [23], 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 first found experimentally what is the classification function that seems to balance these goals (i.e. neither of the number of active stations nor the reallocation cost is the largest observed), and then we provided theoretical guarantees for all functions considered.
- 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 maintains both, station usage and reallocation cost, below the maximum observed. 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*** w_c, such that $0 < w_c \leq |\tau_c|$. I.e., c must transmit to some station in S at least one packet within each w_c consecutive time slots in τ_c [3]. 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. Note that a client is assigned to a station from its arrival time or when it is reallocated to this station until its departure time or when it is reallocated to another station (not only at the

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

instant time that it transmits). 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 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 [23] 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 scaling factor that generalizes this cost to different settings. For our simulations, we set $\rho = 1$, since ρ is also a multiplicative factor in our reallocation metric and, hence, does not provide additional information about the performance of our protocols in terms of reallocation.

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 possibly be reduced by 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,

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

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)***-performance*** if the following holds for any input.

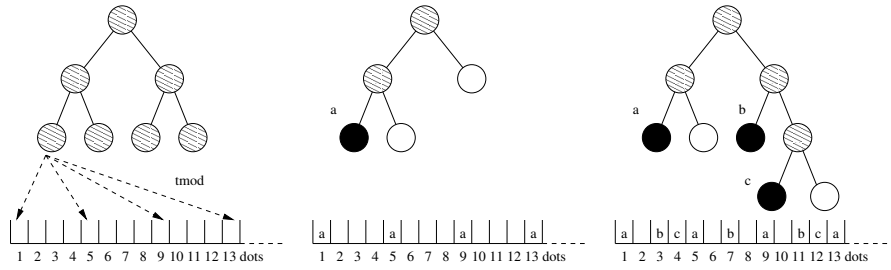
$$\max_t \frac{S(ALG, t)}{S(OPT, t)} \leq \alpha$$
$$\max_{t:\mathcal{R}(ALG,t)>0} \frac{\mathcal{R}(ALG, t)}{\mathcal{D}(ALG, t)} \leq \beta.$$

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 reallocation cost is only measured at the time when ALG reallocates some clients, i.e., when $\mathcal{R}(ALG, t) > 0$, because it is not meaningful to consider times in between reallocation events. The rationale of comparing $\mathcal{R}(ALG, t)$ against $\mathcal{D}(ALG, t)$ is as follows. When clients do not depart, the WS problem admits very good approximation performance even without reallocation (recall in the introduction that in such case there is online algorithm that differs from the optimal offline algorithm by only an additive term [11]). Therefore, we are motivated to study how algorithms may benefit from reallocation when there is departure by reusing the space released by the departure.

Notice that the above 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.

4 Algorithms

Broadcast Trees. A common theme in WS algorithms with *periodic* transmission schedules is to represent those schedules with *Broadcast Trees* [16, 10, 23]. Broadcast trees are a convenient representation because they allow to visualize easily how the laxities are combined. Consider for instance two clients a and b, both with laxity 2. Both clients may be assigned to the same station



(a) Mapping node - time-slot. (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 time slot $1, 5, 9$, etc. are reserved for it. (c) Open leaf (white node) corresponds to available slot.

alternating their transmissions. This assignment is represented by one binary tree where a and b hang from the root of a broadcast tree, modeling such station schedule. 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 of a broadcast tree 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.

Notice that once a client c is assigned to a node i, no other client can be assigned as a subtree of i, because all the time slots represented by i have been reserved for c. (Refer to Figure 1(b)) However, sibling clients are possible because they represent interleaving reservations (as in the example with a and b in the previous paragraph). Thus, if at any internal node only one child has a client assigned, an empty leaf is placed in the other child, making explicit the availability of the corresponding (periodic) time slot. Consequently, in broadcast trees all nodes have exactly zero or all possible children. Consider for instance the tree shown in Figure 1(c), where black nodes represent clients assigned and white nodes represent available slots. The transmission schedule in this example is depicted in the figure. Refer to [16, 10] for further details on broadcast trees.

WS algorithms. In [16] Chan et al. presented a WS algorithm that allocates clients with laxities that are powers of 2 preserving the following invariant. For each station, the broadcast tree modeling the station schedule has at most one available leaf at each depth. In order to preserve this invariant, when a client departs from a tree, the remaining clients in the same tree are rearranged for free. This invariant allows to upper bound the space available at each tree, but if reallocations among trees are possible, the same idea can be extended to all trees simultaneously. Indeed, that is the approach followed in the algorithm

Preemptive Reallocation (PR) [23], 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 of $H(C(t))$ for time slot t, where $H(C(t)) = \lceil \sum_{c \in C(t)} 1/w_c \rceil$, because the sum of all empty leaves (i.e., the sum of the inverse of laxities of all clients that could be placed in those leaves) is less than 1. Such guarantee is met re-establishing the invariant each time a client departs, possibly through reallocations among trees, at a constant cost per client reallocated between trees (within the tree are still free). It was shown experimentally that for various inputs the number of clients reallocated, amortized on the number of arrivals and departures, is constant [23]. However, we show in Lemma 1 that there are arrival/departure schedules for which the amortized cost in PR is unbounded. Furthermore, we show in Lemma 2 that if we simply modify PR to reallocate the sibling subtree of smaller weight (rather than the subtree with less clients) to restore the invariant, there are arrival schedules for which the reallocation-cost ratio is exponential for our cost function (Equation 1).

A WS algorithm with provable bounded reallocation cost guarantees was shown also in [23]. 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. At any time t, CR has an additive overhead on station usage of at most $1+\log(\min\{\max_{c \in C(t)} w_c, \lceil \lceil C(t) \rceil \rceil\} / \min_{c \in C(t)} w_c)$ [4], for laxities that are powers of 2. To attain constant amortized reallocation cost, clients are moved to/from the distinguished station only after the number of clients in the system has halved/doubled. However, for the reallocation cost function in Equation 1, that is a reallocation cost that depends on the resource requirements of the clients reallocated, CR has an arbitrarily bad reallocation cost ratio, as we show in Lemma 3.

Classified Preemptive Reallocation. The negative results in Lemmas 1, 2, and 3 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 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

[4] Throughout, log means $\log_2$ unless otherwise stated.

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. Refer to Algorithm 2 for further details on the classification.

Algorithm
  upon *arrival or departure of a client* c **do**
    if *arrival* **then** `allocate`$(c, \langle w_{low}, w_{high}\rangle)$
    else `consolidate`$(c, \langle w_{low}, w_{high}\rangle)$
  endupon
Procedure `allocate`$(c, \langle w_{low}, w_{high}\rangle)$
  for *each depth* $i = \lfloor \log w_c \rfloor - \lceil \log w_{low} \rceil$ *down to* 0 **do**
    for *each active station* s *of class* $\langle w_{low}, w_{high}, 1/\lfloor\lfloor B/b_c\rfloor\rfloor\rangle$ **do**
      if *there is a leaf* ℓ *available at depth* i *in the broadcast tree of* s **then**
        allocate to ℓ a new subtree with client c assigned at depth $\lfloor \log w_c \rfloor - i - \lceil \log w_{low} \rceil$ of the broadcast subtree
        return
      end
    end
  end
  activate a new station s in class $\langle w_{low}, w_{high}, 1/\lfloor\lfloor B/b_c\rfloor\rfloor\rangle$
  choose one of the leaves ℓ at depth 0 of the broadcast subtrees of s
  allocate to ℓ a new subtree with client c assigned at depth $\lfloor \log w_c \rfloor - \lceil \log w_{low} \rceil$ of the broadcast subtree
Procedure `consolidate`$(c, \langle w_{low}, w_{high}\rangle)$
  for *each depth* $i = \lfloor \log w_c \rfloor - \lceil \log w_{low} \rceil$ *down to* 1 **do**
    if *there are two active stations of class* $\langle w_{low}, w_{high}, 1/\lfloor\lfloor B/b_c\rfloor\rfloor\rangle$ *both with a leaf at depth* i *available* **then** reallocate sibling subtree of smaller weight
    else return
  end
  `// reallocations cleared a whole broadcast subtree`
  if *there are two active stations of class* $\langle w_{low}, w_{high}, 1/\lfloor\lfloor B/b_c\rfloor\rfloor\rangle$ *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

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.

