A self-organizing client / server allocation algorithm for applications with non-linear cost functions

Hanno Hildmann, Sébastien Nicolas
NEC Laboratories Europe, Kurfürsten-Anlage 36, D-69115 Heidelberg, Germany
{hanno.hildmann; sebastien.nicolas}@neclab.eu

Abstract—**We subject a previously presented client-server allocation algorithm to a revised performance criterion to account for non-linear cost functions. We identify the conditions under which the original approach is not guaranteed to outperform the benchmark approach and propose a revised cost prediction method to improve performance for those scenarios. We present empirical results to show the measured benefit of the revised approach over the benchmark as well as the original work.**

I. Introduction & Outline of the paper

In our increasingly mobile world wireless access networks become integral to communication as well as information exchange. Continuous and reliable wireless connectivity is increasing in importance, as well as in its part of the global energy consumption. To address this issue, we previously [1] presented a nature inspired algorithm for the dynamic association of clients to servers in wireless access networks (cf. [2] for the results). The approach was validated through field-testing a prototype implementation (WiFi routers). The performance of both, our simulations as well as the implemented prototype, was measured by the number of servers that could be turned off without compromising the quality of service.

In our original simulation, the potential area of influence of a wireless router was considered circular. However, the power required to provide coverage for a circular area increases non-linearly with the radius of the covered area. Assuming the ability of a server in a wireless access network to regulate (i.e. increase or decrease, in a controlled manner) the radius r of the coverage area, then simply counting the number of active servers may not be a realistic fitness measurement for a population of servers providing service to a population of clients. More specifically, if the ratio between the operational cost (energy consumption) of a server and the energy required to provide coverage is not strongly tipped towards the operational cost, then reducing the number of active servers by increasing the areas of coverage of a few servers may actually result in an increased power consumption of the overall system.

We will first show that the original algorithm does no longer outperform the benchmark (no optimization at all, clients connect to the closest server) if the operational power requirements of the servers fall under a certain ratio (compared to the power requirements for providing coverage). After having shown the limitations of the original algorithm we will then present an alternative version, designed to minimize the overlap in coverage (i.e. reduce redundancy in the provided coverage) and compare its performance to both the benchmark as well as the original algorithm.

II. Self-organizing client-server allocation

A. The original 'Hub Contest' algorithm

1) The original approach works as follows: servers operating below capacity compete for clients with other servers (in their immediate vicinity). Contests are decided using a stochastic decision function which basically follows the *"rich gets richer"* paradigm (i.e. the chances of winning a contest are increasingly favouring the server which already has more clients). This results in the association of clients to as few servers as possible; idle servers will become inactive. The motivation for this is that inactivity is associated with an operational benefit, which is assumed to outweigh the additional operational cost of increasing load or the coverage area of the active servers. The amended approach presented in this paper addresses scenarios where this assumption does not hold.

2) The probability: P_A (server A wins a random client from server B) is calculated by $P_A = \frac{(c+x_A)^\alpha}{(c+x_A)^\alpha+(c+x_B)^\alpha}$; with x_S the number of clients allocated to server S and c, α tuneable parameters. There are two variations of this algorithm, differing only in what is identified first, the other server (the competitor) or the client (object of the contest).

3) The algorithm: on the next page we outline the general structure of the original approach (after servers and client have already been chosen). For brevity the outline covers both, the active as well as the passive behaviour of the server.

```
Input: servers $A$, $B$; client $c$; optimal load $target$
while $\top$ do
    if $A$ is active then
        if load $<$ target then
            compete with $B$ for client $c$
            if win then
                accept client from $B$
            for $\Delta t$ do
                if $A$ is challenged by $B$ then
                    compete with $B$ for client
                    if lose then
                        handover client to $B$
                        if load $<$ minimum then
                            power down $A$
    if $A$ is inactive then
        wait $\Delta t$
        activate if the situation requires it
```

Algorithm 1: The original *"Hub Contest"* Algorithm [3]

4) Energy consumption model: the underlying assumptions for our work are (a) that there is some level of redundancy in the system (which we will leverage) and (b) that by powering down parts of the infrastructure (the original aim of the algorithm) we can gain some operational benefit. To model this we distinguish three factors that, together, account for the entire resource (energy) consumption $\mathcal{E}$ of the system:

