A distributed power-saving framework for LTE Het-Nets exploiting Almost Blank Subframes

Antonio Virdis(1), Giovanni Stea(1), Dario Sabella(2)(*), Marco Caretti(3)

(1) Dipartimento di Ingegneria dell'Informazione, University of Pisa, Italy
a.virdis@iet.unipi.it, giovanni.stea@unipi.it

(2) Intel Deutschland GmbH, Munich – dario.sabella@intel.com

(3) Telecom Italia, Turin, Italy – marco.caretti@telecomitalia.it

Technical Report, University of Pisa, March 2017*

* An abridged version of this manuscript has been accepted for publication in *IEEE Transactions on Green Communications and Networking* in June 2017.

(*) This work was carried out while Dario Sabella was with Telecom Italia, Italy.

***Abstract*—Almost Blank Subframes (ABS) have been defined in LTE as a means to coordinate transmissions in heterogeneous networks (HetNets), composed of macro and micro eNodeBs: the macro issues ABS periods, and refrains from transmitting during ABSs, thus creating interference-free subframes for the micros. Micros report their capacity demands to the macro via the X2 interface, and the latter provisions the ABS period accordingly. Existing algorithms for ABS provisioning usually share resources proportionally among HetNet nodes in a long-term perspective (e.g., based on traffic forecast). We argue instead that this mechanism can be exploited to *save power* in the HetNet: in fact, during ABSs, the macro consumes less power, since it only transmits pilot signals. Dually, the micros may inhibit data transmission themselves in some subframes, and optimally decide *when* to do this based on knowledge of the ABS period. This allows us to define a power saving framework that works *in the short term*, modifying the ABS pattern at the fastest possible pace, serving the HetNet traffic at reduced power cost. Our framework is designed using only standard signaling. Simulations show that the algorithm consumes less power than its competitors, especially at low loads, and improves the UE QoS.**

***Index Terms*—LTE, Interference Control, ABS, e-ICIC**

1. Introduction

Improving cell-edge performance in urban areas and increasing the energy efficiency of base stations have been among the main issues of LTE-Advanced research in recent years (e.g. [1],[2]). These requirements are met by using denser deployments, using a higher number of base stations, having diverse characteristics in terms of transmission power, coverage and power consumption: high-power nodes, called *macro*, cover large areas, whereas low-power ones, generally called *micro*, are used to boost capacity or extend coverage in specific zones (Figure 1). Dense deployments with heterogeneous base stations are called *HetNets*.

To allow efficient operations in HetNets, 3GPP introduced a set of techniques called *enhanced Inter-Cell Interference Coordination* (eICIC), since Rel-9. Notably, two mechanisms have been proposed. A fist one, *cell-range expansion* (CRE) can be used to favor the association of UEs to micro nodes, possibly offloading the macro. As a side effect, UEs at the edge of macro and micro cells may experience poor channel conditions due to interference. For this reason, a second mechanism has been devised, i.e. allowing the macro to inhibit data transmission during certain subframes, called *Almost Blank Subframes* (ABSs), to reduce interference to cell-edge UEs in micro cells. During ABSs, however, the macro still transmits pilot signals, hence ensuring correct operation of the UEs under its control. ABSs are arranged by the macro in *ABS periods* (APs) of 40 subframes, whose composition is transmitted to all the micros in the HetNet through the X2 interface. A slightly different embodiment of the ABS concept, discussed in [3], consists in allowing the macro to transmit at a reduced power during ABSs. This allows the macro to serve *some* UEs (those nearer to it), possibly with a reduced SINR, and consuming less power. On the other hand, UEs attached to micros will perceive a mitigated interference from the macro, hence will be able to hear their serving micro.

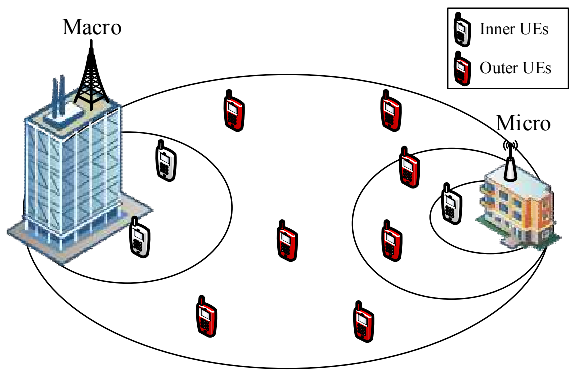

Figure 1 – Example of HetNet deployment.

Recent research [1] has shown that a node that only transmits pilot signals – such as a macro during ABSs – consumes *less power* than in a normal subframe, since it can power down some circuitry. This creates a power saving opportunity: provisioning an AP so that the macro packs all its data traffic into as few subframes as possible, and then powers down some circuitry and transmits ABSs in the rest of the subframes, allows it to carry the same traffic at a reduced power cost. Moreover, micro nodes may adopt a similar behavior: when they receive an AP, they decide how many SFs they need to use, and of which type (whether ABSs or non-ABSs) to carry their traffic. Therefore, they too can pack their traffic into as few subframes as possible, and only transmit pilot signals in the other ones, thus saving some more power.

Now, both a macro and a micro need to estimate the load that they will carry in the next AP in order to decide when to power down. Since that load may vary in the short term, there is a clear need for a *dynamic* ABS provisioning framework. The vast majority of the existing schemes (e.g. [4], [5]-[9]) consider a *semi-static* ABS partitioning, based on long-term capacity requirements. This way, they implicitly give up the opportunity of riding the peaks and valleys of the traffic demand at the nodes, which is instead the principal lever for power saving. There is, besides, no practical impediment to having a new, different AP at every AP boundary, if the required computations complete in time. Even the few dynamic schemes appeared so far, however, (e.g., [10]) do not consider power saving opportunities. Moreover, to the best of our knowledge, all schemes presented so far *either* assume that macro transmission is completely inhibited during ABSs, *or* assume low-power data transmission during them, and never consider that switching between *both* solutions, or employing them both simultaneously in the same AP, may improve performance and/or save power.

In this paper, we propose a practical framework for power saving that exploits *dynamic* ABS provisioning. Based on periodic reports from the micros and its own aggregate load and channel quality measures, the macro node selects how many “idle” ABSs (I-ABSs, those where the macro does not transmit any data), how many “low-power” ABSs (LP-ABSs, those where the macro does transmit data at a reduced power), and how many non-ABSs to provision in the next AP, so that all the HetNet load is carried, if at all possible, at the minimum power cost. When the network is overloaded (i.e., the capacity demands of the micros and the macro cannot be accommodated simultaneously), our framework degrades the service proportionally at all the coordinated nodes.

The points of strength of our framework are the following: first, to the best of our knowledge, ours is the first framework to leverage ABS provisioning for power saving. Moreover, it is the only one that takes into account *both* I-ABSs *and* LP-ABSs simultaneously in a unified framework. We show that slight variations in the HetNet scenarios - e.g., in the position or traffic of UEs - such as those that may occur dynamically at the same timescale at which our framework operates, make *different combinations of these mechanisms* optimal from a power saving standpoint. Using both I-ABSs and LP-ABSs allows more energy-efficient transmissions, hence greater power savings or higher throughputs, than using either of the two mechanisms in isolation. It is well known that power saving comes at a cost in terms of increased latency: however, our framework also *minimizes the latency* for a given level of power saving, thus offering the best possible trade-off. Moreover, since our power-saving mechanism (i.e., keeping nodes off as much as possible) also reduces interference, it increases the transmission efficiency, which further reduces latency for a given traffic load. Second, but not less important, our framework employs only *standardized signaling*, i.e. nodes are assumed to know *only* what the standard allows them to about the status of the HetNet. This makes our framework practically implementable. Third, it does not require complex computations: both the macro and the micro nodes run only *simple algorithms*, that are independent of the number of users and scale well with the number of nodes. Fourth, it works under broad hypotheses, e.g., with arbitrary numbers of micros and network topologies, it accommodates a large class of power consumption models, and does not rely on a particular traffic or UE mobility model. Besides, it is orthogonal to algorithms running at both *slower* timescales, such as CRE bias selection for user association or network topology adaptation through selective node switching, and *faster* timescales, such as MAC-level scheduling. These will generate input data to our framework, which will then maximize the power saving.

We evaluate our framework via simulation against a dynamic ABS provisioning scheme, showing that its power savings are remarkably higher, and that this does not come with any performance degradation: on the contrary, the cell throughput stays the same, and the user delay distribution improves considerably.

The rest of the paper is organized as follows: Section 2 reports background on LTE. Section 3 reviews the related work, and we describe the system model in Section 4. Our power-saving framework is explained in Section 5, and Section 6 reports performance results. We conclude the paper in Section 7.

2. Background on LTE

In this section we describe those features of LTE that are more relevant to the problem at hand, i.e. downlink resource allocation at the MAC layer and the eICIC framework.

In an LTE network, an eNodeB (eNB) allocates resources to its UEs, composing transmission schedules (called *subframes, SFs*) periodically. A *cell* is an area where UEs are associated to an eNB, hence share resources taken from the same SF. Scheduling within a cell occurs on every Transmission Time Interval (TTI), whose duration is 1ms, and consists in the eNB allocating a vector of *Resource Blocks* (RBs) to UEs (one RB goes to one UE only[2]). The number of bytes per RB is indirectly determined by the Channel Quality Indicator (CQI) reported by the UE to which that RB is allocated. The CQI depends on the measured Signal-to-Noise-and-Interference Ratio (SINR), and is reported periodically or on demand. Downlink transmissions are protected by a Hybrid ARQ (H-ARQ) scheme: the UE reports an ACK/NACK, and the eNB may reschedule the same transmission up to four times in a future TTI. H-ARQ errors often originate when the reported channel state (as per the UE CQI) is better than the one at the time of transmission.

The eNBs are often categorized according to their role and radiation power. Large-scale cellular coverage is provided essentially by *macro* eNBs, i.e. high-power, large-coverage eNBs. Lower-power eNBs are normally called *micro*, or *pico,* or *femto* eNBs (depending on their power and possibly other factors), and provide *additional localized* capacity, to increase the UE data rate where needed. For this reason, we will henceforth refer to all of them as *micro* eNBs for simplicity, any difference between them being immaterial for the purpose of this paper. Micro cells, thus, are embedded within macro cells, and UEs in micro cells suffer interference from the macros.

The eNBs communicate with each other – or, possibly, with other network entities – through the so-called X2 interface [11], which includes both a control and a data plane. The protocol stack for the control plane is shown in Figure 2. Signaling information are generated by the X2 Application Protocol (X2AP) [12], which defines a large set of procedures and messages for supporting inter-eNB operations, including those related to eICIC, which we describe next. The X2 uses the Stream Control Transmission Protocol (SCTP) at layer 4, which establishes and maintains the association between two peering eNBs, and whatever link-level technology is available below the IP layer.

[2] Multi-user Multiple-Input/Multiple-Output (MIMO) techniques are outside the scope of this paper.

In order to limit inter-cell interference, the standard [13] allows a macro to define and enforce *Almost Blank Subframes* (ABSs), i.e. SFs where it either refrains from transmitting downlink data altogether, limiting to common reference signals and pilot signals [14], or transmits downlink data at a reduced power, so that UEs under micros will experience a better SINR. We have already distinguished *idle ABSs* (I-ABSs), and *low-power ABSs,* (LP-ABSs). The acronym ABS will denote both when no distinction is needed.

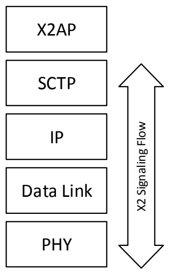

Figure 2 - Description of the X2 stack.

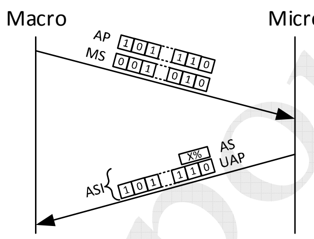

Figure 3 - Standard-based message exchange between macro and micro.

ABSs are organized into an *ABS pattern* (AP), composed of 40 consecutive SFs in an FDD deployment, which is repeated periodically. The AP can be varied at the end of the period by the macro. An AP is sent to the micros within an X2 message, as a vector of binary values, stating whether a SF is/is not an ABS, as shown in Figure 3. Moreover, the macro sends another bit vector to complement the first, called the *Measurement Subset (MS)*, which indicates to the micros *in which SFs their UEs should measure* interference. A macro may thus instruct a micro to evaluate CQIs in ABSs, to assess whether the interference in these is acceptable or not. On request of their macro node, micro nodes report periodic ABS Status IEs (ASIs), still using the X2 interface (see again Figure 3). The latter include:

- ABS Status (AS): A value from 0 to 100 stating the *percentage of employed RBs* over the total available on the subset of ABSs defined in the next field. By reporting AS=100, micros signal that they need more ABSs.
- Usable ABS Pattern Info (UAP): a string of binaries indicating the set of ABSs over which the previous percentage is computed. This must be a subset of the ABSs of the last AP, possibly the whole set of them, composed by ABSs where the measured interference is below a configurable value.

The overall exchange of information takes 10+6*n* bytes per AP at the application level, *n* being the number of micros [12].

3. Related Work

The study of eICIC techniques has received considerable attention in the last few years, and many research papers are being published on the topic. Almost all works on eICIC pursue one or more of the following objectives: i) selecting an ABS ratio; ii) influencing or selecting the UEs association to macro and micro nodes, and iii) selecting the transmission power used by either or both the macro and the micros. Table 1 matches some recent works with the above topics.

TABLE 1 - LIST OF RELATED WORKS AND RESPECTIVE TOPICS

	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[15]	[16]	[17]	[18]	[20]	[21]	[22]	[23]	[24]	[25]	[26]
ABS ratio	✓	✓	✓	✓	✓	✓	✓	✓	✓				✓			✓		
UE association	✓	✓			✓			✓		✓	✓	✓	✓			✓		
Transmission power		✓		✓		✓								✓	✓	✓	✓	✓

More in detail, works that select the ABS ratios either operate at a network-wide scale, i.e. coordinate many macros together, or at the macro-cell scale, by coordinating *one* macro with its micros. Our work can be framed within this last stream. In [15], instead, a hybrid approach is defined: groups of macros coordinate with the same target micro, so that each micro UE is protected from the highest-interfering macro. Unlike ours, however, most of these works assume that ABS ratios can only be chosen within a` small, predefined set (e.g. 1/8, 1/4, etc.), and/or are configured statically or semi-statically, at a pace of tens of minutes [4]. The only works we found that tackle the selection of the ABS dynamically are [10] and [16]. Work [10] defines an algorithm to select the ABS ratio in an AP based on the number of resources requested by the macro UEs and micro *inner* and *outer* UEs, these being, respectively, those UEs that can/cannot hear the serving micro well when the macro transmits. The number of resources for each group X of UEs is computed as $V_X = \sum_{i \in X} B_i / R_i$, where B_i is the amount of data in the buffer for user i , and R_i is the channel rate. This scheme thus requires the algorithm to be *omniscient*, i.e. to know both the buffer status and the CQIs for each UE anywhere in the HetNet. This requires a non-negligible communication overhead, which cannot be mapped on the standard ABS signaling. The ABS ratio is then computed as $\theta = \sqrt{V_{\mu-outer}} \Big/ \left(\sqrt{V_{\mu-outer}} + \sqrt{V_{\mu-inner} + V_M} \right)$ (μ and M denoting the micro and the macro, respectively). Work [16] advocates selecting *almost blank resource blocks (ABRB)* instead of ABSs, and shows how to do so in a near-optimal way, with the objective of maximizing a concave function of the overall physical-layer data rate, assuming that the central controller possesses a HetNet-wide topology graph, stating which eNB interferes with which UE. On one hand, ABRBs are not supported in LTE, and standard UE CQI reporting alone does not allow one to construct a topology graph. On the other, the above scheme requires that, on each AP, per-UE information is conveyed to a central HetNet controller, which solves several convex optimization problems, whose size depends on the number of eNBs and UEs in the HetNet. Convex optimization has superquadratic complexity in general, and no computational results are reported in [16] to understand at which timescales and HetNet sizes or populations this is feasible.