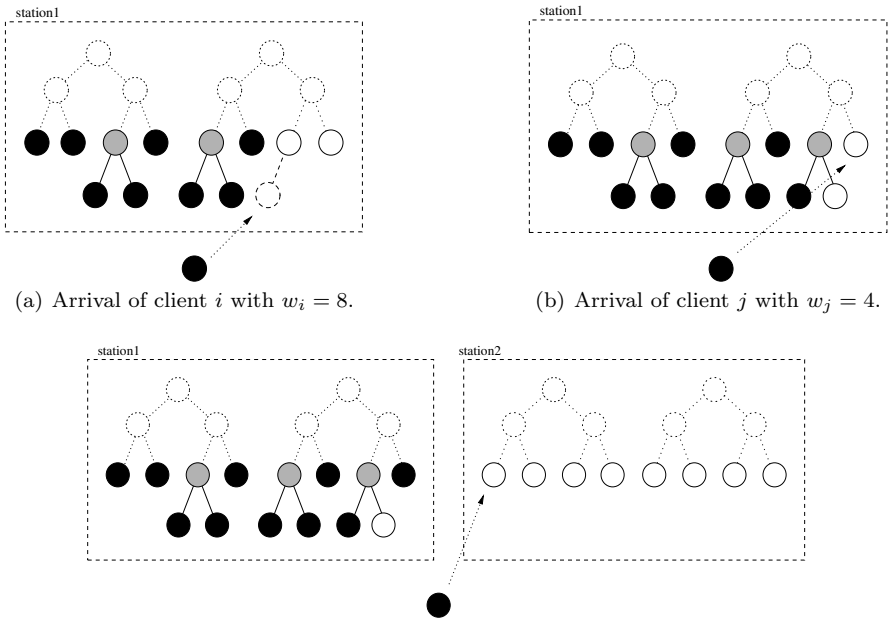
The mechanism to allocate an arriving client can be described as follows. Upon arrival, a client c is classified according to its laxity and bandwidth requirement. Specifically, c is assigned to a class for clients with bandwidth requirement $B/\lfloor\lfloor B/b_c\rfloor\rfloor$ and laxity in $[w_{low}, w_{high})$, for some w_{low} and w_{high} that depend on the classification chosen, as shown in Algorithm 2. 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,

Algorithm 2: Class Computation. $\lfloor\lfloor x\rfloor\rfloor$ is the largest power of 2 that is not larger than x. The parameter $factor$ indicates how the client classes are defined.

```
Function findLaxityClass(c,factor)
    if 1 ≤ ⌊⌊w_c⌋⌋ < 2 then return ⟨1, 2⟩
    if 2 ≤ ⌊⌊w_c⌋⌋ < 4 then return ⟨2, 4⟩
    w ← 4
    if factor = constant then
        while ⌊⌊w_c⌋⌋ ≥ 2w do                          // w_high = 2w_low
            w ← 2w
        end
        return ⟨w, 2w⟩
    end
    else if factor = logarithmic then
        while ⌊⌊w_c⌋⌋ ≥ w log_2 w do                    // w_high = w_low log_2 w_low
            w ← w log_2 w
        end
        return ⟨w, w log_2 w⟩
    end
    else // factor = linear
        while ⌊⌊w_c⌋⌋ ≥ w^2 do                          // w_high = w_low^2
            w ← w^2
        end
        return ⟨w, w^2⟩
    end
```

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



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

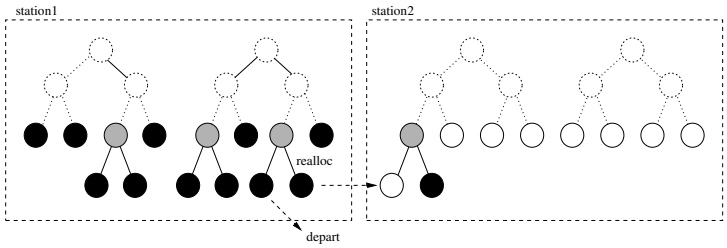
(b) Arrival of client j with $w_j = 4$.

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

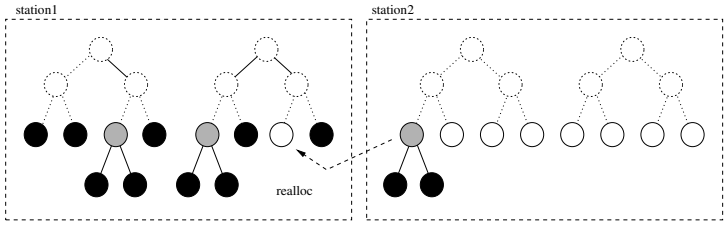
Fig. 2 Illustration of allocation mechanism. Class: laxities $[4, 16)$, bandwidth 1/2. Subtrees are depicted connected to a broadcast tree to reflect their location in the station schedule.

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 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 [16] 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 station usage. Indeed, for any given allocation/reallocation policy, an adversarial input can be shown so that either the laxity is stretched or the station usage is not optimal. Hence, in our model we assume that when a client



(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.

is reallocated the laxity may be stretched, folding the cost in the reallocation cost.

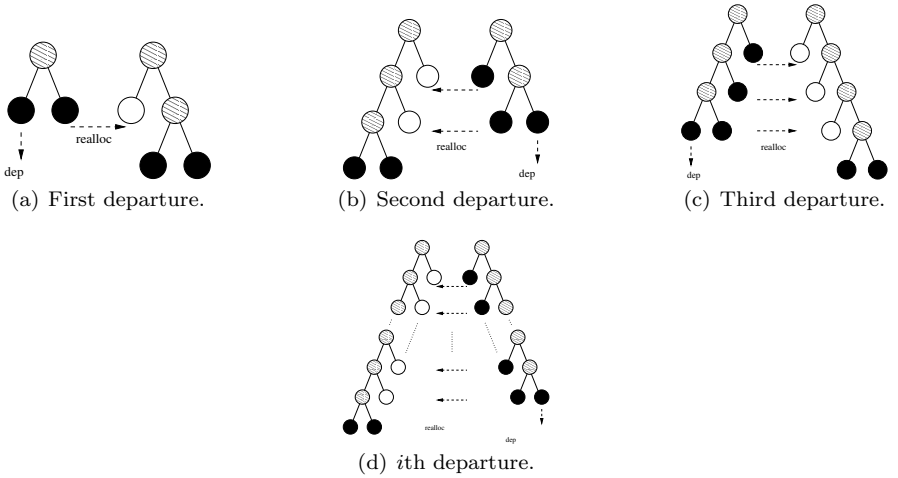
5 Analysis

We start with negative results in Lemmas 1, 2, and 3, which apply to WS, and to SA fixing $b_c = B$ for all clients. The proofs are all based on showing an adversarial client set for which the claim holds.

Lemma 1 *There exists a client arrival/departure schedule such that, in Preemptive Reallocation [23], the ratio of number of clients reallocated against the number of arrivals plus departures is unbounded.*

Proof Consider the following adversarial client arrival/departure schedule divided in rounds. In the first round, 2 clients of laxity 2 arrive. Then, for each round $r = 2, 3, 4, \ldots$, two clients of laxity 2^r arrive and, after these clients have been allocated, a client of laxity 2^{r-1} departs. Figure 4 shows the status of the forest right before each departure.

To compute the reallocation cost, consider any round $r \geq 2$. After the departure, two leaves are left available at depth r of the forest. For example, refer to Figure 4(c) depicting round 4. After the client at depth 4 departs, two leaves are left available at that depth. To restore the invariant, PR reallocates the sibling subtrees of the available leaves, so that they are assigned to the same



(a) First departure. (b) Second departure. (c) Third departure.

(d) ith departure.

Fig. 4 Illustration of Lemma 1.