- $\texttt{base}$, the consumption when turned off, such as e.g. physical access infrastructure, lighting, memory, etc.
- $\texttt{op}$, required to operate a server as active (no service)
- $\texttt{sc}_i$, the cost incurred by server i to provide service

To simplify matters we can calculate the average service cost $\texttt{sc}^S_{\texttt{av}}$ for any server $i \in S$ (the set of servers) idle or active:

$$\texttt{sc}^S_{\texttt{av}} = \frac{\sum\limits_{i \in S} \texttt{sc}_i}{|S|} \tag{1}$$

The aggregated energy consumption $\mathcal{E}$ of the system is then:

$$\mathcal{E} = |S| \times \left(\texttt{base} + \texttt{sc}^S_{\texttt{av}}\right) + |S_{active}| \times \texttt{op} \tag{2}$$

With $\texttt{base}$ and $\texttt{op}$ fixed, changes (Δ) in $\mathcal{E}$ come from the change in $|S_{active}|$ (active servers) and the service cost $\texttt{sc}_{\texttt{av}}$:

$$\Delta\mathcal{E} = (|S| \times \Delta\texttt{sc}^S_{\texttt{av}}) + (\Delta|S_{active}| \times \texttt{op}) \tag{3}$$

Given the above we can conjecture that by changing the client-server allocation (and thus changing the number of active servers) we may improve on the aggregated resource (energy) consumption of the system (i.e. achieve a negative $\Delta\mathcal{E}$) IFF:

$$-(\Delta|S_{active}| \times \texttt{op}) > (|S| \times \Delta\texttt{sc}^S_{\texttt{av}}) \tag{4}$$

This assumption (that the energy savings from reducing active servers outweighs the resulting increase in energy consumption for providing service) has been shown to hold for wireless access networks consisting of routers and mobile devices [1].

5) Proof of concept implementation & performance: from a hardware point of view the approach has been demonstrated to be feasible for the domain of wireless access networks [1]. In the prototype implementation the required functionalities are already present in standard hardware (i.e. most commercially available routers). Furthermore, since the implementation is exclusively on the server side (in our prototype implementation the servers are WiFi routers), there is no change required to the hardware or software of the clients (the WiFi enabled devices); only the server software needs to be upgraded.

Regarding the performance of the approach we showed that for small range access networks where the operational energy of an idle router is higher than the energy required by this router to provide 100% of its WiFi connections, our algorithm results in an overall reduction of energy consumption.

B. Client-server allocation for non-linear cost functions

The presented approach performs well under certain conditions but we argue that these conditions are not always realistic.

1) Motivation: the original algorithm is designed with the intention to reduce the number of active servers on the assumption that potential increase in service cost is outweighed by the reduction in operational cost. We argue that this assumption does not always hold. To quantify this, we re-write the mathematical formulation from above as follows:

Let's look at $\texttt{sc}^{S_{active}}_{\texttt{av}}$, the average service cost for the active servers, instead of - as before (cf. Eq. 1) - at $\texttt{sc}^S_{\texttt{av}}$:

$$\texttt{sc}^{S_{active}}_{\texttt{av}} = \frac{\sum\limits_{i \in S_{active}} \texttt{sc}_i}{|S_{active}|}$$

The aggregated energy consumption $\mathcal{E}$ of the system is then:

$$\mathcal{E} = (|S| \times \texttt{base}) + (|S_{active}| \times \texttt{op}) + (|S_{active}| \times \texttt{sc}^{S_{active}}_{\texttt{av}})$$

With $\texttt{base}$ a constant, changes in $\mathcal{E}$ are determined by changes in S_{active} (and in the resulting service cost $\texttt{sc}^{S_{active}}_{\texttt{av}}$).

$$\Delta\mathcal{E} = \Delta(|S_{active}| \times \texttt{op}) + \Delta(|S_{active}| \times \texttt{sc}^{S_{active}}_{\texttt{av}})$$

This implies that changing the client-server allocation is only beneficial ($\approx$ reducing the energy consumption of the system, i.e. $\Delta\mathcal{E} < 0$) if $-\Delta(|S_{active}| \times \texttt{op}) > \Delta(|S_{active}| \times \texttt{sc}^{S_{active}}_{\texttt{av}})$.

This implies a fix-point where the sign of $\Delta\mathcal{E}$ changes, which in turn means that there is an area in the solution space after which minimizing the number of active servers does no longer improve the solution (here: results in a negative $\Delta\mathcal{E}$).