Works that optimize the UE association usually aim at offloading macro traffic to micros by varying the CRE Bias [17], and are often associated to static ABS-selection algorithms. A small subset of these, instead, propose methods for directly *selecting* the serving node of a UE, based on cell load and/or channel quality information [15]. Work [18] considers a heterogeneous TDMA network with cellular and Wi-Fi base stations occupying non-interfering spectra, and selects to which infrastructure a UE should connect, so that the cellular operator minimizes the energy cost of its network, while still getting revenue from it. Energy cost reduction is done by switching off cellular eNBs, something that is infeasible at per-AP timescales[3]. Work [20] shows that setting a load-aware CRE bias leads to higher spectral efficiency than associations based on the received power only. Work [21] tackles joint UE association and ABRB provisioning in a unified framework, assuming massive MIMO and Joint Transmission from clusters of eNBs, and computes an upper bound on network performance. Works that focus on the transmission power generally aim at selecting the power of macros (during ABSs or LP-ABSs), and/or micros (during non-ABSs) [22]. The transmission power is generally set *semi-statically* based on the network topology. Work [5] proposes an analytical framework based on stochastic geometry to set the power level of LP-ABSs, the ABS ratio, the CRE bias and the thresholds for inner/outer UE classification. Work [23], instead, advocates a dynamic approach and varies the transmission power at a faster pace, using the UE positions as an input. Work [24] uses reinforcement learning to have a micro learn what RBs to use to serve its UEs, at what power, and using what CRE, implicitly positing ABRBs. Moreover, it leverages dynamic Carrier Aggregation to allow a UE to be served by both a macro and a micro, on non-overlapping carriers. It is worth noting that varying the power (hence the network coverage) and/or the CRE at a fast pace is often criticized (see e.g. [10]). Such approach, in fact, may lead to unexpected ripples on the network at large, such as interference fluctuation, etc., which in turn may reduce the effectiveness of channel prediction mechanisms and lead to higher error rates [22],[25]. Finally, within this group, [26] proposes to use arrays of antennas on the micro side, to adapt the radiation pattern thus protecting macro edge users from interference.

Schemes for selecting UE association, node transmission power and UE classification thresholds, all in a long-term perspective, can indeed be used to set the input values for our framework. As far as works targeting ABS ratios are concerned, ours differs from them in several respects. First of all, it is meant to work in the short term (i.e., per-AP timescales), whereas all the above ones (except [10] and [16]) consider long-term decisions. Long-term decisions can indeed leverage more complex algorithms, but they require *modeling* the aspects of the problem – specifically, forecasting channel qualities and traffic arrivals – and assuming that models hold over long timespans. Our work, instead, relies on *measures* rather than

[3] The switch-on/off of a cellular node takes several tens of seconds, and forces mass re-associations of UEs [19].

models, and employs fast algorithms that run in few milliseconds. Second, almost all the above works (including [16]) investigate how *capacity* or *spectral efficiency* are affected by ABS provisioning. To do so, they make assumptions (typically, full-buffer traffic) that are incompatible with those under which a power-saving scheme makes sense at all. Third, our work takes into account the constraints of the protocol layers of the LTE standard. In a real LTE network, UE reporting is limited to periodic CQIs, and the X2AP clearly defines what information can be shared and transmitted among the coordinated eNBs for ABS management. Only few works from the above list ([25], [26], [16]) do consider communications among the eNBs. Some of them just mention the X2 interface, others define their own (non-standard) signaling architecture for ABS provisioning. Only two works use known simulators to test their results, namely [27] uses the Vienna Simulator [28], whereas [29] uses Lte-Sim [30]. All the other use ad-hoc system-level simulators, often purported to be "3GPP-compliant" as they rely on [31] for modeling physical-layer aspects. None of them provide any detail as to how, if at all, what is *above* the PHY (notably resource allocation at the MAC level or inter-eNB communication) is modeled. Finally, all the above works consider either I- or LP-ABSs, but not both simultaneously, and none discuss optimal SF placement.

4. SYSTEM MODEL

This section details the hypotheses and assumptions underlying our work. We focus on a HetNet composed of a macro node, acting as a master for one or more micro nodes, with which it performs dynamic eICIC using ABSs as a tool. All nodes manage SFs of B RBs.

UEs may be associated to either the macro or micro nodes (but not to both). Such association – which changes at timescales much larger than those where our framework operates – is an input datum, and the means by which it is obtained (e.g., selection of a suitable CRE bias) are outside the scope of this paper. We assume that the eNBs (both macro and micros) can classify their UEs into *inner* and *outer*: the former are those that can hear their eNB sufficiently well even when interference from the other HetNet nodes is considered, and the latter are those that cannot, hence must be protected. The criteria according to which this is done are outside the scope of this paper: for instance, a micro eNB may classify its UEs based on their attenuation or channel reporting (UEs reporting a high average CQI both during ABSs and non-ABSs will be considered as *inner*). Alternatively, outer UEs at a micro may be those that would not have associated to it, if it were not for the CRE bias. Dually, a macro eNB may classify as *inner* those UEs that have a smaller attenuation, or that report a high average CQI during both non-ABSs and LP-ABSs. Other schemes can be found in [5], [10]. Although UEs are mobile, we assume that their being *inner* or *outer* does not change within an AP. This is reasonable, since an AP is few milliseconds long (40 ms in an FDD deployment, according to the standard). Each node schedules its UEs on each TTI using its own scheduling algorithm. Since per-TTI scheduling works at a faster pace than our framework’s, we deliberately abstain from placing any requirement on scheduling algorithms (nor we require that they be the same at all nodes in the HetNet), other than that they must be able to inhibit the scheduling of *outer* UEs when required.

We assume that a (macro or micro) node has *three* different regimes, in which it can do different things, consuming different amounts of power.

- *idle*, in which case it can *only* transmit pilot signals, and cannot transmit data traffic. Its power consumption p in the *idle* state is a constant value $p = \pi$.
- *active*, operating at either *low power* or *full power*. An active node can transmit data traffic, and its power consumption is a function of the number of transmitted RBs $p = f_\tau(n)$, where $0 \le n \le B$ and $\tau \in \{low, full\}$.

Our assumptions on the power regimes are the following:

1. $\pi \le f_\tau(0)$, i.e., A node always consumes less when idle than when it is active and transmitting no RBs;
2. The power consumption of an active node (in both low-power and full-power regimes) is a *monotonically increasing affine* function of the number of RBs.
3. $f_{low}(n) \le f_{full}(n)$, $0 \le n \le B$, i.e. the low-power regime consumes less than the full-power one for the same number of RBs.

Assumption 1) implies that it is beneficial to have a node be *idle* when there is nothing to transmit. Besides the obvious point of monotonicity, 2) implies that – from a power-saving standpoint – once a node decides to be active in a SF, the optimal thing to do is to allocate as many RBs as required, given the traffic demand: in fact, the marginal power cost of an RB is a wide-sense decreasing function. This implies that a work-conserving per-TTI scheduler can be used. Finally, 3) is a self-evident consistency condition on the two power regimes of active nodes. Power models that verify all the above conditions are those used in [1], [32]-[34], also shown in Figure 4. As we show later on, the only relevant data at each node, however, are the *maximum* power it consumes in active full-power and low-power SFs, call them $\Pi, \overline{\Pi}$, and its power consumption in the *idle* state π , all of which are easily measurable. The actual values of functions $f_\tau(\ )$ will depend on the *node*, hence will differ for a macro and a micro, and may also be different for micros in the same HetNet.

We assume that nodes communicate through the X2 interface, using *only* the standard signaling described in Section 2. Aside from that, nodes only possess *local* knowledge, i.e. a macro node is *not* aware of its micros’ position or power model, or the number, inner/outer classification and required load of micros’ UEs, etc.

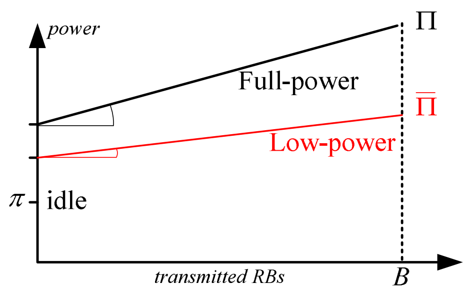

Figure 4 – Example of a power model for a node.

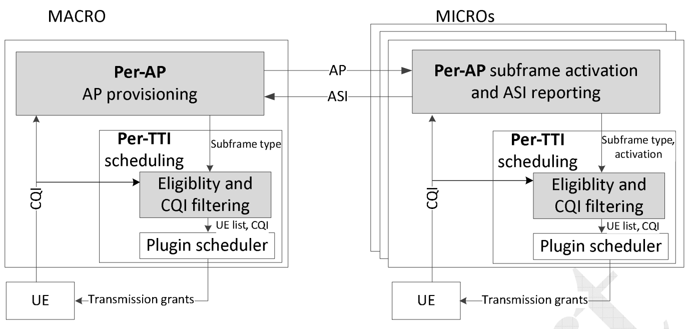

Figure 5 - High-level representation of the proposed framework.

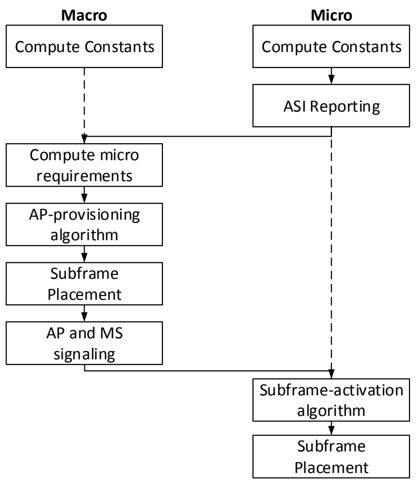

Figure 6 - Main operations of the proposed framework.

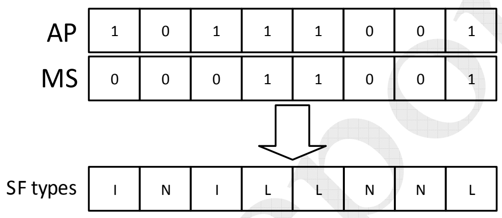

Figure 7 - Example of subframe types in an AP.

TABLE 2 – TYPES OF SUBFRAMES, ELIGIBLE UES AND POWER REGIMES

Node		Non-ABS	I-ABS	LP-ABS
Macro	Eligible UEs	Outer/inner	-	Inner only
	Power regime	Full-power	idle	Low-power
Micro (active)	Eligible UEs	Inner only	Outer/inner	Outer/inner
	Power regime	Low-power	Full-power	Full-power
Micro (idle)	Eligible UEs	-	-	-
	Power regime	idle	idle	idle

5. Distributed Power-saving Framework

This section details our power saving framework (PSF). We first provide a high-level view of PSF and its main operation, sketched respectively in Figure 5 and Figure 6, and then present each block in detail. PSF operates at both the macro and the micros, and on two different timescales: the *AP timescale*, and the *TTI timescale*. At the AP timescale, the macro collects the ASI reports from the micros, computes a new AP and transmits it to the micros. The micros, in turn, obtain the new AP. Based on the information reported therein and on a forecast of their own traffic, they select *which SFs to be active in*, and which to be idle in instead. Finally, at the end of the AP, micros prepare the ASI for the macro. At the TTI timescale, both the macro and the micro select *which subset* of UEs to schedule, according to the settings of the ongoing AP.

Assume that an AP consists of T consecutive SFs, and that $AP[j]$, $MS[j]$ are the bits in the AP and MS vectors related to SF *j*. The macro – and, specifically, its *AP provisioning* (APP) algorithm – periodically selects SFs to be I-ABS, LP-ABS or non-ABS, and communicates that decision to the micros. The three possible SF types are identified by different combinations of the bits in the AP and MS vectors: if $AP[j]=1$, then $MS[j]=0$ will identify that SF as an I-ABS, and $MS[j]=1$

will identify it as an LP-ABS. If, on the other hand, $AP[j]=0$, then the SF will be a non-ABS one. An example of the above combinations with the resulting SF types is shown in Figure 7.

The macro will transmit at its full-power regime to both *inner* and *outer* UEs during non-ABS. It will instead transmit data at a *low-power* regime in LP-ABSs, to its *inner* UEs only. Finally, it will be *idle* in I-ABSs.

On the other hand, the micros will serve both inner and outer UEs during ABSs (both I- and LP-ABSs), using their full-power regime, and serve only inner UEs during non-ABSs, using their low-power regime to reduce the interference made on the macro's outer UEs. Moreover, a micro may decide to be *idle* in some SFs (of any type), if it does not need all the capacity, so as to reduce its own power consumption. The different combinations are summarized in Table 2.

The APP at the macro selects the type of each SF in the AP so as to minimize the macro's power consumption, provided that the load of *both* the macro *and* the micros is carried, if this is possible at all. Following Figure 4 and Table 2, the macro should declare *as few non-ABSs as possible*, e.g., just enough to carry the load of its own *outer* UEs. The rest of the AP should then consist of – possibly – some LP-ABSs, if there are still *inner* UEs that could not be served during non-ABSs, and then as many I-ABSs as possible, to reap the energy benefits of going idle. By doing this, in turn, the macro creates an ideal environment for the micros, which in turn see an AP with plenty of ABSs, at least when the network load is low, and can exploit them to serve their own UEs at reduced interference. This paves the way to further power saving opportunities *at the micros* as well: micros may in fact decide to go idle themselves at some SFs, *both* ABSs and non-ABSs, if they do not need the available downlink capacity. Mirroring what happens at the macro, *mutatis mutandis*, a micro should stay active as little as necessary to carry its load, and select the optimal combination of SFs where it is active, based on its inner/outer UE load and distribution, and on the interference suffered by the macro.

UE scheduling at both the macro and the micros will be done by a per-TTI scheduler. The latter can be seen as a plug-in for our framework, to which an *eligibility* module will feed the list of eligible UEs, depending on the type of SF as per Table 2. The eligibility module should also present the scheduler with the most recent CQIs reported by each UE *in the same type of SF as the current one*, since the interference measured by the UEs – hence their CQIs – will be different depending on the type of SFs. For instance, the CQIs reported by UEs attached to the micro would typically be much lower in non-ABSs than in I-ABSs.

Hereafter, we first describe the algorithms run at the micro, and then move to describing those at the macro.

1. Algorithms run at the micro

All the micros run the same algorithms. When the micro receives an AP, it decides which SFs to be active or idle in, according to its *SF activation* (SA) algorithm. This decision is made so as to *carry the micro's load* (if this is possible at all), using *as little power as possible*.

In order to do this, we need to estimate the micro's *expected load*, i.e. the number of bytes that we expect to transmit in the next AP. We do this by measuring what happens in the past AP, and use the measure thus obtained as a forecast. Since we ultimately need to decide whether or not we need to use the RBs of the various SFs in the AP, it is convenient to normalize the expected load to the average per-RB capacity. The latter, however, depends on the *SF type*, which determines the power level of the micro, the interference suffered from the macro, and which UEs are targeted. Let C_N, C_I, C_L be the average per-RB capacity in non-ABS, I-ABS, and LP-ABS, respectively, measured in the past AP by logging the number of transmitted bytes and allocated RBs, and let $\mathbf{X} \equiv \{\mathrm{N,I,L}\}$ be the set of SF types. C_N, C_I, C_L. Define coefficients $\alpha_N = C_N / C_I$, $\alpha_L = C_L / C_I$, i.e., the capacity per-RB *normalized to I-ABSs'*. For symmetry, one may also define $\alpha_I = C_I / C_I = 1$. Since C_N, C_I, C_L will change over time, α_N and α_L will change as well. Thus, all three coefficients are initialized to a default of 1 when the system starts, and they are updated at the end of each AP. If no data is available to update them (e.g., because inner UEs have all been served during ABSs in the last AP), the *last* computed value is carried over to the next AP. α_N and α_L can be expected to be smaller than one. However, this is not necessarily the case, due to several possible reasons, such as: fading peaks that change CQIs considerably in an AP, or the fact that a significant fraction of UEs may appear/disappear within an AP, so that the set over which the measurements are taken changes significantly from one SF to the other. In any case, our algorithms work regardless of the values of α_N and α_L.