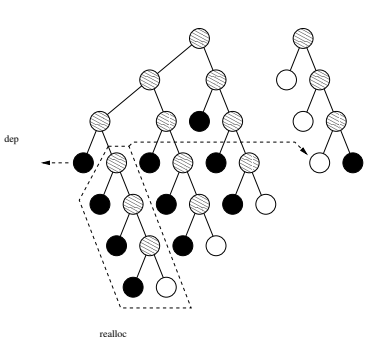
parent node. In doing so, now two leaves are left available at depth $r-1$ of the forest. Because PR reallocates the subtree with less clients assigned, similar reallocations are repeated transitively up through the forest until one of the trees is left empty. (Refer to Figure 4(c).) Then, the number of reallocated clients in round r is r, whereas the number of arriving or departing clients in each round is always 3. Given that the number of rounds is infinite, the overall reallocation cost ratio is unbounded. □

Lemma 2 *For Preemptive Reallocation [23], 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 $\max_{t:\mathcal{R}(t)>0} \mathcal{R}(t)/\mathcal{D}(t) \geq \rho(2^d-1)^2/2^d$.*

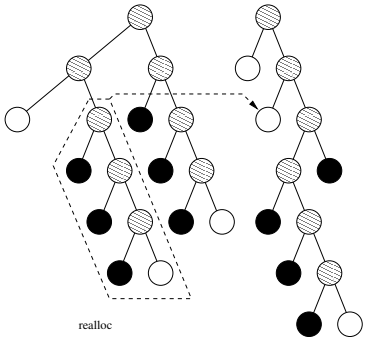
Proof Given $d > 0$, consider the following adversarial client arrival/departure schedule divided in phases. First a client of laxity 2^d arrives. After this client was assigned, a sequence of clients arrive one by one so that a new client arrives only after the previous client was assigned. The sequence of laxities of those clients is the following.

$$
\begin{aligned}
&2^{d+1}, 2^{d+2}, \ldots, 2^{2d-1}, 2^{2d},\\
&2^{d}, 2^{d+1}, \ldots, 2^{2d-2}, 2^{2d-1},\\
&2^{d-1}, 2^{d}, 2^{d+1}, \ldots, 2^{2d-2},\\
&\ldots\\
&2^2, 2^3, \ldots, 2^{d-1}, 2^{d}, 2^{d+1}.
\end{aligned}
$$

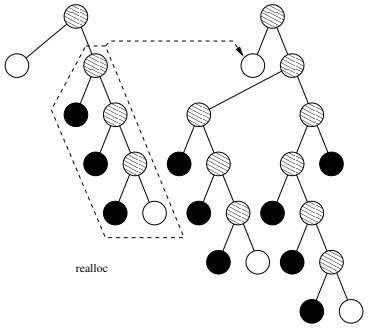
Then, another client of laxity 2^d arrives. Figure 5(a) illustrates the assignment of clients by PR for $d = 3$. Finally, after all clients have been assigned, the client that arrived first departs. No other client arrives or departs afterwards. The client departure leaves two leaves available at depth d. Then, the sibling subtree of smaller weight is reallocated (refer to Figure 5(a)). In turn, this reallocation leaves two leaves available at depth $d - 1$, which triggers the reallocation of the sibling subtree of smaller weight (refer to Figure 5(b)). These transitive reallocations continue upwards the tree depth-by-depth up to depth 2 (refer to Figure 5(c)), when the last reallocation leaves one of the trees empty (refer to Figure 5(d)).



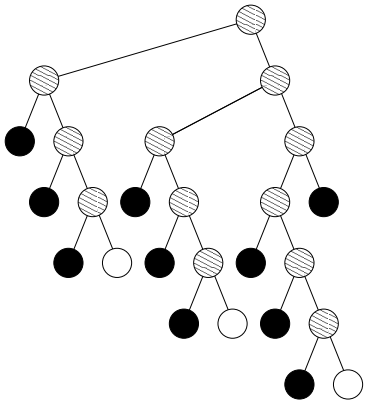
(a) Before first reallocation.



(b) Before second reallocation.



(c) Before third reallocation.



(d) Final assignment.

Fig. 5 Illustration of Lemma 2.

Then, at the time slot t when all clients have been reallocated, we have

$$\begin{aligned}\frac{\mathcal{R}(t)}{\mathcal{D}(t)} &= \frac{\rho \sum_{c \in R(t)} 1/w_c}{1/2^d} \\ &= \frac{\rho \sum_{i=2}^{d+1} \sum_{j=0}^{d-1} 1/2^{i+j}}{1/2^d} \\ &= \frac{\rho(2^d-1)^2}{2^d}.\end{aligned}$$

□

Lemma 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 [23], we have* $\max_{t:\mathcal{R}(t)>0} \mathcal{R}(t)/\mathcal{D}(t) \geq \frac{\rho/4}{7 \cdot 2^x} w$.

Proof We use the terminology "channel" in [23] in this proof. The thresholds to reallocate from/to the big channel in CR are the following [23]. For any time t, if a client c allocated to the big channel has laxity $w_c < \lceil \lceil |C(t)| \rceil \rceil$, c is reallocated to other channel according to w_c, call it w_c-channel. On the other hand, if at any time t a client c that is *not* allocated to the big channel has laxity $w_c > 2\lceil \lceil |C(t)| \rceil \rceil$, then c is reallocated to the big channel.

Consider an adversarial scenario where the system has 2^x clients with laxity 2^{x+2} and $7 \cdot 2^x$ clients with laxity w, where w is a power of 2 such that $w \geq 2^{x+5}$. (The order in which these clients have arrived is irrelevant.) Because the total number of clients is 2^{x+3}, the clients with laxity $w \geq 2^{x+5} > 2 \cdot 2^{x+3}$ are allocated to the big channel, whereas the clients with laxity $2^{x+2} < 2^{x+3}$ are allocated to a (2^{x+2})-channel. After these clients have been allocated, adversarially, all the clients with laxity w depart. Because the new number of clients in the system is now 2^x, the remaining clients, all with laxity $2^{x+2} > 2 \cdot 2^x$ have to be reallocated to the big channel. Then, at time t after reallocation, the following holds.

$$\begin{aligned}\frac{\mathcal{R}(t)}{\mathcal{D}(t)} &= \frac{\rho \sum_{c \in R(t)} 1/w_c}{\sum_{c \in D(t)} 1/w_c} \\ &= \frac{\rho/4}{7 \cdot 2^x} w.\end{aligned}$$

□

The above lemmas show that the application of previous WS reallocation algorithms to SA is not feasible. The following theorem gives guarantees on station usage and reallocation cost for CPR. The proof starts by analyzing CPS to show 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.

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

Proof We start by showing that the invariant in Algorithm 1 is preserved. Recall that the invariant is the following. At any time slot t and for any class of clients $\langle w_{low}, w_{high}, x\rangle$, there is at most one leaf available at any depth larger than $\lceil \log w_{low} \rceil$ of the forest. There might be more than one leaf available at depth $\lceil \log w_{low} \rceil$ (an empty broadcast subtree), but only in one station in the class.

The arrival of clients does not change the invariant, but the departure of a client c at a given depth $i > \lceil \log w_{low} \rceil$ may change the number of leaves available at depth i. If there was no leaf available at depth i before the departure, the number of available leaves at depth i is at most one after departure and the invariant is preserved. If, on the other hand, there was a leaf ℓ available at depth i, either the sibling of c or the sibling of ℓ will be reallocated in Line 1 of the algorithm. This reallocation leaves two sibling leaves available at depth i, which combined yield a leaf available at depth $i - 1$. The same argument applies transitively upwards the tree. If the invariant is re-established before reaching depth $\lceil \log w_{low} \rceil$, we are done. If on the other hand a broadcast subtree is emptied, the invariant is re-established (if necessary) reallocating a whole broadcast subtree in Line 1. Notice that reallocating one subtree is enough to re-establish the invariant, since before the departure there was (at most) one station with empty subtrees, and the departure (possibly followed by reallocations) may empty only one subtree.