Intuitively this does make sense: we know that in the cases where the operational cost of an idle server is negligibly small, turning servers off will not, by itself, have a relevant effect on the aggregated energy consumption. On the other hand, if the operational cost accounts for e.g. 99% of the power consumption then turning off servers will almost certainly decrease the overall power consumption. The question then is when does the operational cost have an impact and, more interestingly, when does this impact result in an increase instead of a decrease of the aggregated power consumption.

These considerations suggest that the tipping point when the ratio between $\mathtt{op}$ and $\mathtt{sc}_{\mathtt{av}}^{S_{active}}$ does have an impact on the entire system's power consumption is not as dramatic as *'the operational cost does account for a vast majority of the infrastructure's power consumption'* but instead is more along the lines of *'if the average service cost exceeds the energy consumed by the idle base station'*.

While the results presented in [2] stand - and even though we have shown that the original approach is beneficial for WiFi routers and clients [1] - we argue that there are many conceivable scenarios where e.g. the energy required to provide coverage exceeds the energy required to keep the hardware running by a multiple. The remainder of this paper is dedicated to formulating and evaluating an improved algorithm, specifically designed to perform well in such scenarios.

2) Revising the service cost function: on the assumption that the power requirements for delivering a service are relevant in the calculation of the entire system's power consumption we present an amended energy consumption model.

In §II-A4 we listed $\mathtt{sc}_i$ as one of the three factors that, together, account for the total consumed energy. In Equation 1 we used this to calculate the average power consumption $\mathtt{sc}_{\mathtt{av}}^{S}$ of any server. What was implicitly assumed (and explicitly stated in [1]) is that the power usage $\mathtt{sc}_i$ grows *"at least marginally"* with the number of clients affiliated to i:

$$\mathtt{sc}_i = f(x_i)$$

where $f(x_i)$ is the function calculating the energy cost incurred by server i for providing service to x_i (the clients of i).

We stated in [1] that *"the aggregated energy consumption over the whole network will be reduced* [...] *if $f(x)$ is a sublinear function"*, i.e. that the algorithm performs well if:

$$\lim_{x \to \infty} \frac{f(x)}{x} = 0$$

And therein (in the assumption that $f(x)$ is sub-linear), lies the potential problem: in wireless access networks the energy required to maintain a connection grows with the distance. Increasing a server's area of coverage (which is a likely result when migrating clients to other servers) from A to B increases proportionally to $(r_B^2 - r_A^2)$ (if the coverage is circular, with r_A and r_B the respective radii).

A client that is close to one server may be far from another and the change in power consumption for the servers (resulting from re-allocating the client between them) may be vastly different. We therefore need to consider not only the number of affiliated clients but also the relative distance (r) to their servers. Ignoring constants such as antenna gain, π, etc. and under the assumption that we are dealing with a homogeneous group of clients (with regard to their service requirements), then $\mathtt{sc}_i$, the energy cost incurred by i to maintain connection with its clients x_i, is proportional to (cf. Friis Law):

$$\mathtt{sc}_i = \sum_{k \in x_i} (r_k)^2$$

Practically speaking we may not know the distance between the clients and the server (e.g. in 2G networks) and base stations transmit to all mobile terminals at the same power. We therefore use the value of the maximum distance (i.e. the radius of the coverage area required to serve all the server's clients):

$$\mathtt{sc}_i = \max_{k \in x_i} (r_k)^2 \times |x_i|$$

3) Amending the stochastic decision function: to evaluate the impact of re-allocating a specific client k from server i to server j we compare the current as well as the hypothetical situation (i.e. i and j versus i' and j', where $x_{i'} = x_i \backslash \{k\}$ and $x_{j'} = x_j \cap \{k\}$, in other words: i, j are the servers before the re-allocation and i', j' are the servers afterwards).

$$(\mathtt{sc}_i + \mathtt{sc}_j) \quad \text{vs.} \quad (\mathtt{sc}_{i'} + \mathtt{sc}_{j'})$$

This is a straight forward comparison and using it allows us to consider the differences in cost increases (instead of simply counting the number of affiliated clients, as done in [1]).