The *expected load* of the micro is represented by two values K, K_{inner}, representing the *overall* expected load and the load of *inner-UEs* only, both measured in multiples of C_I. The algorithm that estimates K for the *next* AP is shown in Figure 8, and it computes the number of RBs that would be needed to clear all the micro's backlog in the *ongoing* AP. K is initialized to zero at the start of an AP, and is increased by the number of allocated RBs (including those for retransmissions), times α_x, in every type-*x* SF. At the end of the AP, K is increased by the number of RBs required to clear the residual backlog at the micro. K_{inner} is computed the same way as K, however, only on *inner* UEs.

```
@AP start:
    let K=0
@type-x SF:
    let b=number of allocated RBs in the SF
    let K=K+b*$\alpha_x$
@AP end:
    let K=K+total backlog/$C_I$
```

Figure 8 – Algorithm to compute the micro’s expected load for the next AP.

Subframe activation algorithm

Once the micro’s expected load has been computed, the SA algorithm is quite straightforward. Let T_x, $x \in \mathbf{X}$, be the number of type-*x* SFs in the next AP (provisioned by the macro and identified by the micro via straightforward bitwise operations on the AP and MS vectors received in the X2 message). The SA computes t_x, i.e., how many type-*x* SFs to be *active* in, so as to minimize the micro power consumption. This is done by solving at optimality the following integer-linear problem (ILP):

$$\begin{aligned}
&\min\left\{\sum_{x\in\mathbf{X}} W_x \cdot t_x\right\} \\
&s.t. \\
&\quad t_x \le T_x && x\in\mathbf{X} \quad (i) \\
&\quad t_I + t_L \ge \min\left\{T_{on}, T_I + T_L\right\} && (ii) \\
&\quad t_N \le \left\lceil \frac{K_{inner}}{B\cdot\alpha_N} \right\rceil && (iii) \\
&\quad B\cdot t_N\cdot\alpha_N \le K_{inner} + (1-b)\cdot\Omega && (iv) \\
&\quad B\cdot t_N\cdot\alpha_N + b\cdot\Omega \ge K_{inner} && (v) \\
&\quad B\cdot\sum_{x\in\mathbf{X}} t_x\cdot\alpha_x + (1-b)\cdot\Omega \ge \min\left\{K, B\cdot\sum_{x\in\mathbf{X}} T_x\cdot\alpha_x\right\} && (vi) \\
&\quad B\cdot\left(t_I\cdot\alpha_I + t_L\cdot\alpha_L\right) + b\cdot\Omega \ge \min\left\{K - K_{inner}, B\cdot\left(T_I\cdot\alpha_I + T_L\cdot\alpha_L\right)\right\} && (vii) \\
&\quad t_x \in \mathbb{Z}^+ && x\in\mathbf{X} \quad (viii) \\
&\quad b\in\{0,1\} && (ix)
\end{aligned} \tag{1}$$

The objective function to be minimized is the variable part of the power consumed by the micro. A fixed baseline of $\pi\cdot T$ will be consumed in any case, even if the micro is always idle. For each type-*x* SF when the micro is active, an *additional* power consumption of W_x is required, with $W_L = W_I = \Pi - \pi$, $W_N = \overline{\Pi} - \pi$ (see again Table 2) to have the micro transmit all the B RBs in that SF. Note that the decision on *how many* RBs are actually allocated in a SF is *not* taken by the SA, nor could it reasonably be: rather, it is taken on a per-TTI basis by the scheduler, which is the only one that knows the *actual* traffic demand at that TTI. Thus, the value of the objective function, plus the constant term $\pi\cdot T$, is an *upper bound* on the overall power that will be consumed by the micro in the next AP, and as tight as an upper bound can be unless clairvoyance of future loads and CQIs is assumed. Constraint (*i*) models the upper bound on the number of active SFs of each type given by the AP provisioning. Constraint (*ii*) guarantees that the micro is active for a minimum number of SFs $T_{on} \geq 0$ to serve both inner and outer UEs, if this is compatible with the AP decisions made at the macro (hence the *min* on the right-hand side). This may help a network engineer to strike a better trade-off between energy consumption and latency. In fact, an AP is typically long (i.e., 40ms), and if a micro has no traffic during one AP, then its expected load would be null for the next AP as well, hence the SA would set the micro to be idle for the whole *next* AP. Thus, any downlink traffic arriving in that AP would be neglected until the successive AP (when it would be counted in K, and possibly K_{inner}, as remaining backlog). By setting T_{on} to a non-null value, one guarantees that enough active SFs always occur in an AP, hence the delay is reduced at the expenses of a higher power consumption. Constraint (*iii*) states that, since you can only serve inner UEs in non-ABSs, these must not exceed the number necessary to carry expected load K_{inner}. Constraints (*iv-vii*) state that the overall normalized capacity must be sufficient to carry the expected load K, if it can be carried at all. The easiest way to explain these constraints is to understand that (*iii*) allows that $B \cdot t_N \cdot \alpha_N > K_{inner}$ (as per the ceiling operator), but this does not mean that the leftover capacity $B \cdot t_N \cdot \alpha_N - K_{inner}$ can be made available to *outer* UEs, which are ineligible in non-ABSs. Therefore, we must discriminate whether $B \cdot t_N \cdot \alpha_N \leq K_{inner}$ or $B \cdot t_N \cdot \alpha_N \geq K_{inner}$, and we do this by employing a helper binary variable *b*, such that $b = 1 \Rightarrow B \cdot t_N \cdot \alpha_N \leq K_{inner}$, and $b = 0 \Rightarrow B \cdot t_N \cdot \alpha_N \geq K_{inner}$. The above two implications are written in constraints (*iv-v*), by multiplying alternatively *b* or 1-*b* by a large positive constant Ω, so that either constraint is inactive depending on the value of *b*. Now, if *b*=1 (i.e., the capacity in non-ABSs does not exceed the requests of inner UEs), then the total capacity in active SFs is sufficient if $B \cdot \sum_{x \in \mathbf{X}} t_x \cdot \alpha_x \geq K$. However, it may be that K exceeds the allocable capacity $B \cdot \sum_{x \in \mathbf{X}} T_x \cdot \alpha_x$, hence that constraint must be reformulated as (*vi*). On the other hand, if *b*=0 (i.e., the capacity in non-ABSs may exceed the request of inner UEs), then we only need to check if there are enough active LP-ABSs and I-ABSs to carry the expected load of *outer* UEs, if that load can be carried at all. This is written in constraint (*vii*). Note again that either (*vi*) or (*vii*) is inactive, depending on the value of *b*.

Problem (1) is an ILP with four variables and nine constraints, which is always feasible. ILPs are NP-hard in general (except those whose matrix is totally unimodular, which is not the case here). This means that the best known algorithms (such as *branch-and-cut* [41]) have a worst-case complexity which is exponential in the number of variables. In our case, the number of variables is constant with respect to the HetNet parameters *and* small. We prove the following result in the Appendix.

Lemma 1: *Problem* (1) *admits no more than* $2 \cdot (T/3+1)^3$ *solutions.*

The above number is independent of the number of UEs, their traffic, and the system bandwidth (i.e., constant B). Since the number of *constraints* is independent of all the above as well, the complexity of solving (1) at optimality is $O\left(T^3\right)$. In fact, a brute-force algorithm would just find the optimum by testing all constraints on all the feasible solutions. Note that, since $T = 40$, there are fewer than 5890 solutions, hence that would be something that a modern CPU could do in a matter of milliseconds.

Once problem (1) is solved, a micro knows *how many* SFs of each type to be active in, but it must still decide *which* SFs to activate. This apparently simple problem is in fact non-trivial, hence we postpone addressing it until Section 5.3, i.e., after we present the algorithms for the macro, to avoid disrupting the flow of the discussion.

UEs attached to the micros and related CQIs

		CQI		
UE	category	I-ABS	LP-ABS	Non-ABS
A	inner	15	13	5
B	inner	14	11	6
C	outer	9	6	-
D	outer	8	4	-
…	…	…	…	…

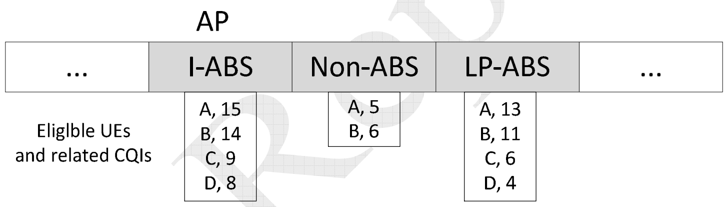

Figure 9 – Eligible UEs per SF type.

Eligibility

The *eligibility* module at the micro runs at every TTI and presents the scheduler with the list of eligible UEs for the current SF, according to Table 2. Moreover, it stores the CQIs of the UEs *together with the SF type* they are related to. When presenting the scheduler with the list of eligible UEs for a type-*x* SF, it also communicates to the scheduler the most recent CQI of that UE *for type-x SFs*, as shown in Figure 9. CQIs can be expected to be considerably different for the same UE in different SF types (this is, in fact, the main reason for the whole ABS mechanism in the first place). Hence, using a CQI measured in a different type of SF would lead to either H-ARQ errors or resource underutilization, depending on whether you overestimate or underestimate it.

ASI reporting

At the end of an AP, the micro reports two values, namely the AS and UAP. According to the standard, the UAP should report the number of ABSs where the interference is small enough[4]. We embody this concept by measuring the average per-RB capacity: this can be expected to be smaller in LP-ABSs than in I-ABSs, due to interference from the macro, hence all it takes is to compare the per-RB capacity of an LP-ABS against a threshold $\sigma \cdot C_I$ to determine whether that SF will

[4] Note that the standard does not provide any quantitative definition.

contribute to the UAP or not. We set the threshold at $\sigma = 0.7$, and we justify later on that σ plays a limited role in our framework, thanks to the way the AP provisioning is formulated. Thus, ASI reporting boils down to the self-explanatory algorithm of Figure 10.

```
@AP start:
   set UAP=$T_L + T_I$; set R=0
@ABS SF:
   measure per-RB capacity C
   let b=number of allocated RBs in SF
   if (SF is LP-ABS and C<$\sigma \cdot C_I$)
      decrease UAP
   else
      let R=R+b
@AP end:
   report UAP
   if (UAP)>0) report AS=R/UAP
   else report AS=0
```

Figure 11 – ASI reporting at the micros.

```
for each micro node i
      let req=ceiling(UAP[i]*AS[i]/B)
      if (AS[i]=100) increase req
      if (UAP[i]<$T_L + T_I$)
         let $\tau_I$[i]=req
         let $\tau_L$[i]=0
      else //(UAP[i]==$T_L + T_I$)
         let $\tau_I$[i]=0
         let $\tau_L$[i]=req
let $\tau_I$=max_i{$\tau_I$[i]}
let $\tau_L$=min{max_i{$\tau_L$[i]}-$\tau_I$,0}
```

Figure 10 – Computing the minimum requirements τ_I, τ_L at the macro.

Note that the per-ABS part of the algorithm is run also when the micro is idle, in which case no RBs are allocated. The ASI reporting algorithm is coherent with the fact that the macro advertises LP-ABSs to the micros by setting the relevant bit in the MS vector: this means instructing the micros to measure the interference in those SFs.

2. Algorithms at the macro

This section details the algorithms being run at the macro. Since many aspects of these are similar, and specular, to those of the micros, we will reuse the same notation whenever possible, there being no ambiguity as to the fact that we are dealing with quantities related to the macro instead.

In order to provision an AP, the macro needs to gather the ASI reports from its micros, based on which it can infer their requirements, and to compute its own expected load as well. Within a micro’s ASI, two data are relevant:

- whether AS=100 or not: in fact, AS=100 denotes a *possible* overload situation, whereas AS<100 indicates that there are unused RBs in active SFs. We say “possible” since it may be that the micro node is actually able to carry its load (e.g., by serving inner UEs during non-ABS frames). However, the standard X2-based signaling does not allow to discriminate the two cases, hence we must assume a worst-case scenario.
- whether or not UAP is equal to the number of ABSs in the previous period $T_L + T_I$. If it is not, then the macro should prefer I-ABS to LP-ABS for that micro, given the choice, since presumably that micro suffers too much interference in the latter.

The cross-product of the above yields four different combinations, described below, together with the inferences that the macro should reasonably make on the micro’s requirements as a consequence – assuming that that micro’s load in the next period stays the same:

a) (UAP == $T_L + T_I$) and (AS<100):

- tolerable interference in LP-ABSs;
- the available ABSs (both I- and LP-) are enough to carry the micro’s load.

b) (UAP == $T_L + T_I$) and (AS==100):

- tolerable interference in LP-ABSs;
- not enough ABSs to carry the current load in the micro. This micro will fare better if the number of either LP-ABSs or I-ABSs is increased in the next AP.

c) (UAP< $T_L + T_I$) and (AS<100):

- intolerable interference in LP-ABSs;
- the available I-ABSs are enough to carry the current load in that micro, hence that micro does not need to use LP-ABSs.

d) (UAP< $T_L + T_I$) and (AS==100):

- intolerable interference in LP-ABSs;
- There is not enough capacity in ABSs to carry that micro’s load. This micro will fare better *only* if the number of *I-ABSs* is increased in the next AP.

Based on the above inferences, the macro computes two values τ_I, τ_L, corresponding to the *minimum* number of I-ABS and LP-ABSs necessary to satisfy the requirements of all the micros simultaneously, using the algorithm in Figure 11.

For each micro *i,* the algorithm computes a required number of ABSs (line 2) based only on those where the interference is tolerable (thus, erring on the safe side). That number is increased by one if the micro signals overload (line 3). That number is then assigned to either τ_L `[i]` or τ_I `[i]` based on whether the interference in LP-ABSs is tolerable or not (lines 4-9). Finally (line 10), the minimum number of I-ABSs τ_I is computed as the maximum of τ_I `[i]` among all the micros. If a micro *j* requested a higher number of ABSs than τ_I, and tolerates that they are LP-ABSs, then it is τ_L `[j]`$> \tau_I$, hence the maximum difference τ_L `[j]`$- \tau_I$, if positive, should make up τ_L (line 11). This way, some of micro *j*’s ABS will be I-ABSs in any case, which can only improve its performance. Figure 12 shows an example of computation for τ_I and τ_L, obtained from four ASI reports sent by as many micros. Note that all the above four cases are covered.

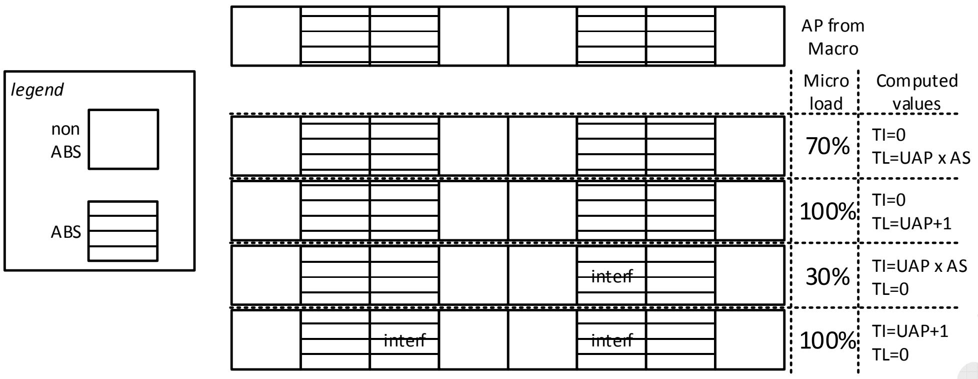

Figure 12 - Micro feedback to the macro node

The macro also needs to compute its own expected load, in the same way as micros do, by measuring its own K, K_{inner} using a similar algorithm as the one in Figure 8. The macro computes its expected load as a multiple of C_N, its average per-RB capacity in non-ABSs, and RBs allocated in LP-ABSs are rescaled by coefficient $\alpha_L = C_L/C_N$. We leave to the alert reader the straightforward adaptation of the algorithm in Figure 8 to compute K, K_{inner} at the macro.