To bound α, we observe that the invariant above guarantees that there is at most one station per class with empty broadcast subtrees. For the stations with non-empty subtrees, aggregating the at most one available leaf at each depth larger than 0 (and smaller than $\lfloor \log w_{high} \rfloor$) of each forest, we have an additional available space of at most one station, throughout all classes. So, the overhead in station usage is the number classes plus one. Additionally, we have to take into account that clients are scheduled to transmission periods that are powers of 2, and with a bandwidth that is a power of 2 fraction of the capacity B, which introduces a multiplicative factor in station usage of at most 4. Thus, we have

$$\max_t \frac{S(ALG, t)}{S(OPT, t)} \leq \max_t \frac{4(1 + \Gamma(ALG, t) + S(OPT, t))}{S(OPT, t)}$$

To bound β, we compute the maximum weight of clients reallocated upon a departure. We notice that, for any class of clients $\langle w_{low}, w_{high}, x\rangle$, in the worst

case a departure at depth $\lfloor \log w_{high} \rfloor$ triggers transitive reallocations upwards up to depth $\lceil \log w_{low} \rceil - 1$ in the forest, followed by a reallocation of a whole broadcast subtree of weight at most $1/\lceil\lceil w_{low} \rceil\rceil$. The aggregated weight of all those reallocations is then $1/\lceil\lceil w_{low} \rceil\rceil + 1/(2\lceil\lceil w_{low} \rceil\rceil) + 1/(4\lceil\lceil w_{low} \rceil\rceil) + \cdots + 1/\lfloor\lfloor w_{high} \rfloor\rfloor = 2/\lceil\lceil w_{low} \rceil\rceil - 1/\lfloor\lfloor w_{high} \rfloor\rfloor$. Replacing, we obtain

$$\begin{aligned}\max_{t:\mathcal{R}(ALG,t)>0} \frac{\mathcal{R}(ALG,t)}{\mathcal{D}(ALG,t)} &\leq \max_{w_{low},w_{high}} \frac{\rho(2/\lceil\lceil w_{low} \rceil\rceil - 1/\lfloor\lfloor w_{high} \rfloor\rfloor)}{1/\lfloor\lfloor w_{high} \rfloor\rfloor} \\ &\leq \max_t \rho(2\lfloor\lfloor w_{high_{\max}}(t) \rfloor\rfloor/\lceil\lceil w_{low_{\max}}(t) \rceil\rceil - 1).\end{aligned}$$

□

Instantiating Theorem 1 in the classification factors of Algorithm 2, we obtain bounds for all three algorithms, shown in Corollary 1.

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

1. ***Constant factor.*** *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*

$$\alpha = 4\left(1 + \frac{1 + \left(1 + \log \frac{\lceil\lceil B/b_{\min}(t)\rceil\rceil}{\lceil\lceil B/b_{\max}(t)\rceil\rceil}\right)\left(1 + \log \frac{\lfloor\lfloor w_{\max}(t)\rfloor\rfloor}{\lfloor\lfloor w_{\min}(t)\rfloor\rfloor}\right)}{H(C(t))}\right)$$

$$\beta = 3\rho.$$

2. ***Logarithm factor.*** *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*

$$\alpha = 4\left(1 + \frac{1 + \left(1 + \log \frac{\lceil\lceil B/b_{\min}(t)\rceil\rceil}{\lceil\lceil B/b_{\max}(t)\rceil\rceil}\right)\left(1 + \frac{\log\lfloor\lfloor w_{\max}(t)\rfloor\rfloor}{\log\log \max\{4, \lfloor\lfloor w_{\min}(t)\rfloor\rfloor\}}\right)}{H(C(t))}\right)$$

$$\beta = \rho(2 \log w_{\max}(t) - 1).$$

3. ***Linear factor.*** *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*

$$\alpha = 4\left(1 + \frac{1 + \left(1 + \log \frac{\lceil\lceil B/b_{\min}(t)\rceil\rceil}{\lceil\lceil B/b_{\max}(t)\rceil\rceil}\right)\left(1 + \log \frac{\log \max\{2, \lfloor\lfloor w_{\max}(t)\rfloor\rfloor\}}{\log \max\{2, \lfloor\lfloor w_{\min}(t)\rfloor\rfloor\}}\right)}{H(C(t))}\right)$$

$$\beta = \rho\left(2\sqrt{w_{\max}(t)} - 1\right).$$

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 $b_{\min}(t) = \min_{c \in C(t)} b_c$.

Proof Using that $S(OPT, t) \geq H(C(t))$, and bounding the values of $\max_t \Gamma(ALG, t)$ and $\max_t \lfloor\lfloor w_{high_{\max}}(t) \rfloor\rfloor/\lceil\lceil w_{low_{\max}}(t) \rceil\rceil$ in Theorem 1, the claim follows. □

We note that the choice of classification factor gives a trade-off on the performance on station usage and reallocation cost, i.e., the station usage improves as we move from constant to logarithm to linear factor while the reallocation cost improves as we move from linear to logarithm to constant factor. We comment that the logarithm classification gives good performance for both measurement.

To provide intuition, we instantiate Corollary 1 on a setting where all laxities are powers of 2 and all bandwidth requirements are the full capacity of a station, as follows.

Corollary 2 *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)-performance 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 results of our experimental simulations of the CPR algorithm. We highlight here that the classification factor (logarithmic) that maintains simultaneously station usage and reallocation cost below the maximum observed was found through experimentation with various functions. For the specific cases presented (constant, logarithmic, and linear factors) we have focused on a scenario where $\forall c \in C, b_c = 1/2^i$, and $w_c = 2^j$, where $i, j \geq 0$ and B was normalized to 1. For all the evaluations the reallocation cost of each client c has been set to the inverse of its laxity $1/w_c$. That is, $\rho = 1$, since the scaling factor ρ is also a multiplicative factor in our

reallocation metric and, hence, does not provide additional information about the performance of our protocols in terms of reallocation.

Our theoretical bounds on performance apply to worst-case scenarios. Hence, the purpose of these simulations is to complement those bounds evaluating how much better (if anything) our protocol behaves in practice for average cases. Given that the main feature of the protocol is to allocate (and reallocate) "efficiently", we aim to stress such feature considering inputs that entail extremal cases of arrivals. That is, smooth distributions of arrivals as well as batched arrivals. The set of inputs chosen are representative of those cases. Moreover, they are also the customary choices in experimental evaluation for other problems such as job scheduling, packet routing, etc. Other reallocation algorithms were not simulated since, to the best of our knowledge, this is the first time that restrictions on laxity and bandwidth under a reallocation cost proportional to resources requested have been considered.

We have produced various sets of clients (recall that each client is characterized by a time of arrival, a time of departure, a bandwidth, and a laxity). The laxity of each client was chosen independently at random from $\{1, 2, 4, \ldots, w_{\max}\}$, for each $w_{\max} = 1024, 4096$, and 16384. We evaluated three distributions over that range: uniform, biased towards small laxities, and biased towards large laxities. Biased means probability 0.7 of choosing from one half of the range (lower or higher), and then uniform probability within the half chosen. The bandwidth of each client c was chosen at random as $b_c = 1/2^i$ with probability $1/2^i$ for each $i = 1, 2, \ldots$ For each of $n = 4000, 8000$, and 16000 clients, time was discretized in $2n$ slots.

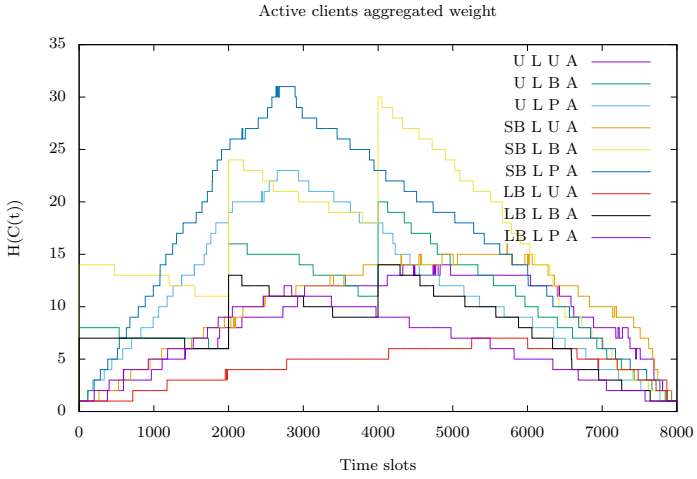


Fig. 6 Cumulative inverse laxity ($H(C(t))$) vs. time for $n = 4000$ and $w_{\max} = 1024$. Key: L: laxity, A: arrival, U: uniform, B: batched, P: Poisson, SB: small-biased, LB: large-biased.