The new (cf. §II-A2) stochastic decision function to determine P_j (the probability of j stealing a client from i) is:

$$P_j = \frac{(\mathtt{sc}_{i'} + \mathtt{sc}_{j'} + 2\mathtt{op})^{\alpha}}{(\mathtt{sc}_i + \mathtt{sc}_j + 2\mathtt{op})^{\alpha} + (\mathtt{sc}_{i'} + \mathtt{sc}_{j'} + 2\mathtt{op})^{\alpha}}$$

It is crucial to remember that the client that is re-allocated is not (as it was in [1]) a random client because the value of P_j will depend on the client's distance to the two servers.

III. Methods and materials

A. Modelling the problem

Of the 6 parameters used to model the problem in [1] we kept the following 3 (omitting *mapsize*, *range* and *density*):

1) *clients* (c): the number of clients (placed randomly)
2) *servers* (s): the number of servers (placed randomly)
3) *speed* (v): the distance a client travels per time step

Contrary to [1], the performance measure is not the number of active servers but their aggregated power consumption.

B. Investigated approaches

We compared the consumed energy for these 3 approaches:

1) The benchmark: clients are allocated to the closest server; servers are never turned off. This is a workable solution and represents the standard process currently used in wireless access networks.

2) The 'linear Hub Contest': our original approach [1], [2], which strictly aims to reduce the number of active servers.

3) The 'non-linear Hub Contest': the revised approach which considers the actual energy consumption of the potential client-server allocations (cf. decision function in §II-B3).

C. Implementation choices

1) Initial placement and allocation of clients: clients are placed randomly and are initially allocated to the closest server.

2) Random placement of servers:: we impose a minimum distance (x) to all other servers for the first 20 attempts to randomly place a server. This failing, the algorithm will reduce the exclusion value x_{old} by a third: $x_{new} = x_{old} \times \frac{2}{3}$. This process of reducing the minimum exclusion distance is repeated five times before a purely random location is chosen.

3) Movement of clients: clients travel towards assigned locations, upon reaching them they are assigned a new one.

D. Performance measure

For the performance evaluation and comparison in the next section, the energy cost $\mathcal{E}_i$ of a server i is calculated - for all three approaches alike - by $\mathcal{E}_i = \texttt{op} + \max_{k \in x_i}(r_k)^2 \times |x_i|$, the consumption $\mathcal{E}$ of the entire system is thus $\mathcal{E} = \sum_{i \in S} \mathcal{E}_i$.

E. Data collection

Simulations were run in batches and multiple times for each data point. We report the averages over these runs, standard deviation (sdev) and other relevant values are reported.

IV. Results and discussion

The results presented were obtained from running simulations of the different approaches in parallel on the same server cluster over a period of 2 weeks. The parameter settings that were used were determined through an initial parameter space exploration based on 150 runs for each of the approaches.

We report the final number of active servers and the resulting power consumption (i.e. after a simulation has terminated).

A. Performance comparison (aggregated consumption $\mathcal{E}$)

Let's first compare the performance of the three approaches in the context of the aggregated power consumption.

1) Methods and materials: running the simulations in parallel yielded data sets of varying sizes due to the different computational cost of the respective simulations and associated running times. Each data point is aggregated (from 3428 simulations for the benchmark, 512 for the original algorithm and 310 for the revised algorithm). Figure 1 shows the resulting consumption for varying operational costs for servers.

Each simulation had 100 servers and 10,000 non-moving clients, both randomly placed; simulations terminated after 5000 iterations (convergence occurred around iteration 500).

2) Results: We recorded the number of active servers at the end of a simulation as well as the respective distances between each server and the allocated clients. The graphs shown in Figure 1 are based on 18 active servers for the original algorithm (min=17.9, max=18.1, sdev=0.04), 70.2 active servers for the revised algorithm (min=69.4, max=70.5, sdev=0.28) and (obviously) 100 active servers for the benchmark case.

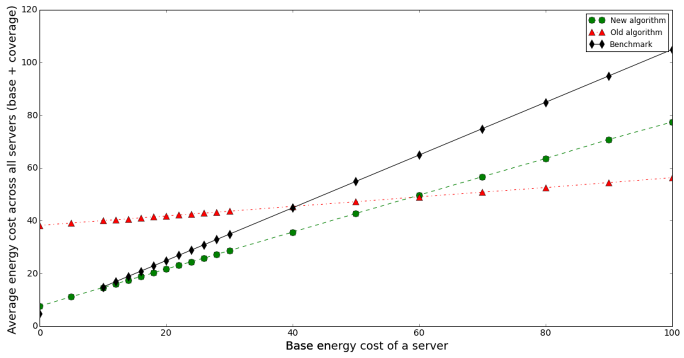

Fig. 1. Comparison of the 3 approaches. On the x-axis we plot the increasing operational cost `op` of a server, the y-axis shows the average server consumption `sc`. Note: in order to facilitate comparison the latter is the *average consumption* for *all* servers, including the ones that are off.