AP provisioning

The AP provisioning at the macro consists in selecting how many SFs of each type will occur in the next AP, i.e., selecting T_I, T_L, T_N such that $\sum_{x\in\mathbf{X}} T_x = T$. The following inequalities constrain the choice:

- *micro requirements*: in order to meet them, it must be $T_I \geq \tau_I$ *and* $T_I + T_L \geq \tau_I + \tau_L$. This last constraint allows LP-ABSs to be upgraded to I-ABSs, if the macro has no need for them.
- *maximum latency constraints*: $T_N \geq T_{on}$. Similarly to what we do at the micros, we need to guarantee that the macro is always able to dedicate some capacity to its *outer* UEs, even if it was completely unloaded in the previous AP. Hence, at least T_{on} non-ABSs should be provisioned in an AP.
- *macro capacity requirements*: the overall capacity at the macro must be sufficient to carry its own expected load. This requires that $T_L \leq \lceil K_{inner}/(B\cdot\alpha_L)\rceil$, and that $T_N\cdot B + \min\{T_L\cdot B\cdot\alpha_L, K_{inner}\} \geq K$. The "min" at the left-hand side of the last inequality is necessary because T_L may be larger than strictly necessary to serve all the *inner* UEs, as per the ceiling in the first inequality, but this does not mean that any leftover capacity can be made available for *outer* UEs in LP-ABSs (similar to problem (1) at the micro).

This said, the power-optimal AP provisioning at the macro is the optimum of the following ILP:

$$\begin{aligned}
&\min\left(W_L \cdot T_L + W_N \cdot T_N\right) && && \\
&s.t. && && \\
&\quad \sum_{x\in\mathbf{X}} T_x = T && && (i) \\
&\quad T_I \geq \tau_I && && (ii) \\
&\quad T_I + T_L \geq \tau_I + \tau_L && && (iii) \\
&\quad T_N \geq T_{on} && && (iv) \\
&\quad T_L \leq \left\lceil \frac{K_{inner}}{B \cdot \alpha_L} \right\rceil && && (v) \\
&\quad B \cdot T_L \cdot \alpha_L \leq K_{inner} + (1-b)\cdot\Omega && && (vi) \\
&\quad B \cdot T_L \cdot \alpha_L + b\cdot\Omega \geq K_{inner} && && (vii) \\
&\quad B \cdot \left(T_N + T_L \cdot \alpha_L\right) + (1-b)\cdot\Omega \geq K && && (viii) \\
&\quad B \cdot T_N + b\cdot\Omega \geq K - K_{inner} && && (ix) \\
&\quad T_x \in \mathbb{Z}^+ && x \in \mathbf{X} && (x) \\
&\quad b \in \{0,1\} && && (xi)
\end{aligned} \qquad (2)$$

Problem (2) is indeed similar to (1). Its objective function includes only the extra power consumed in LP-ABSs and non-ABSs (the macro is in fact idle in I-ABSs), using weights W_x defined as those in the objective of (1) and obtained from the macro’s power model. We remark again that the value of the objective (plus a constant term $\pi \cdot T$) is an upper bound on the power that the macro will consume in the next AP. Constraint (*i*) bounds the length of the AP. Constraints (*ii-iii*) are the micro requirements, (*iv*) is the maximum latency requirement, and (*v-ix*) are the macro requirements, rewritten as linear constraints with the help of binary helper variable b as in (1). The problem is an ILP with four variables and nine constraints[5]. We prove the following in the Appendix:

Lemma 2: *Problem* (2) *admits no more than* $(T+1)\cdot(T+2)$ *feasible solutions.*

Since the number of constraints is constant, solving (2) at optimality is $O\left(T^2\right)$. Moreover, since $T = 40$, there are fewer than 1722 solutions, hence even a brute-force approach would find the optimum in a matter of milliseconds on a modern CPU.

However, unlike (1) problem (2) may actually be *infeasible*. This may occur for three reasons:

1) $T_{on} > T - \left(\tau_I + \tau_L\right)$, i.e. the micro requirements leave too few non-ABSs available for the maximum latency constraint to be met. This follows from (*i*), (*iii*) and (*iv*).
2) $K - K_{inner} > \left[T - \left(\tau_I + \tau_L\right)\right]\cdot B$, i.e., the maximum number of non-ABSs (i.e., the term between square brackets) is not sufficient to carry $K_{out} = K - K_{inner}$, i.e. the expected load of *outer* UEs at the macro.

[5] The alert reader will notice that one T_x variable is redundant, because constraint (*i*) is an equality. We left it in the model nonetheless for the sake of readability.

3) $K > \left[T - \left(\tau_L + \tau_I\right)\right] \cdot B + \alpha_L \cdot \tau_L \cdot B$, i.e., the expected load at the macro is larger than its total exploitable capacity, in both non-ABSs (i.e. the first term at the right-hand side) and LP-ABSs (second term).

Since SF selection affects the performance of the whole network, such *overload* situations must be managed by rescaling the input constants K, K_{inner}, τ_I, τ_L, so that all the constraints become feasible. This can be done by scaling them proportionally, as follows:

1. If $T_{on} > T - \left(\tau_I + \tau_L\right)$ then compute the new $\hat{\tau}_I$ and $\hat{\tau}_L$ as

$$\hat{y} = \frac{T - T_{on}}{\tau_I + \tau_L} \cdot y \text{, with } y \in \{\tau_I, \tau_L\} \text{;}$$

2. If $K - K_{inner} > \left[T - \left(\tau_I + \tau_L\right)\right] \cdot B$ then compute the new $\hat{\tau}_I$, $\hat{\tau}_L$ and $\hat{K}$ as:

$$\hat{K} = \left(T \cdot \frac{\min\left(T, K_{out}/B\right) \times B}{\min\left(T, K_{out}/B\right) + \tau_I + \tau_L} \right) + K_{inner}, \; \hat{y} = \frac{T}{\min\left(T, K_{out}/B\right) + \tau_I + \tau_L} \cdot y \text{, with } y \in \{\tau_I, \tau_L\} \text{;}$$

3. If $K > \left[T - \left(\tau_L + \tau_I\right)\right] \cdot B + \alpha_L \cdot \tau_L \cdot B$ then compute the new $\hat{K}, \hat{K}_{inner}$, $\hat{\tau}_I, \hat{\tau}_L$ as:

$$\hat{K}_{inner} = \left(T - K_{out}/B\right) \cdot \frac{\min\left(T, K_{inner}/B\right) \cdot B}{\min\left(T, K_{inner}/B\right) + \tau_I + \tau_L\left(1 - \alpha_L\right)}, \quad \hat{K} = \left(K - K_{inner}\right) + \hat{K}_{inner},$$

$$\hat{\tau}_L = \frac{\left(T - K_{out}/B\right)}{1 - \alpha} \times \frac{\tau_L\left(1 - \alpha\right)}{\min\left(T, K_{inner}/B\right) + \tau_I + \tau_L\left(1 - \alpha_L\right)}, \; \hat{\tau}_I = \left(T - K_{out}/B\right) \times \frac{\tau_I}{\min\left(T, K_{inner}/B\right) + \tau_I + \tau_L\left(1 - \alpha_L\right)}.$$

Note that the way constraints (ii-iii) are written implies that the ASI reporting threshold σ plays a very limited role in our scheme. In fact, it only serves the purpose of discriminating between case b) and d) above, when the micro has a high load (ASI=100). A higher σ will tend towards d), limiting the possibility of using LP-ABSs. At low loads i.e., when $\tau_L + \tau_I + \lceil K/B \rceil$ is well below *T*, the macro will select $T_I \geq \tau_L + \tau_I$ regardless of the value of τ_L, hence of the threshold σ, since this is more favorable from an energy and capacity standpoint. If the macro ends up using LP-ABSs at all at low loads, it will be because it is more favorable, power-wise, to serve its own *inner* UEs using LP-ABSs instead of non-ABSs, and not because of the micro requirements.

We terminate this section by mentioning that the eligibility module at the macro works the same as the one for the micros, *mutatis mutandis* according to Table 2.

3. The problem of optimal subframe placement

The macro must *place* all types of SFs - whose numbers it has just computed by solving (2) - within the AP. Similarly, each micro must select which SFs to be active in, given the AP provisioned by the macro and the numbers obtained by solving (1). The two problems share the same objective, i.e. to enable the relevant node to serve its users *as often as possible*, so as to minimize the maximum latency: with reference to Figure 13, where a repeating AP of 10 TTIs is assumed for ease of reading, the topmost placing is such that an *outer* macro UE will have to wait up to seven TTIs before being served (i.e., if its traffic arrives at time 3, it will have to wait until time 10 for a non-ABS), whereas the maximum latency for an *inner* UE will be five TTIs (i.e., for traffic arriving at time 5). On the other hand, if the same number of SFs are arranged as in the bottom AP, the above latencies are reduced to three and two, respectively. To the best of our knowledge, no paper dealing with ABS describes *how to place* SFs in the AP.

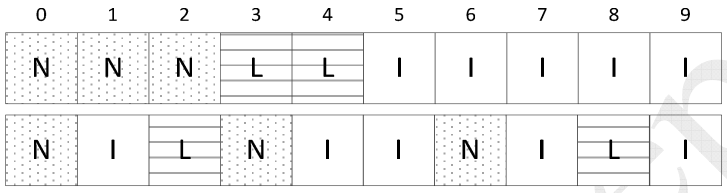

Figure 13 – Two different subframe placements in an AP.

The algorithm for optimal placing at the macro is intuitively straightforward, but slightly tricky to formalize. Assume that the SFs in the AP are numbered from 0 to $T-1$. First, all the *non-ABSs* are placed so as to minimize the *maximum distance* between consecutive ones (keeping into account wrap-around at the end of the AP). This guarantees that *outer* UEs can be served as frequently as possible, and is achieved by placing non-ABSs at positions $\lfloor i \cdot T/T_N \rfloor$, with *i* ranging from 0 to $T_N - 1$. This is what happens at the bottom of Figure 13, where $T_N = 3$ is assumed, and leaves the AP with T_N disjoint intervals (i.e., 1-2, 4-5, 7-9). Some of these intervals ($T \bmod T_N$, in fact) have a length $\lceil T/T_N \rceil - 1$, whereas the remaining ones include $\lfloor T/T_N \rfloor - 1$ SFs, i.e. one less than the former, unless T is an integer multiple of T_N. In our example, there is one "longer" interval (7-9), and two "shorter" ones (1-2 and 4-5).

Then, all the *LP-ABSs* are placed in these intervals so as to minimize the maximum distance between consecutive *active* SFs (whether non-ABSs or LP-ABSs) at the macro, thus increasing the frequency of transmission opportunities for *inner* UEs. This is done by assigning either $\lfloor T_L/T_N \rfloor$ or $\lfloor T_L/T_N \rfloor + 1$ LP-ABSs to the intervals, taking care to assign *more* LPs to larger intervals first, and splitting each interval evenly. In the example of Figure 13, we need to place two LP-ABSs, hence we will have either zero or one LP-ABS per interval. The only "larger" interval (7-9) will get one LP-ABS, and the remaining one will go to the first of the two smallest intervals, i.e., 1-2.

The pseudocode for the placement algorithm is reported in Figure 14. Lines 2-5 compute all the involved constants (i.e., how many long and short intervals, and how many of these will get one more LP-ABS than the other intervals of the same length). The outer *for* cycle assigns non-ABSs, and the inner *for* cycle assign LP-ABSs within an interval. The alert reader will notice that the placement algorithm at the macro is $O(T)$.

At the micro, the added complication is that the t_I, t_L and t_N activations selected by the SA algorithm can occur only in SFs of matching type, whose positions are *fixed*. Finding the “most regular spacing” of activations in these settings is challenging – to the point that even formally defining what a “most regular spacing” should be is all but trivial. Moreover, it would involve solving at optimality an ILP with $O(T)$ variables, which would clearly be impractical. Since what really matters is to avoid *clustering* activations in the same region of the AP (thus unduly leaving the micro idle for too many consecutive SFs), we use a simple heuristic that leverages the even spacing already done by the macro. For each SF *type* x, we activate the $\lfloor i \cdot T_x / t_x \rfloor$-th SF *of that type*, with *i* ranging from 0 to $t_x - 1$. To reduce the chance of activations of adjacent SFs of different types, the AP vector is scanned circularly starting from a *different* offset for each SF type, e.g., $r, \left| r + \lfloor T/3 \rfloor \right|_{\text{mod}\,T}, \left| r + \lfloor 2 \cdot T/3 \rfloor \right|_{\text{mod}\,T}$, if all three t_I, t_L and t_N are non null, where r is a random offset taken uniformly in $[0,T[$ at each AP. Value r is used to desynchronize activations among micros, which further reduces the interference at low loads. The placement algorithm at the micro is $O(T)$ as well.

```
init: all SFs are I-ABS
let q=$\lfloor T_L/T_N \rfloor$, r=$T_L \bmod T_N$
let NLongIntervals=$T \bmod T_N$, NShortIntervals=$T_N-(T \bmod T_N)$
let OneMoreLPLong=min(r,NLongIntervals)
let OneMoreLPShort=min(r-OneMoreLPLong,NShortIntervals)
for i=0 to $T_N-1$
    mark SF $\lfloor i \cdot T/T_N \rfloor$ as non-ABS
    let NLPs=q
    let $I=\lfloor (i+1) \cdot T/T_N \rfloor - \lfloor i \cdot T/T_N \rfloor - 1$// interval length
  let $O=\lfloor i \cdot T/T_N \rfloor + 1$            // interval starting offset
  if (I == $\lceil T/T_N \rceil - 1$) and (OneMoreLPLong>0)
    let NLPs=q+1
    let OneMoreLPLong=OneMoreLPLong-1
  if (I == $\lfloor T/T_N \rfloor - 1$) and (OneMoreLPShort>0)
    let NLPs=q+1
    let OneMoreLPShort=OneMoreLPShort-1
  for j = 1 to NLPs
    mark SF $O+\lfloor j \cdot I/(NLPs+1) \rfloor$ as LP-ABS
```

Figure 14 – Subframe placement at the macro.

4. Communication and computation overhead and reporting periods

As shown in the previous subsections, computing a new AP involves exchanging messages and running algorithms at both the macro and the micros. The whole timing is shown in Figure 15. Algorithms at the macro and the micros cannot run in parallel, since the latter take the output of the former as an input, and – simple and fast as they may be – cannot be supposed to run in zero time. Assume that AP $j+1$ starts at $(j+1)\cdot T$, and call x the *reporting offset*, i.e. the time it takes for our framework to prepare a new AP. The computations for AP $j+1$ must start by $(j+1)\cdot T-x$, hence must make reference to the most recent AP completed by then (i.e., the $j-1$-th in the figure, since the j-th is ongoing).