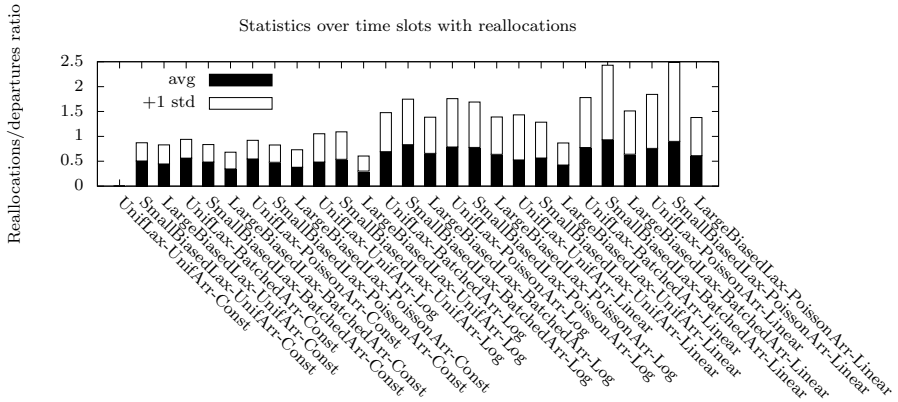


Fig. 7 Reallocation/Departure ratio statistics for different classification factors, laxity distributions, and arrival distributions, for $n = 4000$ and $w_{\max} = 1024$.

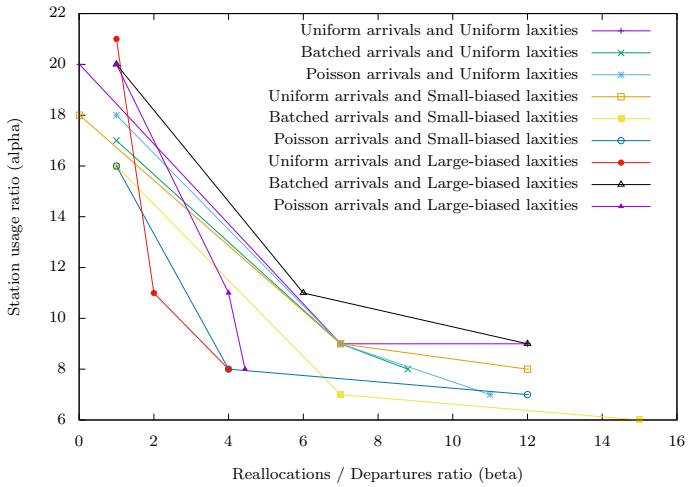


Fig. 8 Worst case α vs. β. $n = 4000$, $w_{\max} = 1024$, $w_{\min} = 1$, $\rho = 1$. Each colored line corresponds to the 3 data points obtained for each input. The leftmost point corresponds to the Constant factor algorithm, the middle point corresponds to the Logarithmic factor, and the rightmost point corresponds to the Linear factor.

The arrival time of each client was chosen: (a) uniformly at random within the interval $[1, 2n]$; (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 mean rate $\lambda = 0.7$. The choice of a Poisson process intends to model another case where the arrival schedule does not include bursts, whereas the value chosen for λ intends to model an arrival schedule that is somewhat dense (0.7 expected arrivals per unit of time

until all n clients have arrived). 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. The inputs for $n = 4000$ and $w_{\max} = 1024$ are illustrated in Figure 6 showing the $H(C(t))$ function, which is a lower bound on the optimal number of stations needed.

With respect to the protocol, three different classification factors: constant, logarithmic, and linear, were used, as detailed in Algorithm 2. We implemented the protocol and input generator in Java 8. The simulations were carried out on one of the Linux servers at Pace University. The specifications are Intel®Xeon®CPU X5450 @ 3.00GHz, 2GB RAM, 150GB HD, running Debian 8 x64.

For each of the 243 scenarios that arise from the combination of the above variants (3 $w_{\max}$, 3 laxity distributions, 3 arrival distributions, 3 numbers of clients, and 3 protocols), we evaluated experimentally the (α, β)-performance of CPR. Our simulations showed that the performance in practical settings is indeed as expected or better than the theoretical bounds (as in Corollary 1). The discussion and plots that follow, refer to $n = 4000$ and $w_{\max} = 1024$, but similar results were obtained for the other cases. The source code, the input data, and the raw output data are publicly available in [27].

It can be seen in Figures 9, 10, and 11 that the reallocation vs. departures weight ratio (bounded by β) is frequently at most 1. For constant factor classification on uniform arrival distribution and uniform laxity no client was ever reallocated. Hence, this case is not plotted. Also, the ratio is defined on reallocation events. Hence, no data points are shown in time slots without reallocations.

To quantify the latter observations, we compute statistics of the reallocation vs. departures weight ratio, over time slots where some client has been reallocated. The results are shown in Figure 7. It can be seen that in all cases the average plus one standard deviation is below 2.5. For comparison, we compute the bounds β proved in Corollary 1. Recall that the sample space for $w_{\max}$ in the simulations was $[1, 1024]$. Nevertheless, being pessimistic and replacing $w_{\max} = 8$, and the value $\rho = 1$ used in our simulations, we have that the theoretical upper bound is $\beta \geq 3$ for all classification factors. For larger values of $w_{\max}$ the gap between our observations and the theoretical bound is even larger, showing that on realistic inputs our protocol behaves much better than the worst-case theoretical bounds.

With respect to station usage, Figure 12 shows that 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 α) is most of the time below 4, and frequently below 2. We make this observation more precise by computing the percentage of time slots when the station usage ratio against $H(C(t))$ is below 4 for each combination of classification factor and arrival distribution. The results are shown in Table 1.

Should the reallocation ratio be minimized, the constant factor classification achieves better performance at a higher station usage. On the other hand, if station usage must be kept low, the linear factor classification performs bet-

Laxity distribution	Arrival distribution	Factor	percentage
Unif	Unif	Const	69.0875
SmallBiased	Unif	Const	76.5625
LargeBiased	Unif	Const	9.575
Unif	Batched	Const	83.4
SmallBiased	Batched	Const	89.225
LargeBiased	Batched	Const	79.3125
Unif	Poisson	Const	73.3875
SmallBiased	Poisson	Const	80.9
LargeBiased	Poisson	Const	41.475
Unif	Unif	Log	90.1875
SmallBiased	Unif	Log	91.9375
LargeBiased	Unif	Log	75.925
Unif	Batched	Log	95.0
SmallBiased	Batched	Log	95.3375
LargeBiased	Batched	Log	88.825
Unif	Poisson	Log	90.0625
SmallBiased	Poisson	Log	94.05
LargeBiased	Poisson	Log	78.4375
Unif	Unif	Linear	91.05
SmallBiased	Unif	Linear	91.9375
LargeBiased	Unif	Linear	86.725
Unif	Batched	Linear	96.0
SmallBiased	Batched	Linear	95.925
LargeBiased	Batched	Linear	90.875
Unif	Poisson	Linear	92.5875
SmallBiased	Poisson	Linear	94.7375
LargeBiased	Poisson	Linear	83.275

Table 1 Percentage of time slots when the station usage ratio is below 4, for each classification factor, laxity distribution, and arrival distribution, for $n = 4000$ and $w_{\max} = 1024$.

ter incurring in higher reallocation cost. The logarithmic factor balances both costs. Figure 8 illustrates these trade offs. In comparison with the bounds proved in Corollary 1, for the scenarios simulated CPR behaves better than expected. As we see in the figure, these trade-offs appear in all input distributions, although in some the impact is milder (e.g. large-biased laxities with uniform or Poisson arrivals).

The inputs chosen for our evaluation are intuitively representative of a variety of likely cases. Namely, bursts and smooth arrivals, more/even/less demanding clients, etc. Should a comparison among factors regardless of distributions be needed (e.g., if the distribution is unknown, but the extremal values of bandwidths, laxities, and $H(C(t))$ are known) the worst-case guarantees in the analysis must be used.

7 Conclusions and Future Work

In this paper, we study a dynamic allocation problem SA and associated reallocation algorithms assuming that clients have laxity and bandwidth requirements. We characterize these algorithms by defining the notion (α, β)-performance as combination of the competitive ratio on station usage (α) and

the cost of reallocations (β). We show that previous protocols that work well for unit cost per client reallocation do not work well when the cost is more general. We then present a new protocol called Classified Preemptive Reallocation and prove bounds on both of our performance metrics. We also present experimental simulation results on average cases supplementing our theoretical analysis on worst case.

There are a few future directions. To further understand the performance of algorithms, it is desirable to derive lower bounds on the performance ratio of a general algorithms. In this paper we assume that each station has the same capacity. An obvious generalization is to consider stations having different capacities. In addition, we may extend cost model to introduce a weight to each client and the reallocation cost is then calculated as a weighted cost. In terms of the setting, we aim to quantify the resources required to complete all requests from clients. A direction is to consider limited resources and striking a balance between completing more clients and not violating the resource limitation.