Figure 1 shows that that our concern about the original algorithm was justified: when comparing the performance of the original algorithm against the benchmark we find that the the benefit only occurs when the operational cost `op` of a server exceeds a threshold (here at a cost of approx. 40).

Contrary to this, the revised approach does consistently outperform the benchmark (see discussion), however, for very high operational costs (here at a cost of approx. 60) it is outperformed by the original approach (cf. §IV-B for a discussion on the impact of the ratio between `op` and `sc` on performance).

3) Discussion: As suspected, the ratio between `op`, the power required to operate a server as active (without providing any service) and $\texttt{sc}_i$, the power required by server i to provide service to all its clients, impacts the performance of the original algorithm. Furthermore, we see that neither the original nor the revised algorithm are consistently superior to the other.

Although we have claimed above that the new approach consistently outperforms the benchmark, this is not true in the extreme cases where the operational cost is minimal or non-existent. The explanation for this is simple: if there is no operational cost and shutting down servers has no benefit by itself then the benchmark (allocating clients to the closest server) is very likely to be the best possible solution, experimenting with alternatives can only result in sub-optimal solutions.

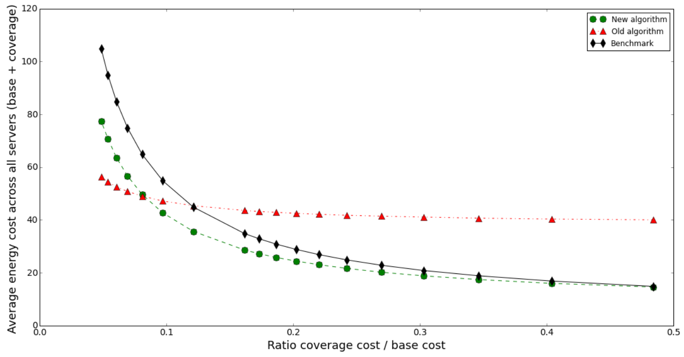

Fig. 2. The results presented in Figure 1, plotted against the ratio (x-axis) between service cost (`sc`) and operational cost (`op`). In our scenario, the original algorithm is outperformed when `sc` increases above 8-15% of `op`.

We maintain that, practically, our algorithm is superior to the benchmark because we argue that, realistically, servers will always have some operational cost associated with them.

B. Ratio: service cost ($\mathtt{sc}$) / operational cost ($\mathtt{op}$)

The motivation for this paper is that the original algorithm does not consider the ratio between service cost ($\mathtt{sc}_i$) and operational cost ($\mathtt{op}$) of a server but assumes that reducing the number of active servers is always beneficial. Our results show that this is not always the case.

While the impact of the ratio is difficult to visualize in a meaningful way the above graph (Figure 2) gives an indication of the order of magnitude involved. Depending on the scenario, it should be clear to an engineer whether the power drain for the actual service is negligible or whether it constitutes a factor that would render the original algorithm ineffective.

C. Control parameter α

In §II-B3 we describe the amended decision function. As before in [1] we include tuning parameters to facilitate the tailoring of the algorithm to a specific scenario. α effectively enables us to control the sensitivity of the algorithm.

When decisions are made with a probability $P = \frac{(A)^\alpha}{(A)^\alpha + (B)^\alpha}$, then the larger the value for α is, the more impact the difference between A and B will have on the decision.

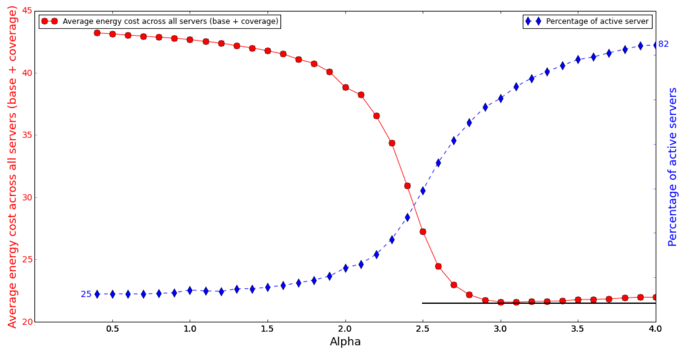

Fig. 3. The impact α has in our scenario: small values (]0,1.5]) have little, while too large values (]3,4]) can increase the number of active servers without reducing the overall power consumption further. In our simulations the optimal value for α is 3 and this value was used in all simulations.