However, the communication and computation overhead of our framework is considerably shorter than one AP, as we show later on. Thus, one may ask if having fresher, more recent reports from the micros may be beneficial, and how this can be accomplished. All it takes is to interpret the standard liberally, and assume that an AP means a *generic* period of T consecutive SFs with an arbitrary offset. Define a *reporting period* $\left[j\cdot T-x,(j+1)\cdot T-x\right)$ (consisting of T SFs), and assume that the micros make their report based on *reporting periods* instead of APs. This would allow a new AP to be computed using the most recent possible traffic, CQI and interference estimates. This setting generalizes the standard, which can be seen as the particular case $x=T$. For this to happen, the macro and its micros would then need to agree on the value of x, using extra (non-standard) X2 signaling.

Note that the standard does not specify the *period* at which APs can be modified. The latter must be a multiple of the AP, and we call it *reconfiguration period* henceforth.

As far as computational overhead is concerned, we remark that the two ILPs (1) and (2) have *constant worst-case solution times*, i.e., independent of the number of micros, number of UEs, their traffic, the system bandwidth *B*. The algorithms for computing a node’s load are $O(T)$, and the algorithm to collate micro’s requirements at the macro (the one in Figure 11) is $O(n)$, *n* being the number of micros. Finally, the two SF placing algorithms at the macro and the micro are $O(T)$ as well. As far as communication overhead is concerned, the traffic flowing along the X2 is 10+6*n* bytes per AP, hence $O(n)$, which is what the standard prescribes. No per-UE information are sent through the X2, unlike with [10] or [16].

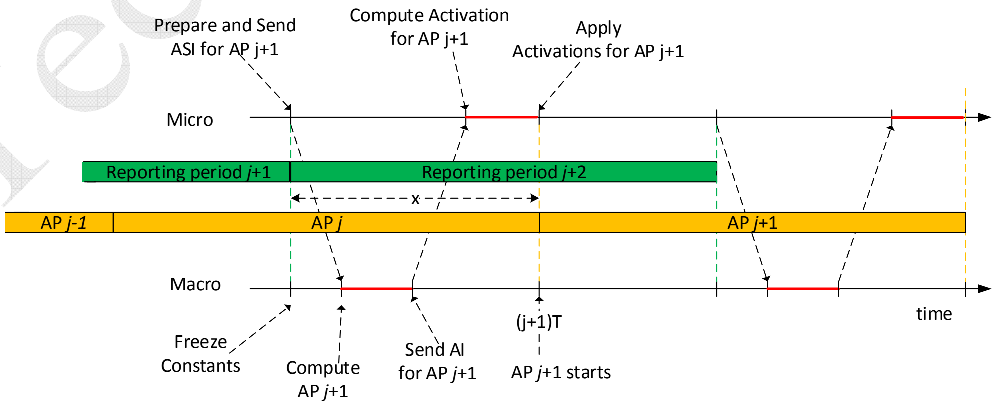

Figure 15 - Message exchange for the preparation of a new AP.

6. PERFORMANCE EVALUATION

In this section, we evaluate the performance of PSF. Our evaluation is carried out using SimuLTE [36]-[37], a system-level simulator developed for the OMNeT++ simulation framework [38]. SimuLTE simulates the data plane of the LTE/LTE-A radio access network. The SimuLTE protocol stack includes all the LTE protocol layers, i.e., a Packet Data Convergence Protocol – Radio Resource Control (PDCP-RRC), Radio Link Control (RLC), MAC and PHY. It also includes models for *macro*-, *micro*-, *pico*-eNBs, with different radiation profiles (both isotropic and anisotropic), functions for MAC-level scheduling in downlink and uplink directions and X2-based inter-eNB communication. Note that the benefits of having an OMNeT++-based simulator include leveraging the INET library [39], which already includes validated models of well-known Internet protocols (therein including SCTP for the X2, IP, TCP, UDP, etc.).

We extended SimuLTE to support ABSs and ABS signaling over X2, and we included an implementation of the PSF framework. Both the APP and SA problems are built dynamically during the simulation, and solved at runtime using the state-of-the-art solver for ILPs, i.e. CPLEX [35]. CPLEX employs two algorithms to solve ILPs. The first one is called *presolver*, and its job is to reformulate the instance using preprocessing and probing techniques [40], i.e. to remove or fix some variables and/or rewrite constraints so that the problem admits a tighter LP relaxation. In some cases, especially when instances are small, this may succeed in solving an ILP at optimality itself. Otherwise, the presolver will at least pre-pare a better formulation to be passed to the second algorithm, i.e. the solver proper. The latter uses the *branch-and-cut* algorithm [41], an improvement of *branch-and-bound* where new constraints ("cuts") are added to the LP relaxation of the problem whenever the latter has a non-integer optimum. Branch-and-cut is generally faster than branch-and-bound, and can be stopped earlier when a pre-specified *optimality gap* is reached, i.e., when the incumbent integer solution is within a given range of the optimum of the LP relaxation, the latter being a lower bound to the optimum. An optimality gap of 0.5% is set, hence the solutions are guaranteed to be at least 99.5% optimal.

UE classification is performed at the start of the simulation, based on the attenuation to the serving node, with a threshold of 80dB. This allows you to have both inner and outer UEs at the macro and the micro. Only downlink traffic is simulated, and we assume that the traffic is generated by applications running on a server and forwarded by a router to the serving cell of the receiver. Applications alternate between active and inactive states, whose duration is extracted from a Weibull distribution. When active, the application generates fixed-length packets every 20 ms, unless otherwise specified. In our evaluations, we often vary the offered load by varying the packet size, whose default value is 50 bytes. Scheduling at the MAC layer is done according to a MaxCQI policy, i.e., eligible UEs are sorted by decreasing CQI and each one is allocated enough RBs to empty its queue, until either all RBs are occupied or no eligible UEs remain. The propagation and transmissions delays of the X2 are assumed to be negligible, given the small inter-node distance and size of the ABS X2 messages. We measure the end-to-end application delay, the HetNet throughput measured at the MAC layer, and the HetNet power consumption, computed as the sum of the consumption for all the eNBs in the system. For the latter we only display the part that can be actually affected by the ABS decision process, i.e. the one exceeding the baseline π . Confidence intervals at the 95% level are shown only when visible. The main simulation parameters are reported in Table 3, whereas the values used in the power models are described in Table 4, and are taken from [34]. With these values, a macro (micro) in its low-power regime consumes the same power as a micro (pico) at its full-power regime.

TABLE 3 – MAIN SIMULATION PARAMETERS

Parameter	**Value**
Bandwidth	10 MHz – 50 RBS
Carrier Frequency	2 GHz
Path Loss Model	ITU Urban Macro **Errore.**
Fading Model	Jakes
Simulation Time	30 s
Warm-up time	3 s
# Indep. Replicas	5
Packet size (default)	50 bytes
Inter-packet time	20ms
Active period duration	Weibull, mean 0.8s
Silence period duration	Weibull, mean 1.0s

TABLE 4 - POWER MODEL

	Macro		**Micro**	
Scenario	1 Micro	2 Micros	1 Micro	2 Micros
Tx Power (Low)	30 dBm	21 dBm	14 dBm	14 dBm
Tx Power (High)	40 dBm	40 dBm	30 dBm	21 dBm
Antenna pattern	Anisotropic		Anisotropic	
π	174		7	
Π	445.5 W		342.5 W	
$\overline{\Pi}$	38.5 W		21 W	
$f(0)$	320 W		16 W	

1. Benchmarking

We first benchmark the performance of PSF in various configurations, to explore trade-offs between power consumption and UE QoS. We consider a scenario with one macro and one micro node where UEs are dropped uniformly within a circle whose radius is 250 m, and are associated to a serving eNB on a highest-received-power basis, as we show in Figure 16. The default CRE bias is zero. With this deployment, the number of inner UEs is around 5% of the total at the macro, and 25% at the micro. In Figure 17 we show the distribution of the user application delay for three values of T_{on} when the number of UEs ranges from 50 to 300. Each bar represents the interval between the 90th (upper edge) and 10th (lower edge) percentiles. The solid line marks the median, and the two dashed lines mark the 25th and 75th percentiles. From such graphs we can see that, when T_{on} is low, all the percentiles decrease with the load. Although this might seem counterintuitive, it can be easily justified analyzing the number of SF per type, which is shown in Figure 18: the higher the number of UEs, the higher the number of SF where the eNBs will be active, hence the faster it reacts to UEs traffic requests. If we increase the value of T_{on}, delay percentiles decrease globally at low and medium loads, and the number of non-ABSs increases as well. At higher loads, instead, increasing T_{on} is less effective, as eNBs are already active most of the time, but it still reduces the 90th percentile. On the other hand, using larger values of T_{on} increases the power consumption, especially at low loads, as demonstrated by Figure 19. Increasing T_{on}, in fact, increases both the number of non-ABSs and the number of ABSs where the micro is active. By tuning the value of T_{on}, a network operator can trade the power consumption of the HetNet for the UE QoS.

We then show that placing ABSs smartly within an AP brings benefits. We compare our optimal placing against two baselines: one (*On/Off*) where subframes of the same type are clustered, and a second one (*split*) where clustering occurs in the two halves of the AP. An exemplary placement of 10 I-ABSs and 10 LP-ABSs using the three algorithms is shown in Figure 20, whereas in Figure 21 we show the distribution of the application delay for the three placing algorithms with an increasing number of UEs. As we can see, the optimal placing yields lower delays in general.

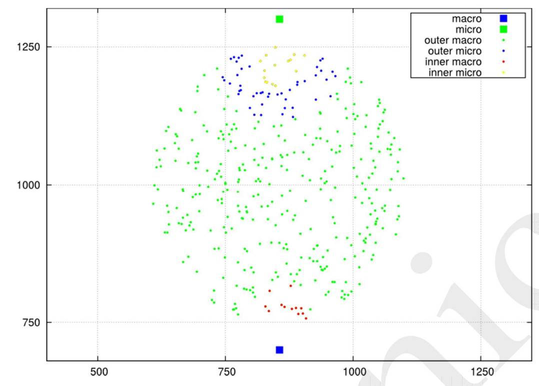

Figure 16 – Node deployment and UE association.

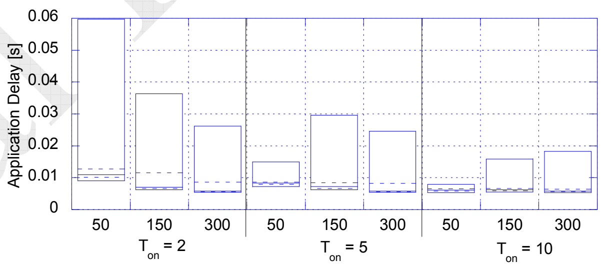

Figure 17 – Application delay with increasing values of T_{on}.

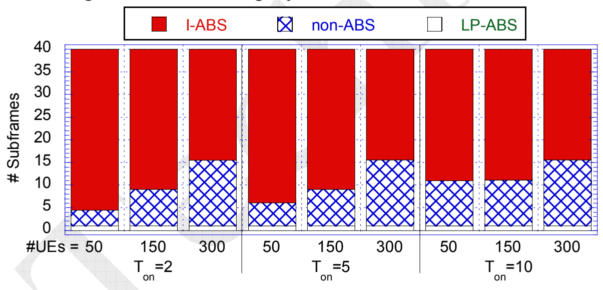

Figure 18 – Number of subframes of each type with increasing values of T_{on}.

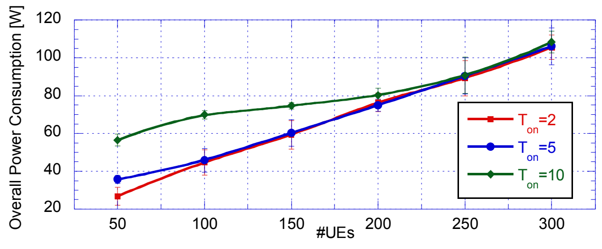

Figure 19 – HetNet power consumption (variable part) with increasing values of T_{on}.

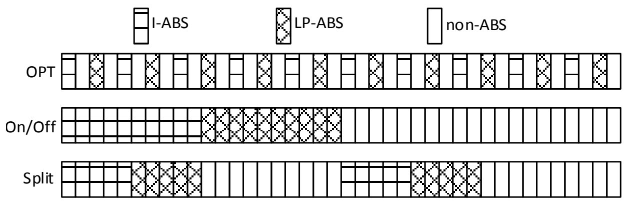

Figure 20 – Examples of positioning of 10 I-ABS, 10 LP-ABS with three different algorithms.

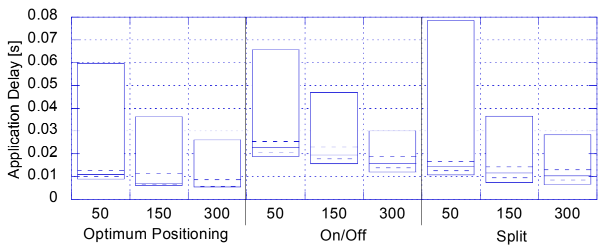

Figure 21 – Application delay for various positioning algorithms, T_{on}=2.

We now separate the effects on power consumption of dynamic APP and SA. We do this by comparing PSF with a system with a *fixed* ABS ratio, equal to 3/8, with and without SA. While both are able to carry the same traffic, as shown in Figure 22, they do so by consuming different powers, as shown in Figure 23. Figure 24 and Figure 25 show the power consumptions of the macro and the micro separately. Figure 24 shows that most of the power is saved by provisioning as many ABSs as possible, i.e., sending the macro to sleep as often as possible, which is the contribution of APP. On the other hand, SA has an impact too, since employing it with a fixed ABS ratio reduces the power consumption at the micro, as shown in Figure 25. To make the comparison more challenging, we also show in Figure 25 the power that the micro consumes if it does not run SA, but just goes to sleep whenever its backlog is zero (tagged as *DS* in the figure, for "*dynamic switch-off*"). The figure shows that DS does save power with respect to staying always on, but it still leaves a wide margin, which is instead covered by SA. SA improves on DS by around 50%. The impact of SA on the *overall* power reduction in the HetNet, in general, depends on the number of micro UEs, the number of micros and their power model. In this scenario, it is comparatively minor, since a single micro with few users consumes considerably less power than the macro. However, we will show later on that SA yields another benefit, i.e. it favors desynchronization of micro activations, which reduces interference in a multi-micro HetNet.

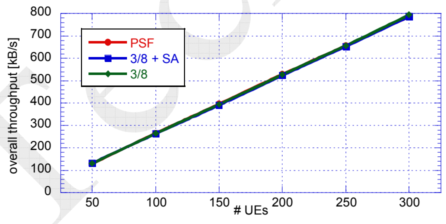

Figure 22 – HetNet MAC-layer throughput with dynamic ABS provisioning, and a static one with/without SA

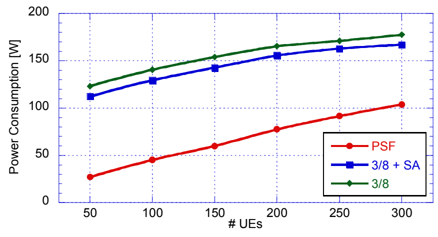

Figure 23 – Power consumption with dynamic ABS provisioning, and a static one with/without SA

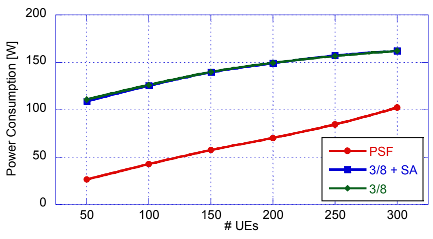

Figure 24 - Power consumption of the Macro node with dynamic ABS provisioning, and with a static one with/without SA

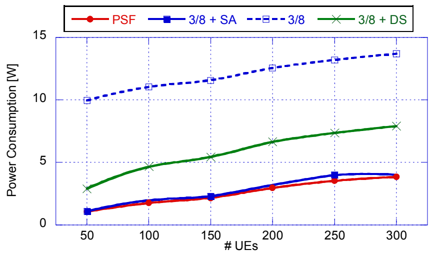

Figure 25 - Power consumption of the Micro node with dynamic ABS provisioning, and with a static one with/without SA

We now show that PSF works with various Cell-Range Expansion Bias (CREB) values. Figure 26 shows that the overall power consumption decreases with the CREB, as more UEs are served by the micro, which in turn has a lower power consumption than a macro. The number of micro UEs ranges from 12% to 30% when varying the CREB from 0dB to 6dB. The effects on QoS are negligible at the considered load, as we show in Figure 27. Such increase in the micro load is coped with by the PSF by adapting the number of non-ABS and I-ABS, as shown in Figure 28. We remark that setting the optimal CREB requires considering many factors, e.g., the computational capabilities of a micro, or interference from other, non-coordinated macros, and it is clearly outside the scope of this paper.