Acknowledgements The authors thank the support from a Visiting Fellowship and the initiative Networks Sciences & Technologies (NeST) by School of EEE & CS, University of Liverpool, as well as Pace University NYFC SRC Award and Kenan Fund Award.

References

1. Adamy, U., Erlebach, T.: Online coloring of intervals with bandwidth. In: Proceedings of the 1st International Workshop on Approximation and Online Algorithms, *Lecture Notes in Computer Science*, vol. 2909, pp. 1–12. Springer (2003)
2. Albers, S., Fujiwara, H.: Energy-efficient algorithms for flow time minimization. ACM Transactions on Algorithms **3**(4), 49 (2007)
3. Albers, S., Hellwig, M.: On the value of job migration in online makespan minimization. In: Proceedings of the 20th Annual European Symposium on Algorithms, *Lecture Notes in Computer Science*, vol. 7501, pp. 84–95. Springer (2012)
4. Andrews, M., Antonakopoulos, S., Zhang, L.: Minimum-cost network design with (dis)economies of scale. In: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science, pp. 585–592. IEEE Computer Society (2010)
5. Azar, Y.: On-line load balancing. In: Proceedings of Developments from a June 1996 Seminar on Online Algorithms: The State of the Art, pp. 178–195. Springer-Verlag (1996)
6. Azar, Y., Litichevskey, A.: Maximizing throughput in multi-queue switches. Algorithmica **45**, 69–90 (2006)
7. Balogh, J., Békési, J., Galambos, G.: New lower bounds for certain classes of bin packing algorithms. In: Proceedings of the 8th International Workshop on Approximation and Online Algorithms (WAOA), pp. 25–36 (2010)
8. Bansal, N., Chan, H.L., Pruhs, K.: Speed scaling with an arbitrary power function. ACM Transactions on Algorithms **9**(2), 18:1–18:14 (2013)
9. Bar-Noy, A., Bhatia, R., Naor, J., Schieber, B.: Minimizing service and operation costs of periodic scheduling. In: Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 11–20 (1998)
10. Bar-Noy, A., Ladner, R.E.: Windows scheduling problems for broadcast systems. SIAM Journal on Computing **32**(4), 1091–1113 (2003)
11. Bar-Noy, A., Ladner, R.E., Tamir, T.: Windows scheduling as a restricted version of bin packing. ACM Transactions on Algorithms **3**(3), 28 (2007)

12. Baruah, S., Goossens, J.: Scheduling real-time tasks: Algorithms and complexity. In: J. Leung (ed.) Handbook of Scheduling: Algorithms, Models and Performance Analysis, pp. 15–1–15–41. CRC Press (2004)
13. Becchetti, L., Leonardi, S., Marchetti-Spaccamela, A., Vitaletti, A., Diggavi, S., Muthukrishnan, S., Nandagopal, T.: Parallel scheduling problems in next generation wireless networks. Networks **45**(1), 9–22 (2005)
14. Bender, M.A., Farach-Colton, M., Fekete, S.P., Fineman, J.T., Gilbert, S.: Reallocation problems in scheduling. In: Proceedings of the 25th ACM Symposium on Parallelism in Algorithms and Architectures, pp. 271–279. ACM (2013)
15. Chan, H., Chan, J.W., Lam, T.W., Lee, L., Mak, K., Wong, P.W.H.: Optimizing throughput and energy in online deadline scheduling. ACM Transactions on Algorithms **6**(1), 1–22 (2009)
16. Chan, W.T., Wong, P.: On-line windows scheduling of temporary items. In: Proceedings of the 15th International Symposium on Algorithms and Computation, *Lecture Notes in Computer Science*, vol. 3341, pp. 259–270. Springer (2004)
17. Coffman Jr, E.G., Csirik, J., Galambos, G., Martello, S., Vigo, D.: Bin packing approximation algorithms: survey and classification. In: Handbook of Combinatorial Optimization, pp. 455–531. Springer (2013)
18. Coffman, Jr., E.G., Galambos, G., Martello, S., Vigo, D.: Bin packing approximation algorithms: Combinatorial analysis. In: D.Z. Du, P.M. Pardalos (eds.) Handbook of Combinatorial Optimization, pp. 151–207. Kluwer Academic Publishers (1998)
19. Coffman, Jr., E.G., Garey, M.R., Johnson, D.S.: Bin packing approximation algorithms: A survey. In: D.S. Hochbaum (ed.) Approximation Algorithms for NP-Hard Problems, pp. 46–93. PWS (1996)
20. Cominardi, L., Giust, F., Bernardos, C.J., de la Oliva, A.: Distributed mobility management solutions for next mobile network architectures. Computer Networks **121**, 124–136 (2017)
21. Epstein, L.: Bin packing with rejection revisited. Algorithmica **56**(4), 505–528 (2010)
22. Epstein, L., Erlebach, T., Levin, A.: Variable sized online interval coloring with bandwidth. Algorithmica **53**(3), 385–401 (2009)
23. Farach-Colton, M., Leal, K., Mosteiro, M.A., Thraves, C.: Dynamic windows scheduling with reallocation. In: Proceedings of the 13th International Symposium on Experimental Algorithms, *Lecture Notes in Computer Science*, vol. 8504, pp. 99–110. Springer (2014)
24. Feldman, J., Mehta, A., Mirrokni, V., Muthukrishnan, S.: Online stochastic matching: Beating 1-1/e. In: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science, pp. 117 –126. IEEE Computer Society (2009)
25. Fernández Anta, A., Kowalski, D.R., Mosteiro, M.A., Wong, P.W.H.: Station assignment with applications to sensing. In: Proceedings of the 9th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics, *Lecture Notes in Computer Science*, vol. 8243, pp. 155–169. Springer (2013)
26. Gupta, A., Krishnaswamy, R., Pruhs, K.: Online primal-dual for non-linear optimization with applications to speed scaling. In: Proceedings of the 10th Workshop on Approximation and Online Algorithms, *Lecture Notes in Computer Science*, vol. 7846, pp. 173–186. Springer (2012)
27. Halper, A., Mosteiro, M.A., Rossikova, Y., Wong, P.W.H.: Station assignment with reallocation simulator code and data. `http://csis.pace.edu/~mmosteiro/pub/sourceBSreallocJournal/` (2017)
28. Holte, R., Mok, A., Rosier, L., Tulchinsky, I., Varvel, D.: The pinwheel: a real-time scheduling problem. In: Proceedings of the 22nd Annual Hawaii International Conference on System Sciences, vol. II, Software Track, pp. 693–702 (1989)
29. Jacobs, T., Longo, S.: A new perspective on the windows scheduling problem. CoRR **abs/1410.7237** (2014). URL `http://arxiv.org/abs/1410.7237`
30. Ji, B., Gupta, G.R., Sharma, M., Lin, X., Shroff, N.B.: Achieving optimal throughput and near-optimal asymptotic delay performance in multi-channel wireless networks with low complexity: A practical greedy scheduling policy. IEEE/ACM Transactions on Networking **23**(3), 880–893 (2015)
31. Kalyanasundaram, B., Pruhs, K.: An optimal deterministic algorithm for online b-matching. Theoretical Computer Science **233**(1-2), 319–325 (2000)

32. Kanjo, E., Benford, S., Paxton, M., Chamberlain, A., Fraser, D.S., Woodgate, D., Crellin, D., Woolard, A.: Mobgeosen: facilitating personal geosensor data collection and visualization using mobile phones. Personal and Ubiquitous Computing **12**(8), 599–607 (2008)
33. Khan, W.Z., Xiang, Y., Aalsalem, M.Y., Arshad, Q.: Mobile phone sensing systems: A survey. IEEE Communications Surveys and Tutorials **15**(1), 402–427 (2013)
34. Mosteiro, M.A., Rossikova, Y., Wong, P.W.: Station assignment with reallocation. In: Proceedings of the 14th International Symposium on Experimental Algorithms, Lecture Notes in Computer Science, pp. 151–164. Springer (2015)
35. Restuccia, F., Das, S.K., Payton, J.: Incentive mechanisms for participatory sensing: Survey and research challenges. TOSN **12**(2), 13:1–13:40 (2016)
36. Sanders, P., Sivadasan, N., Skutella, M.: Online scheduling with bounded migration. In: Proceedings of the 31st International Colloquium on Automata, Languages and Programming, *Lecture Notes in Computer Science*, vol. 3142, pp. 1111–1122. Springer (2004)
37. Sanders, P., Sivadasan, N., Skutella, M.: Online scheduling with bounded migration. Math. Oper. Res. **34**(2), 481–498 (2009)
38. Sha, K., Zhan, G., Shi, W., Lumley, M., Wiholm, C., Arnetz, B.: Spa: A smart phone assisted chronic illness self-management system with participatory sensing. In: Proceedings of the 2Nd International Workshop on Systems and Networking Support for Health Care and Assisted Living Environments, HealthNet '08, pp. 5:1–5:3. ACM, New York, NY, USA (2008)