The results shown in Figure 3 are the average values from 541 simulations. With increasing α the decision function becomes more sensitive to potential but small improvements; for values between 2 and 4 we see a sharp increase in active servers and a substantial reduction in the overall consumed power. The optimal value is reached for $\alpha = 3$, after which the performance of the algorithm decreases slightly.

The average sdev over the entire simulation and all data points is 0.75 (min=0.41, max=1.7, sdev= 0.39), for the lower values (in the right half of the plot) it is as low as 0.57.

D. Dynamic environments

In the original investigation [1], [2] we considered dynamic environments, i.e. we simulated moving clients. In the context of wireless access networks this is one of the fundamental properties of the scenario as well as one of the reasons why brute-forcing the optimal solution is not an option.

The above discussed results are obtained for static clients, the motivation being that the goal is to compare the performances of the algorithms under as isolated settings as possible. However, a performance evaluation of the new algorithm would not be meaningful if we were to omit the comparison of the algorithms for dynamic scenarios.

As before in [1], the convergence speed of the algorithm affects performance (since only a certain amount of time is available to find a solution before the scenario has changed sufficiently to debunk it). Due to this, we also tested a variation of our new algorithm where the process of choosing a client to hand over is not random: clients are chosen on the basis of how much an exchange will benefit the overall solution. This version of the algorithm, while substantially more goal directed, does not consider any information about the scenario nor does it make use of any intelligent choice mechanism.

1) Materials and methods: we compare the performance of the algorithms for increasing velocity of the clients. In the simulations the velocity of clients is represented by Euclidean distance between the locations a client may occupy in two consecutive iterations.

In total we simulated the benchmark case 1174 times, the original algorithm 235 times, our new algorithm 149 times and the tailored version of our algorithm 130 times.

2) Results: We first compared (for all three algorithms) the number of active servers (Figure 4) and then evaluated the resulting aggregated energy consumption (Figure 5).

a) The impact of dynamicity on active servers: the results shown in Figure 4 are the averages, with a mean sdev of 1.71 (original algorithm, min=1.1, max=3.06, sdev=0.57), 2.16 (new algorithm, min=1.44, max=3.6, sdev=0.33) and 2.12 (improved new algorithm, min=1.69, max=3.22, sdev=0.33).

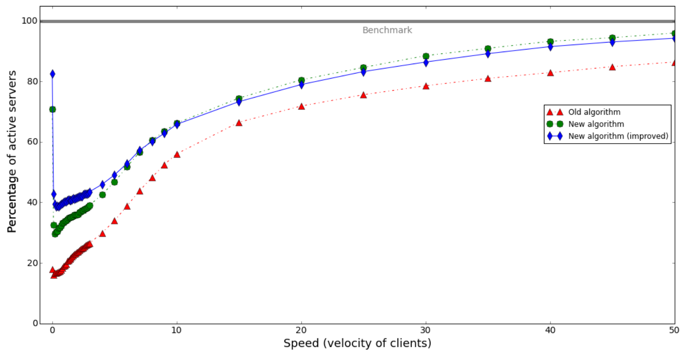

Fig. 4. The impact of increasing velocity of client movement on the performance of the algorithms. Since in the benchmark case servers remain active this provides an upper bound. As in [1], it seems that small amounts of entropy can improve the performance, probably by removing dead-locks.

For all tested algorithms, increased dynamicity (velocity of clients) causes a rise in active servers. Interestingly, adding a small amount of dynamicity to a static scenario does dramatically improve the number of active servers (see discussion).

b) The impact of dynamicity on power consumption: as before, Figure 5 reports averages; the respective mean sdev are 0.53 (original algorithm, min=0.4, max=0.87, sdev=0.13), 0.69 (new algorithm, min=0.51, max=1.7, sdev=0.19) and 0.75 (improved new algorithm, min=0.45, max=1.11, sdev=0.16).