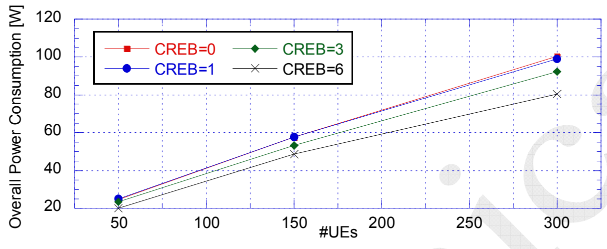

Figure 26 – HetNet power consumption (variable part) with several values of the CREB.

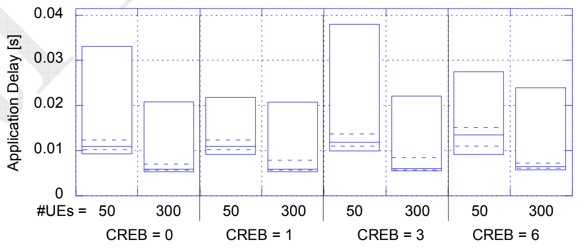

Figure 27 – Application delay with an increasing CREB.

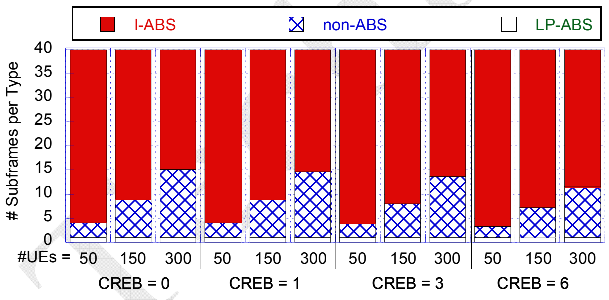

Figure 28 – Number of SFs of each type with an increasing values of CREB

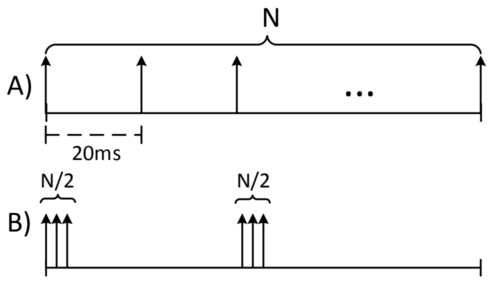

Figure 29 – Examples of smooth (A) and bursty (B) traffic in an active period.

We now show how PSF performs with bursty traffic. We modify traffic generation so that an application packs the same number of packets it would send during an active period into two bursts, as shown in Figure 29. The average number of packet per burst is 20, and the mean rate of applications stays the same.

As we can see in Figure 30, the power consumption is similar to the case of smooth traffic, the relative difference (shown in green and referenced on the right *y* axis) being less than 10% either way depending on the load. With bursty traffic, PSF consumes slightly *more* power at very low loads (e.g., 25 UEs), and *less* power as the load increases, having a break-even point around 50 UEs. This is due to the interplay of two distinct phenomena: on one hand, the fact that traffic is bursty allows the eNB to pack several RLC SDUS (i.e., IP packets with PCDP and RLC headers) into the same MAC-layer transport block, thus reducing the overhead of MAC and RLC headers on MAC-layer transmissions. Thus, a smaller MAC-level traffic is obtained for the same application-level traffic. On the other hand, bursty traffic affects the computation of *K*. Successive values of *K* at the macro are shown in Figure 31. The fact that a non-zero *K* is sometimes followed by a null *K* implies that the PSF uses the former in an AP where no traffic arrives, hence overestimates the expected load for that AP. Figure 32 confirms this, showing that the RBs utilization during non-ABSs at the macro is lower when bursty traffic is transmitted, and such difference is more evident at low loads. Note that the knee in the RB utilization at low loads (regardless of whether traffic is smooth or bursty) is due to the presence of T_{on}>0, which forces more non-ABSs than strictly necessary. However, the burstiness of the traffic does not affect the throughput, as confirmed by Figure 33, where the (negligible) differences are due to the above-mentioned reduction in the number of RLC and MAC headers. Finally, delays are higher with bursty traffic, due to the fact that an arriving packet is delayed not just by packets from other flows, but also from the part of burst ahead of itself that has just arrived at the queue.

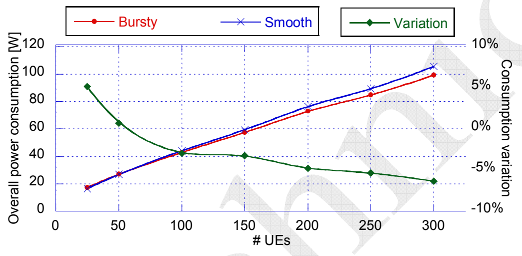

Figure 30 – HetNet power consumption with smooth and bursty traffic (left *y* axis), and relative difference (right *y* axis), with an increasing number of UEs.

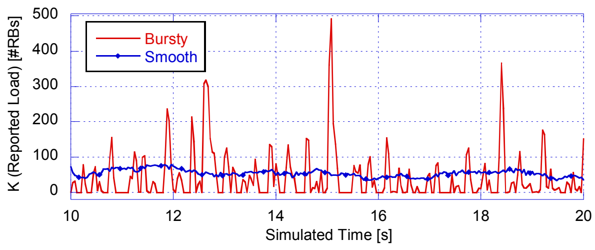

Figure 31 – Reported *K* at the macro with smooth and bursty traffic, in a scenario with 25 UEs.

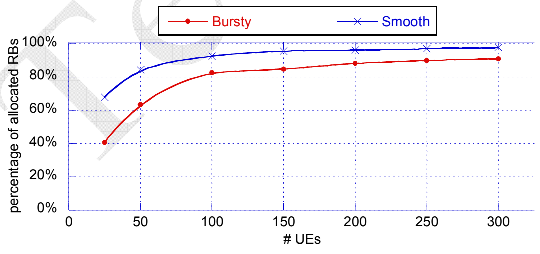

Figure 32 – Percentage of RBs allocated by macro eNB during non-ABS, with smooth and bursty traffic.

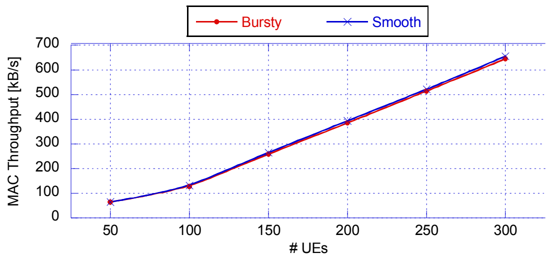

Figure 33 - MAC-layer HetNet throughput with smooth and bursty traffic.

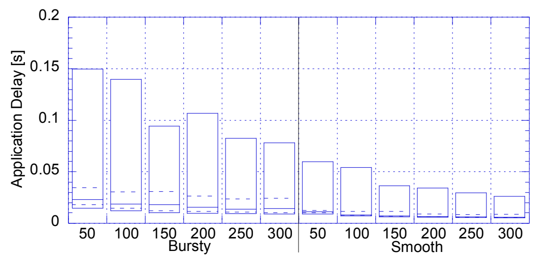

Figure 34 – Application delay with smooth and bursty traffic.

The last two parameters whose effects we test are the *reporting offset* and *reconfiguration period*, both defined in Section 5. In Figure 35 and Figure 36, respectively, we show the distribution of the application delay and the number of SFs per type, for values of the reconfiguration period, ranging from 40 to 800 ms. As we can see, using smaller periods allows faster reactions to traffic variations, hence reduces the application delay. This is also confirmed by the trend in the number of non-ABSs, which decreases slightly as the period increases, leading to small differences in terms of power consumption, as we show in Figure 37.

We then evaluate the robustness of PSF to reporting offsets. For this purpose in Figure 38 we show the application delay for various values of the reporting offset, demonstrating a negligible effect of the latter for values up to 20 ms. Such time is largely sufficient to solve the provisioning problems *and* to complete the handshake via X2. From now on, we assume a reconfiguration period of 40ms, i.e., APs are changed every time, and a reporting offset of 10ms.

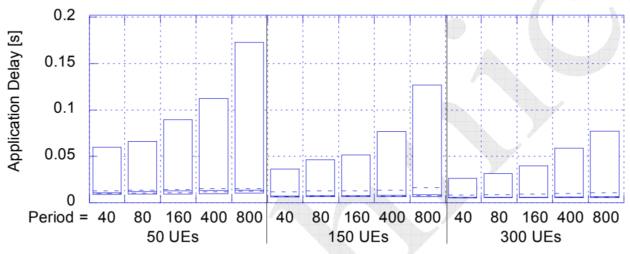

Figure 35 – Application delay with an increasing value of the reconfiguration period, T_{on}=2.

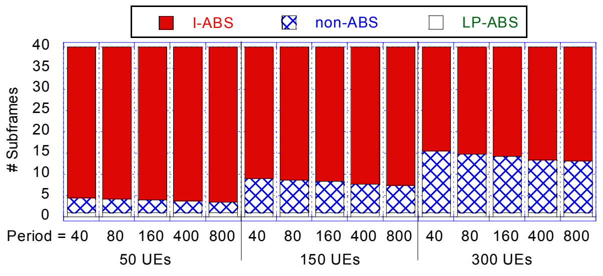

Figure 36 - Number of subframes of each type with an increasing value of the reconfiguration period, T_{on}=2.

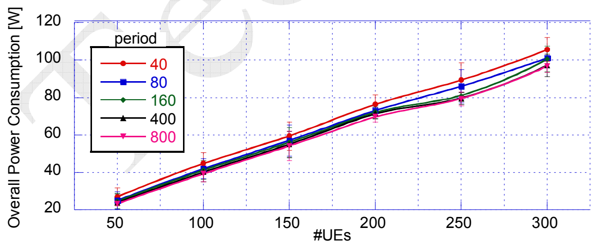

Figure 37 - HetNet power consumption against the number of UEs with several reconfiguration periods and T_{on}=2.

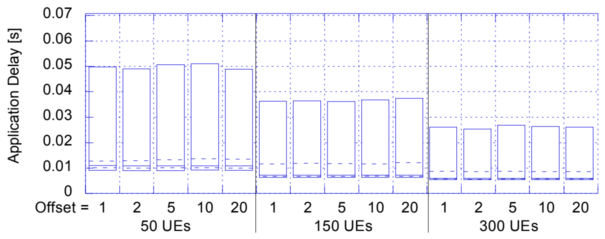

Figure 38 - Application delay with an increasing value of *offset*, T_{on}=2, reconfiguration period 40 ms.

Finally, we verified the feasibility of our framework by measuring the solving time for the ILPs on an off-the-shelf computer equipped with an 8-core Intel i7-4790 CPU @ 3.60GHz and 16 GB RAM. We considered various load configurations, and we obtained values of the solving times always between 0.5 and 5 ms. Figure 39 reports a sample of the solving times for the two ILPs, in a scenario with 50 UEs and one micro (we recall that the solving times of both ILPs are independent of the number of UEs and micros). The displayed times include the time it takes to fill the data structures that CPLEX takes as an input and read the solution that it returns, and may include operating system overheads, since interrupts are not disabled. Moreover, we measured the *total* number of *simplex iterations* per solution over more than 750 instances of each problem. Simplex is used by CPLEX to solve LP relaxations at each node in the tree of a branch-and-cut execution. A simplex iteration amounts to solving a linear system with a number of variables equal to the problem’s (i.e., four at most). The distribution of simplex iterations, shown in Figure 40, provides some insight on the actual algorithmic steps involved in solving our ILPs. First of all, it shows that CPLEX presolver solves *at optimality* a sizable percentage of instances, notably all those where the reported iterations are zero. This happens whenever an instance’s constant values (i.e., the inner/outer load, the minimum *on* time, the constraints posed by micros to the macro and vice-versa) are such that the optimal solution can be found through (often non-obvious) automated logical reasoning alone. In the other cases, branch-and-cut is used, but the overall number of algorithmic steps it performs is indeed limited. When this happens, the solutions are guaranteed to be within a 0.5% optimality gap.

These results complement the theoretical bounds of Lemmas 1 and 2 and the discussion in Section 5.4, showing that the computational requirements of our PSF are well within the reach of today’s hardware.

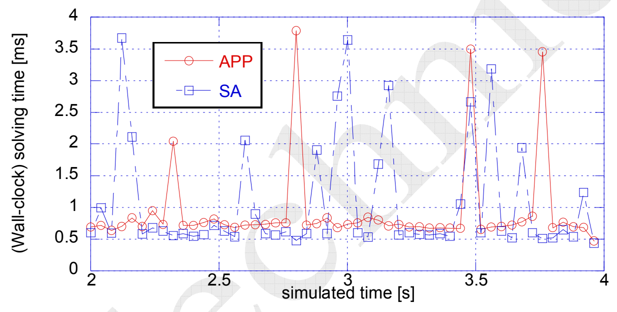

Figure 39 – ILP solving times, one micro and 50UEs.

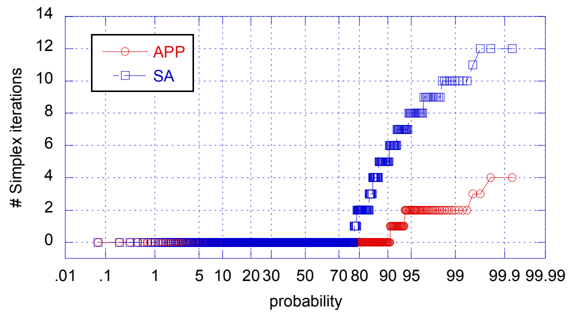

Figure 40 – CDF of the number of simplex iterations.

2. Comparison

We now compare PSF against a baseline with no ABS support and work a work that advocates dynamic computation of ABSs, i.e., [10], and we will refer to them respectively as *noABS* and *RelWork*. To make the comparison more challenging, we enhanced [10] with our ABS-placing algorithm, and we allow a macro node to go to sleep during ABSs (something which [10] did not posit, and that does reduce that scheme’s power consumption). We also remark that [10] assumes HetNet-wide *omniscience* at the macro, while our framework does not.

Single-Micro scenario

We consider a single-micro deployment like the one of the previous sub-section. We start by comparing the application delays for an increasing number of UEs, using two values of T_{on}. As Figure 41 shows, when using T_{on}=10 our framework yields lower delays than all the other configurations, with a significant improvement at lower loads. With T_{on}=2 instead, delays for percentiles up to the 75th are close to the ones of the related work. Figure 42 shows that PSF is the most energy-efficient, with a power saving of up to 72% with T_{on}=2, and up to 41% with T_{on}=10. The latter configuration in fact, achieves lower delays at a cost of an increased power consumption, i.e. allocates more non-ABSs. However, the number of I-ABSs is still significantly higher than the related work, which allocates SFs proportionally to the *ratio* of the macro and micros loads, regardless of their *absolute* offered load. This explains why the related work curve in Figure 37 is approximately flat, and implies that, when the absolute load is low, an excess of non-ABSs is allocated. The fact that, especially at higher loads, using a power-saving algorithm yields lower powers *and* smaller delays (see Figure 41 and Figure 42) appears to be counterintuitive: in fact, power saving is achieved by switching off nodes, and this creates interference-free environments for other nodes, which in turn employ fewer resources to serve their UEs with higher efficiency, hence serving them faster. This is why a solution without ABSs performs the worst in both respects. At low loads, instead, switching off nodes results in an increased latency. These two phenomena can be observed consistently in all the scenarios analyzed in the rest of the paper. In the following experiments, we use T_{on}=2.