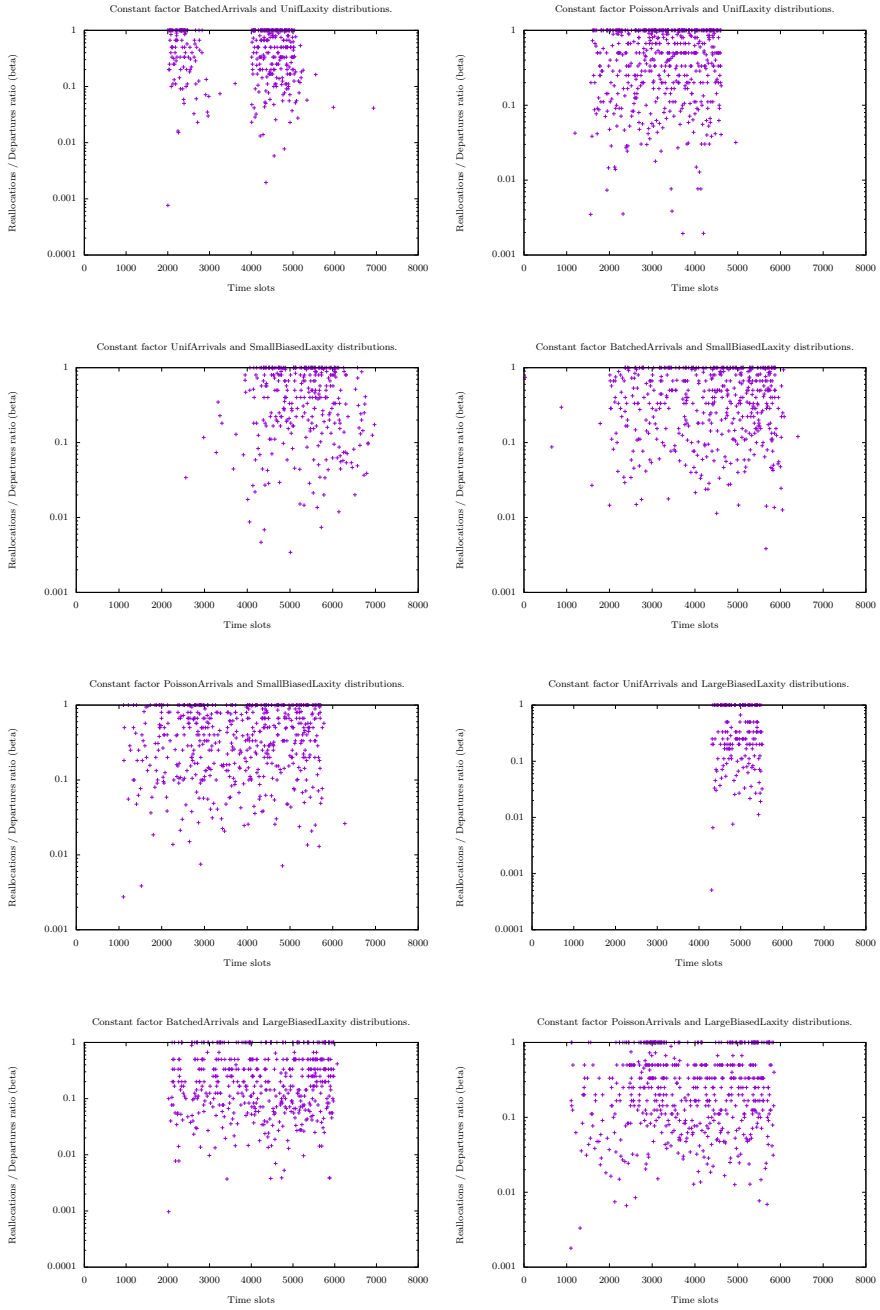


Fig. 9 Reallocation/Departure ratio (β) vs. time for constant classification factor, $n = 4000$ and $w_{\max} = 1024$.

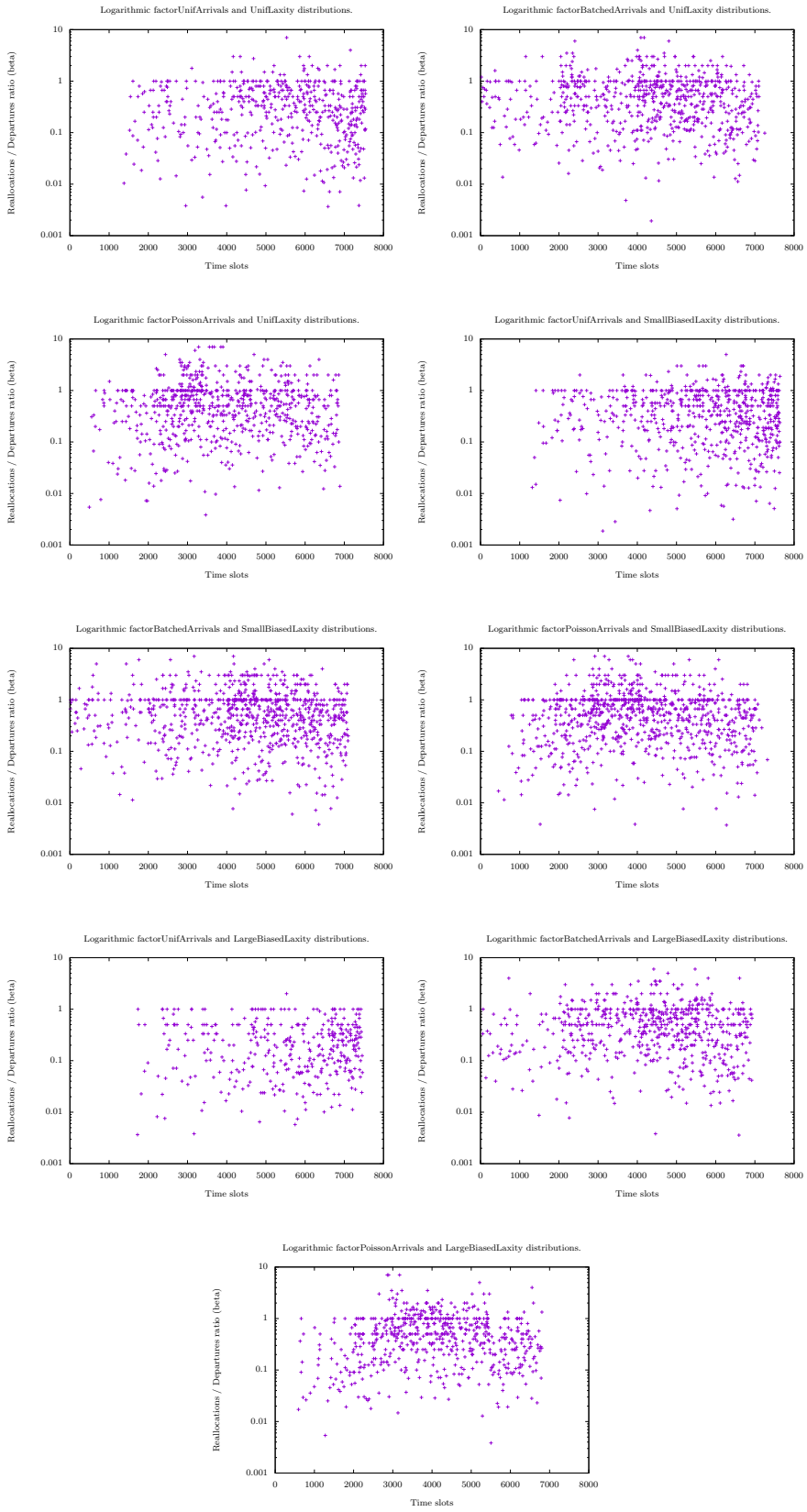


Fig. 10 Reallocation/Departure ratio (β) vs. time for logarithmic classification factor, $n = 4000$ and $w_{\max} = 1024$.