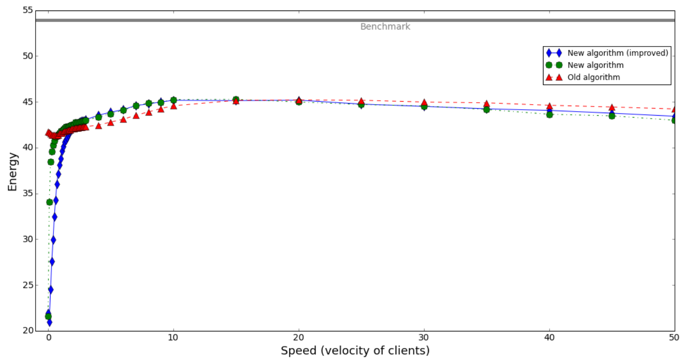

Fig. 5. The good performance for static clients discussed in §IV-A quickly dissipates when dynamic clients are introduced. This is due to the much slower convergence speed of the new algorithm. An improved version of the new algorithm only mitigates this briefly before it also succumbs to dynamicity.

As above we see a massive change when dynamicity is added to a static scenario. For higher velocity of clients the performance of all algorithms converges (though remaining well under the benchmark) and the impact of the revised algorithm is nullified.

3) Discussion: Adding dynamicity to the scenario has a massive affect on the performance of all algorithms. With regard to the number of active servers this indicates that, as for [1], small amounts of entropy help to overcome dead-locks or to avoid getting stuck in local optima. The impact of this can be seen in Figure 4, and it is substantially bigger for the new algorithms than it was for the original one.

As soon as the clients move more than just very small amounts, all algorithms increase the number of active servers, presumably because it takes time (iterations) to re-allocate the clients. As the velocity of clients increases the algorithms are hindered by their convergence speed. Eventually, i.e. when clients randomly appear and disappear, the system would converge to a state where all servers are active (though this is not shown in the graphs of Figures 4 and 5).

The goal is to minimize the aggregated energy consumption of the system, not the number of active servers. Figure 5 plots the performance of the algorithms. We notice that the new algorithm, which performs very well for the static scenarios, very quickly is outperformed by the original algorithm in scenarios with more than small amounts of movement.

We attribute this to the vastly larger convergence time of the new algorithms which effectively prevents the finding of a solution before the scenario has changed again. Only for very large velocity does the original approach perform worse than the new ones; it is our interpretation that this happens when the velocity of the clients reaches a level that effectively turns the sever-client allocation problem into a matter of simply providing coverage over an area. The new algorithms are designed to be conscious of large areas of coverage and thus are more likely to remove redundant (overlapping) coverage. However, this has only a marginal impact when compared to the performance for static scenarios (which effectively sets the bar for what could be achieved if the algorithms had more time to converge to good solutions).

E. Caveat

As §IV-D clearly shows that the convergence speed of the algorithms matters. The fact that the convergence speed of the algorithms strongly affects the performance for dynamic environments indicates that a comparison of the original and the new approach is not meaningful unless both are taking properties of the scenario into account.

The aim of this paper was to address the authors concerns regarding the performance of the original algorithm for non-linear cost functions. We have shown that this is indeed an issue and have offered a proof-of-concept solution to it by designing and testing an algorithm that addresses this issue.

Having informed the community of the shortcomings of our previously presented work we plan to work towards a more sophisticated solution to address non-linear cost functions.

F. Future work

This paper is entitled *"A self-organizing client / server allocation algorithm for applications with non-linear cost functions"*, and we feel that we have indeed presented such an algorithm, albeit a generic one which has not been stream-lined (tailored to the scenario it is applied to) at all. Future work will focus on designing an approach to facilitate client-server allocation for non-linear cost functions which either addresses specific types of scenarios (and thus can be tailored) or which places emphasis on improved convergence speed.

References

[1] H. Hildmann, S. Nicolas, and F. Saffre, "A bio-inspired resource-saving approach to dynamic client-server association," *IEEE Intelligent Systems*, vol. 27, no. 6, Nov. - Dec. 2012.

[2] ——, "Energy optimisation of the wireless access network through aggregation of mobile terminals," in *2012 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Sept. 2012, pp. 1229–1234.

[3] Patent Application, "Method and system for determining allocation of clients to servers," US Patent Application No: 13/339,763 (pending), F. Saffre, H. Hildmann and S. Nicolas (inventors), Dec. 2011.