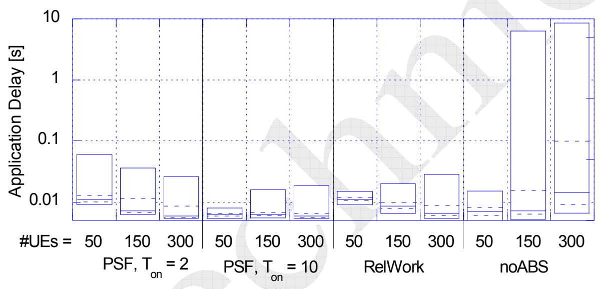

Figure 41 - Application delays with an increasing number of UEs, packet size 50 bytes.

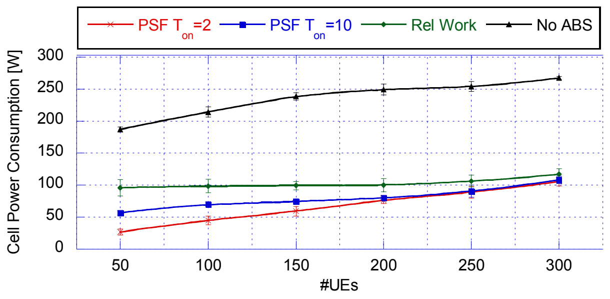

Figure 42 - HetNet power consumption (variable part) with an increasing number of UEs, packet size 50 bytes.

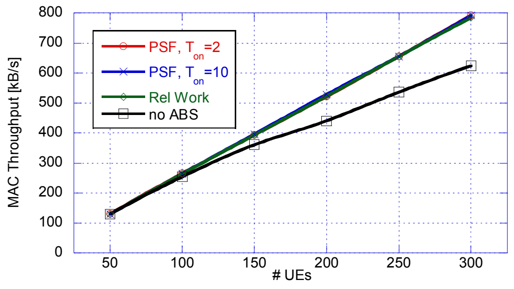

Figure 43 - MAC-level HetNet throughput in a single-micro scenario with an increasing number of UEs

As a next step, we perform a saturation analysis of the system. To that purpose, we place 200 UEs, which gives us enough spatial diversity, and we increase the size of the packets, until the system reaches saturation. Figure 44 represents the overall MAC-level throughput, which shows that using ABSs increases the overall system performance, and that our solution achieves a slightly higher saturation throughput than the related work’s. Moreover, we can see from Figure 45 that the related work scheme cannot keep the *delays* within an acceptable range, already at moderate loads. The fact that its power consumption (shown in Figure 46) is lower, which is due to the overabundance of I-ABSs (see Figure 47) shows that the trade-off point between power saving and QoS is largely suboptimal for the related work. Our framework, instead, consumes less power at low loads, and allocates more power to preserve QoS as the load increases.

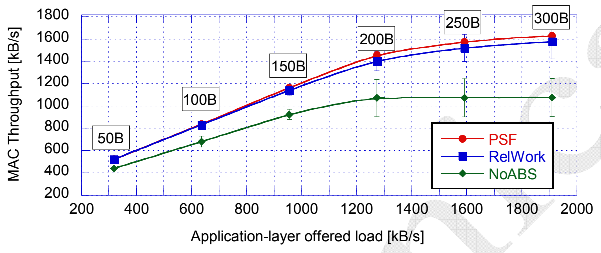

Figure 44 - MAC-layer HetNet throughput against the application-layer offered load. For each offered load, the corresponding application-layer packet size is reported in the labels.

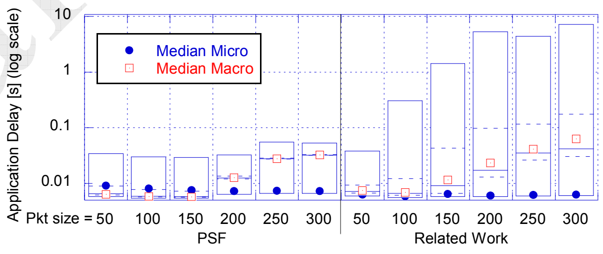

Figure 45 - Application delay with an increasing offered load.

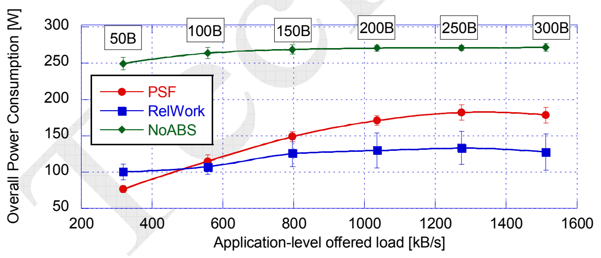

Figure 46 - HetNet power consumption (variable part) against application-layer offered load. For each offered load, the corresponding application-layer packet size is reported in the labels.

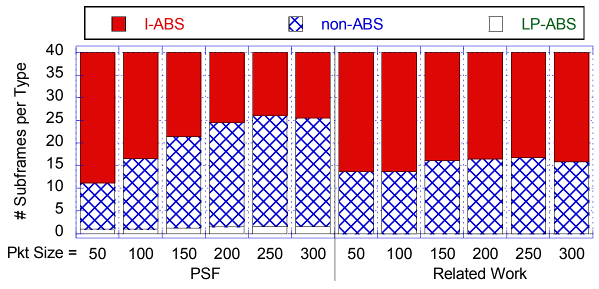

Figure 47 - SF-type allocation with an increasing application-layer offered load, 200 uniformly distributed UEs.

More micros

We now compare the performance of our framework in a scenario with one macro and two micros, with UEs uniformly dropped, as shown in Figure 48. First, we evaluate the QoS by considering an increasing number of UEs generating traffic with fixed bitrate (packets 50 bytes long). Figure 49 shows the distribution of application delays in the three cases. As we can see, our solution outperforms both the baseline and the related work. However, we observe significantly different results with respect to the single micro scenario, and the main reason behind this phenomenon is the interference *between micros*. As explained in Section 5.3, our SA algorithm forces micros to remain active for the minimum required time, *and* de-synchronizes micro activations by picking a random initial offset at each AP. Besides the (comparatively minor) power saving due to the switch-off of the single micros, this mechanism favors a multiplexing over time of micro activations, hence *reduces inter-micro interference*. In fact, we verified that our solution achieves higher CQIs and has a lower rate of decoding errors at the MAC layer, as the rate of simultaneous micro transmissions – which also causes interference – is kept lower than the related work’s (see Figure 50). The fact that at high loads micros transmit *simultaneously* even less often when no ABSs are provisioned is due to the fact that micros cannot serve their outer UEs at all in that case, hence transmit less often altogether. Figure 51 shows that the combined effect of micro switchoff and inter-micro interference reduction allows further power saving opportunities.

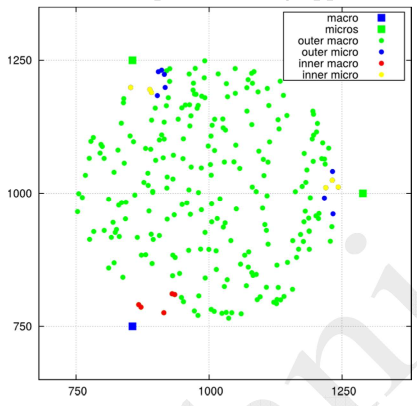

Figure 48 - Node deployment and UE association in a scenario with two micros.

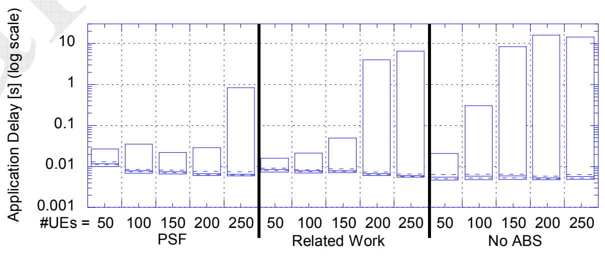

Figure 49 - Application delay with an increasing number of UEs.

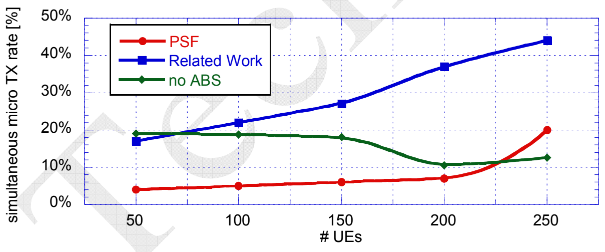

Figure 50 – Percentage of TTIs where micros transmit simultaneously against the number of UEs.

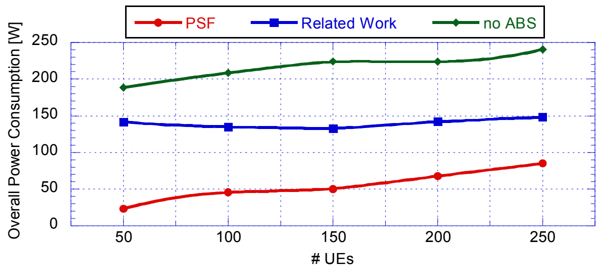

Figure 51 – HetNet power consumption (variable part) with an increasing number of UEs.

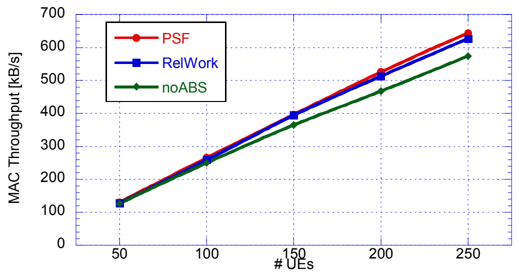

Figure 52 - MAC-level HetNet throughput in a two-micro scenario with an increasing number of UEs

As a further test, we analyze the system in Figure 53, where UEs are deployed according to the same spatial distribution, but micros are closer to the macro node. In this case, the average number of UEs served by the micros decreases, as they suffer a stronger interference from the macro. Moreover, the inter-micro interference increases as well, as the distance between them decreases. Figure 55 shows a trend of application delay that is similar to the previous scenario with micros at the cell border, as confirmed also by the rate of simultaneous micro transmissions that we show in Figure 54. PSF provides lower delays at lower percentiles in almost all cases, performing slightly worse at low loads, mainly due to the low value of T_{on}. The performance of the No-ABS configuration are even worse than the previous case, as micros UEs suffer from a higher interference from macros. Similar considerations can be made for the power consumption, shown in Figure 56.

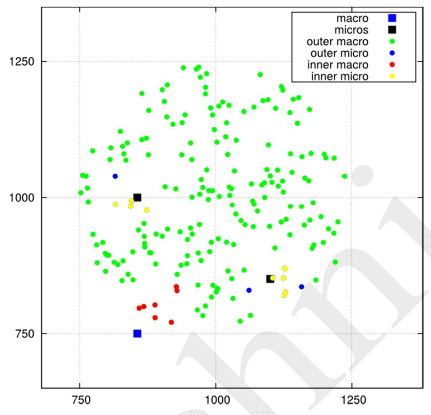

Figure 53 – Node deployment and UE association in a scenario with two micros close to the macro.

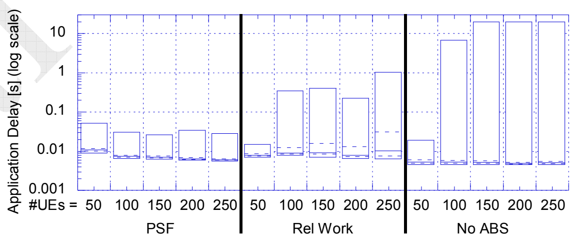

Figure 55 – Application delay with an increasing number of UEs, in a scenario with two micros close to the macro node.

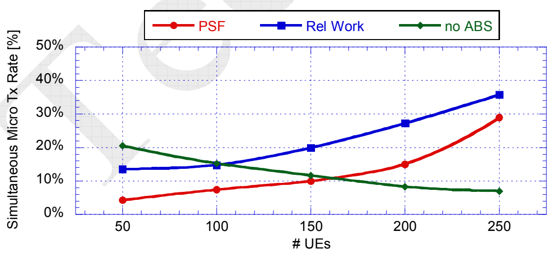

Figure 54 – Percentage of TTIs where micros transmit simultaneously against the number of UEs.

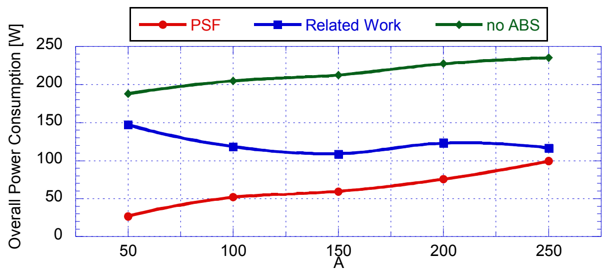

Figure 56 – HetNet power consumption (variable part) with an increasing number of UEs, in a scenario with two micros close to the macro node.

Finally, we show that increasing the number of micros does not change the qualitative outlook of the results. We simulate the scenario in Figure 57, where five micros are under the umbrella of the same macro. Figure 58 shows that the power consumption gap is approximately the same as in the previous cases. Figure 59 shows that the RW scheme fails to keep up with the HetNet demands, achieving a slightly lower throughput. Figure 60 shows that both phenomena are due to the fact that micro de-synchronization brings benefits.

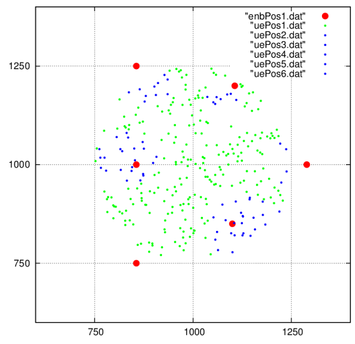

Figure 57 - Node deployment and UE association in a scenario with five micros.

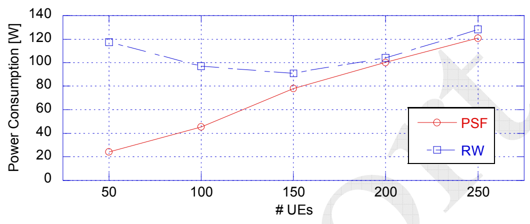

Figure 58 - HetNet power consumption (variable part) with an increasing number of UEs, in a scenario with five micros.

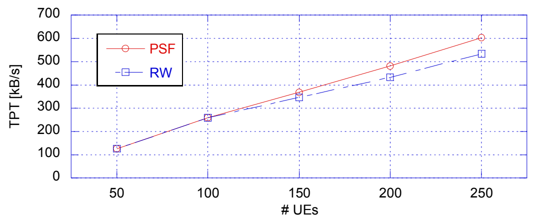

Figure 59 - HetNet throughput in a scenario with five micros.