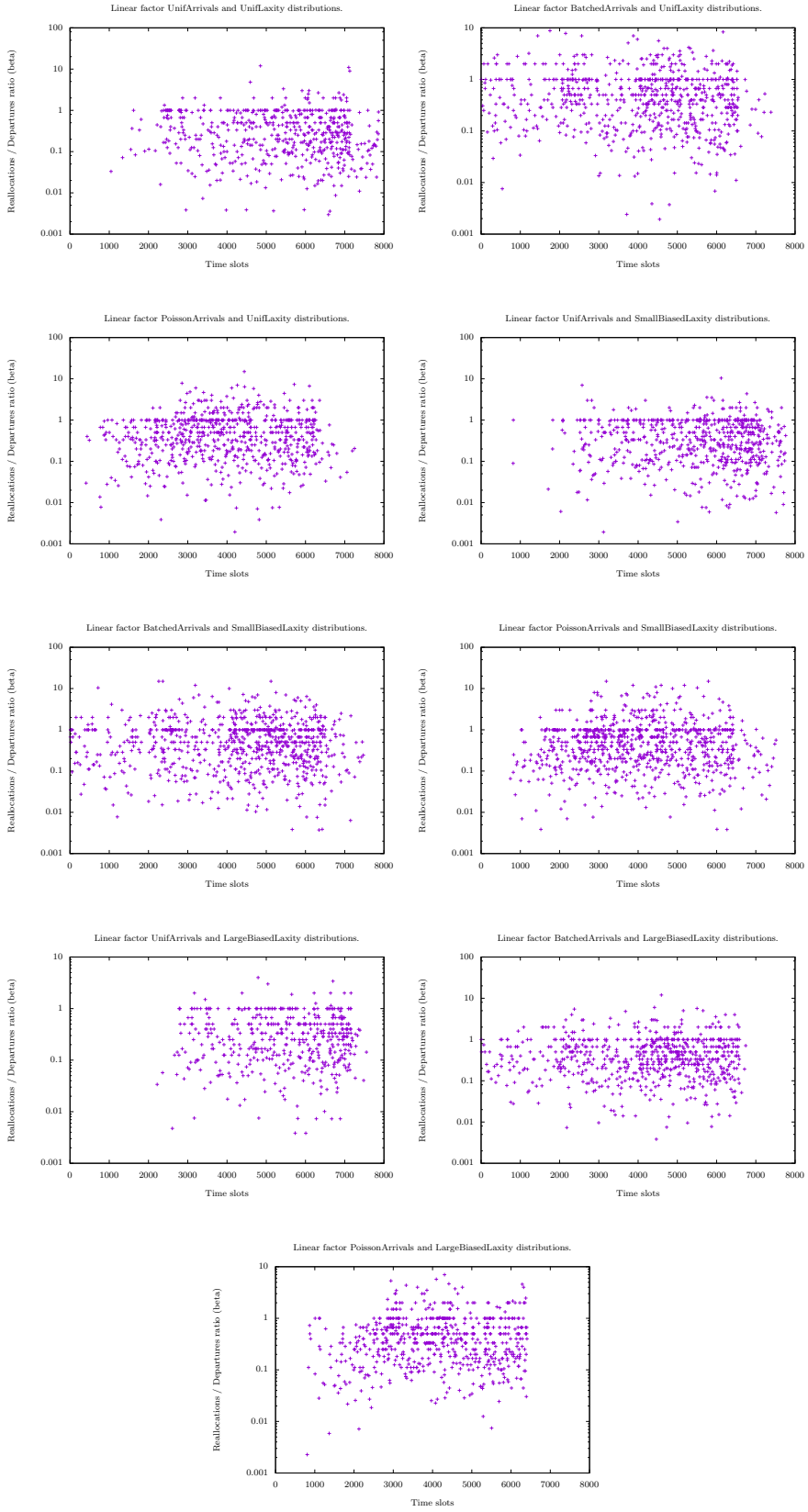


Fig. 11 Reallocation/Departure ratio (β) vs. time for linear classification factor, $n = 4000$ and $w_{\max} = 1024$.

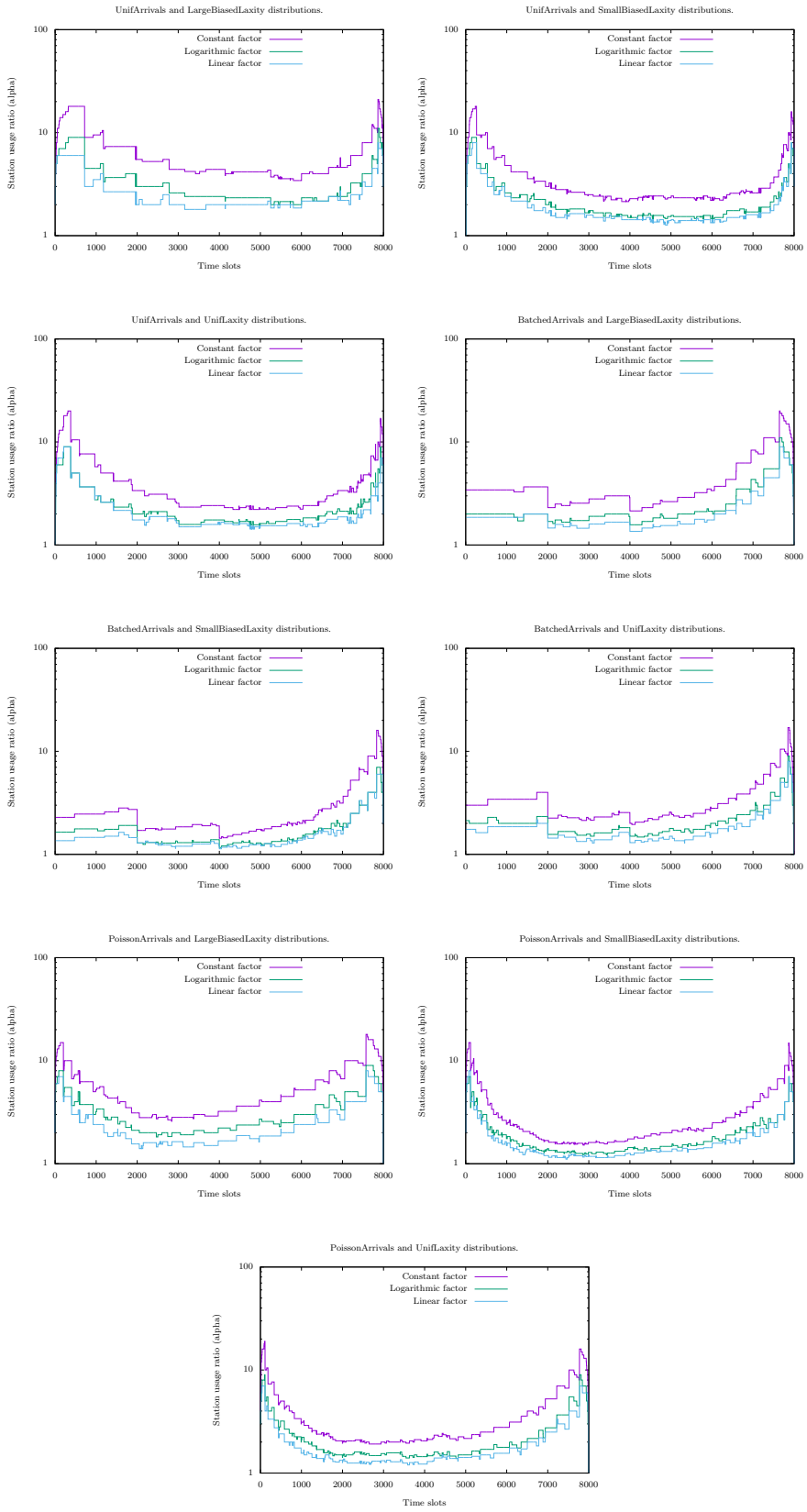


Fig. 12 Station usage ratio (α) vs. time, for $n = 4000$ and $w_{\max} = 1024$.