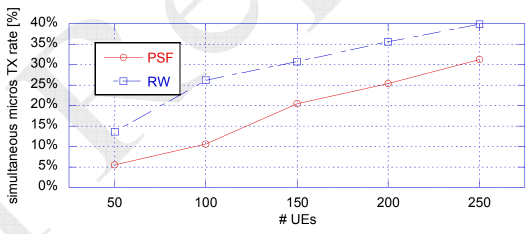

Figure 60 - Percentage of TTIs where more than one micro transmits against the number of UEs

Micros with hot spot

All the results we analysed so far showed an allocation of ABS that was exclusively composed of I-ABS and non-ABS, with only one LP-ABS allocated for CQI measurement. To demonstrate the usefulness of LP-ABS, we now assess the performance of our framework in a hot-spot scenario like the one shown in Figure 61. We used two micros as in the previous section, deploying UEs in two sets: a first one composed of 50 UEs uniformly distributed in the cell, and a second one of 150 UEs uniformly dropped in a hotspot close to the macro node. The traffic volume is varied by increasing the packet size only for UEs within the hotspot, until the system reaches saturation. Figure 62 shows that the performance in terms of MAC cell throughput are similar in the three cases, as most of the traffic is generated in proximity of the macro node, thus benefiting less from the usage of ABSs. However, the amount of power consumed by the three nodes, which is shown in Figure 63, is significantly lower with our framework. The main reason for this is displayed in Figure 64: in this case in fact, our framework adapts its allocation to the unbalanced deployment of UEs serving most of them at low power, thus using LP-ABS and reducing power consumption with no impact on QoS.

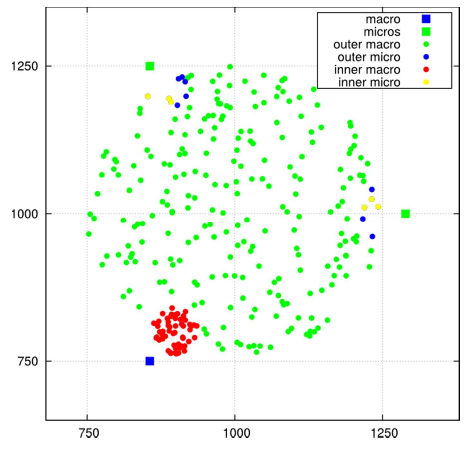

Figure 61 - Node deployment and UE association in a scenario with two micros and a hot spot close to the macro.

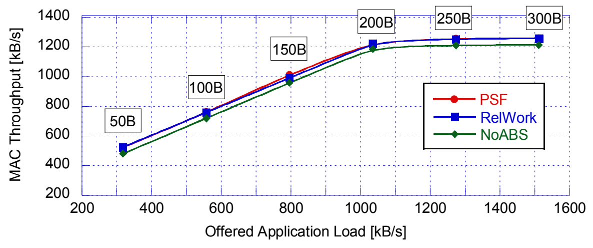

Figure 62 - MAC-layer HetNet throughput against the application-layer offered load. The offered load is increased for hot-spot users only. For each offered load, the corresponding application-layer packet size is reported in the labels.

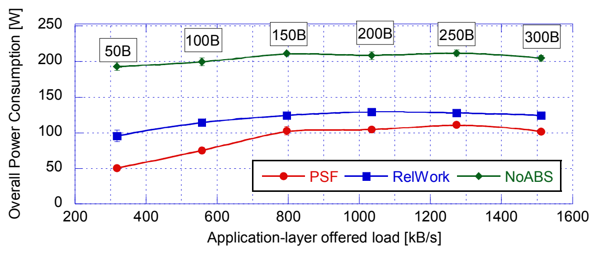

Figure 63 - HetNet power consumption against application-layer offered load. The offered load is increased for hot-spot users only. For each offered load, the corresponding application-layer packet size is reported in the labels.

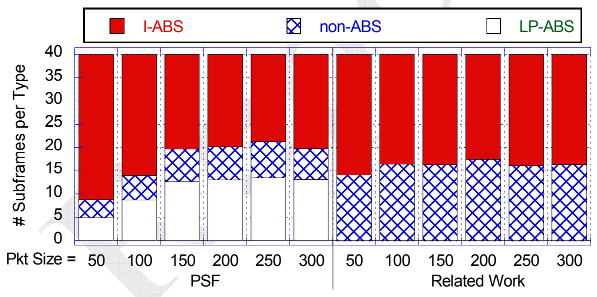

Figure 64 - Subframe-type allocation with an increasing application-layer offered load.

Multiple macro nodes

While investigating multi-macro coordination is outside the scope of this paper, it is important to understand how PSF works in a network with multiple macros. We evaluate the scenario of Figure 65, where we use the same spatial distribution of UEs for the *central* cell as Figure 16, and we deploy three *external* macro nodes at 1km from the center of the central cell, radiating towards it, to create a high interference. The central nodes always run the PSF framework, whereas the *external* nodes may: a) not use the ABS mechanism; b) run *autonomous* instances of PSF, i.e., compute their own ABS pattern; c) run a *coordinated* PSF with the central cell. In the last case, each macro node computes its own values of *K* and reports this and its micro load to a central entity. The latter takes the maximum of all the reported values, runs the ABS provisioning algorithm described in Section 5.2, and returns the result to all the macros, which then apply it and send it to their micros. This makes all the coordinated entities agree on the same ABS pattern. We consider two load levels for the *external* nodes: low load, where the RB demand is 10% of the total, and high load, where it is 50%. We measure the performance in the central cell only.

As we can see in Figure 66, the load at the external nodes increases the power consumption. This is because a higher load implies higher interference, hence lower CQIs, hence more RBs required to serve the same traffic. However, when external cells run *autonomous* PSF instances, the increase in the power consumption is minor. The justification can be found in Figure 67. When no ABSs are used at the *external* nodes in fact, these generate a strong interference potentially in *all* the SFs, causing low CQIs of UEs served by the micro. Running a *coordinated* PSF, instead, achieves maximum protection for micro UEs, significantly increasing their channel quality. However, the same mechanism forces all the macro nodes to transmit during the same SFs (i.e. non ABS), thus increasing inter-macro interference (see left part of Figure 67) and reducing the CQIs of the macro UEs, which are comparatively more numerous and served by a node with a higher power consumption. This justifies the higher power consumption of coordinated PSF. On the other hand, running *autonomous* PSF instances yields a favorable trade-off, increasing the chance that both micro and macro users are scheduled in SFs where external interferers are inactive.

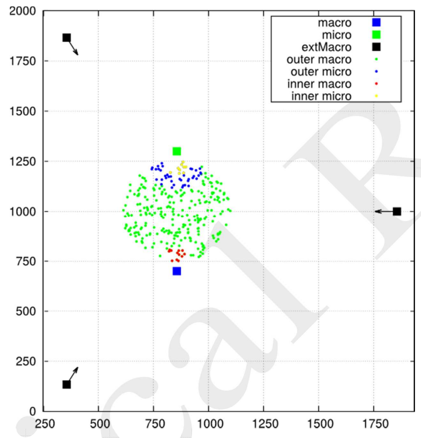

Figure 65 – Node deployment and association in a scenario with interference coming from external cells.

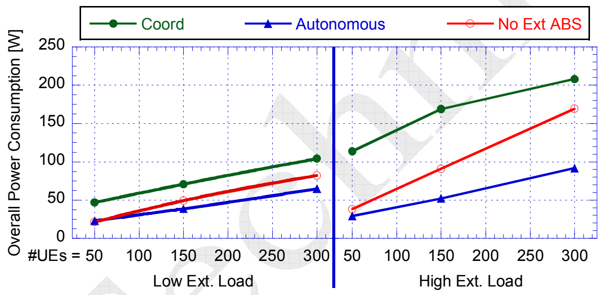

Figure 66 – HetNet power consumption (variable part) in a scenario with interference from external macros in two configurations having respectively low (left) and high (right) load in the external cells.

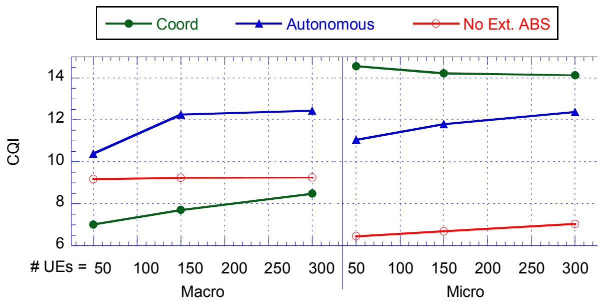

Figure 67 – CQI of Macro (left) and Micro (right) UEs in a scenario with interference from external macros at high load.

7. Conclusions

In this paper, we presented a framework for ABS provisioning that aims at reducing the power consumption in HetNets. Our framework exploits two types of ABSs, idle and low-power ABS, where a macro node respectively refrains from transmitting data or still does so at a lower transmission power. The provisioning process is carried out dynamically at the fastest possible pace, selecting which ABS type to use depending on information on the network load, such as its volume and/or spatial distribution. The framework also accounts for macro-micro communication, which is realized via X2-interface and using standard-compliant signaling only, and its computational burden is independent of the number of users, the system bandwidth and the volume of traffic.

Finally, we evaluated our scheme by means of system-level simulations, comparing it against a legacy system with no ABS support, and a dynamic scheme taken from the literature. We showed that our framework consumes significantly less power at low loads, and preserves QoS as the system approaches saturation by dynamically adapting to different user deployments. Moreover, running autonomous instances of our scheme at macro nodes yields benefits in terms of consumed power and UE channel quality (both macro and micro).

8. Acknowledgements

The subject matter of this paper includes description of results of a joint research project carried out by Telecom Italia and the University of Pisa. Telecom Italia reserves all proprietary rights in any process, procedure, algorithm, article of manufacture, or other result of said project herein described. We would like to thank Antonio Frangioni, professor of Operations Research at the University of Pisa, for his advice on ILP solution algorithms.

References

[1] Dario Sabella, et al., “Energy Management in Mobile Networks Towards 5G”, in M.Z. Shakir et al. (eds.), Energy Management in Wireless Cellular and Ad-hoc Networks, Studies in Systems, Decision and Control, Springer, 2016

[2] Towards Real Energy-efficient Network Design (TREND) project. http://www.fp7-trend.eu/

[3] 3GPP, R1‑113482, Ericsson, ST-Ericsson System performance evaluations on FeICIC, Oct. 2011.

[4] S. Deb, P. Monogioudis, J. Miernik and J. P. Seymour, "Algorithms for Enhanced Inter-Cell Interference Coordination (eICIC) in LTE HetNets," in IEEE/ACM Transactions on Networking, vol. 22, no. 1, pp. 137-150, Feb. 2014.

[5] A.Merwaday, S.Mukherjee, I.Güvenç, "Capacity analysis of LTE-Advanced HetNets with reduced power subframes and range expansion", EURASIP Journal on Wireless Communications and Networking, pp 1-19, 2014

[6] E. Visotsky, *et al.*, "Joint Scheduling for CoMP and eICIC in Heterogeneous Network Deployments," Vehicular Technology Conference (VTC Spring), 2013 IEEE 77th, Dresden, 2013, pp. 1-5.

[7] M. Wang, H. Xia and C. Feng, "Joint eICIC and dynamic point blanking for energy-efficiency in heterogeneous network," Wireless Communications & Signal Processing (WCSP), 2015 International Conference on, Nanjing, 2015, pp. 1-6.

[8] F. Alfarhan, R. Lerbour and Y. Le Helloco, "An Optimization Framework for LTE eICIC and Reduced Power eICIC," 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, CA, 2015, pp. 1-6.

[9] Yu Chen, Xuming Fang and Bo Huang, "Joint ABS power and resource allocations for eICIC in heterogeneous networks," The Sixth International Workshop on Signal Design and Its Applications in Communications, Tokyo, 2013, pp. 92-95.

[10] S. Vasudevan, R. N. Pupala and K. Sivanesan, "Dynamic eICIC — A Proactive Strategy for Improving Spectral Efficiencies of Heterogeneous LTE Cellular Networks by Leveraging User Mobility and Traffic Dynamics," in IEEE Transactions on Wireless Communications, vol. 12, no. 10, pp. 4956-4969, October 2013.

[11] 3GPP TS 36.420 v13.0.0, “Evolved Universal Terrestrial Radio Access Network (E-UTRAN); X2 general aspects and principles (Release 13),” December 2015.

[12] 3GPP TS 36.423 v13.2.0, “Evolved Universal Terrestrial Radio Access Network (E-UTRAN); X2 Application Protocol (X2AP) (Release 13),” December 2015.

[13] 3GPP - TS 36.300 Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2

[14] B. Soret, H. Wang, K. I. Pedersen and C. Rosa, "Multicell cooperation for LTE-advanced heterogeneous network scenarios," in IEEE Wireless Communications, vol. 20, no. 1, pp. 27-34, February 2013.

[15] K. I. Pedersen, B. Soret, S. Barcos Sanchez, G. Pocovi and H. Wang, "Dynamic Enhanced Intercell Interference Coordination for Realistic Networks," in IEEE Transactions on Vehicular Technology, vol. 65, no. 7, pp. 5551-5562, July 2016.

[16] A. Liu, V. Lau, L. Ruan, J. Chen, and D. Xiao, “Hierarchical radio resource optimization for heterogeneous networks with enhanced intercell interference coordination (eicic),” IEEE Trans. Signal Processing, vol. 62, no. 7, pp. 1684–1693, April 2014.

[17] H. Zhou, Y. Ji, X. Wang and B. Zhao, "ADMM based algorithm for eICIC configuration in heterogeneous cellular networks," 2015 IEEE Conference on Computer Communications (INFOCOM), Kowloon, 2015, pp. 343-351.

[18] S. Kim, S. Choi, and B. G. Lee, “A joint algorithm for base station operation and user association in heterogeneous networks,” IEEE Communications Letters, vol. 17, no. 8, pp. 1552–1555, August 2013.

[19] M. Ajmone Marsan, L. Chiaraviglio, D. Ciullo, M. Meo, “Switch-off transients in cellular access networks with sleep modes”, proc. of ICC 2011, Kjoto, Japan, 5-9 June 2011.

[20] U. Siddique, H. Tabassum, E. Hossain, D. In Kim, "Channel-Access-Aware User Association With Interference Coordination in Two-Tier Downlink Cellular Networks," IEEE Transactions on Vehicular Technology, vol. 65, no. 7, pp. 5579-5594, July 2016.

[21] Q. Ye, O. Y. Bursalioglu, H. C. Papadopoulos, C. Caramanis, and J. G. Andrews, “User association and interference management in massive MIMO hetnets,” IEEE Trans. Commun., vol. 64, no. 5, pp. 2049–2065, May 2016.

[22] X. Wang, *et al.*, "Reduced power centralized eICIC for LTE-advanced heterogeneous networks," 2014 IEEE/CIC International Conference on Communications in China (ICCC), Shanghai, 2014, pp. 743-747.

[23] M. Amara and A. Feki, "Optimized ABS in LTE-Advanced Heterogeneous Networks with Adaptive Macro Cell Transmission," 2015 IEEE Globecom Workshops (GC Wkshps), San Diego, CA, 2015, pp. 1-7.

[24] M. Simsek, M. Bennis, İ. Güvenç, “Learning Based Frequency- and Time-Domain Inter-Cell Interference Coordination in HetNets”, IEEE Transactions on Vehicular Technology, Vol. 64, no. 10, pp. 4589-4602, Oct. 2015.

[25] A. Morimoto, N. Miki and Y. Okumura, "Performance Evaluation of Reduced Power Inter-cell Interference Coordination for Downlink in LTE-Advanced Heterogeneous Networks", Proc. 19 European Wireless Conference, Guildford, UK, 2013, pp. 1-6.

[26] A. M. Sadekar and R. H. Hafez, "LTE-A enhanced Inter-cell Interference Coordination (eICIC) with Pico cell adaptive antenna," Network of the Future (NOF), 2015 6th International Conference on the, Montreal, QC, 2015, pp. 1-6.

[27] M.Yassin, *et al.*, "Survey of ICIC techniques in LTE networks under various mobile environment parameters", Wireless Networks (2015), doi:10.1007/s11276-015-1165-z.

[28] C. Mehlführer, J.C. Ikuno, M. Simko, S. Schwarz, M. Rupp, “The Vienna LTE simulators—enabling reproducibility in wireless communications research”. EURASIP Journal on Advances in Signal Processing (JASP), S.I. on Reproducible Research (2011).

[29] M. I. Kamel and K. M. F. Elsayed, "Performance evaluation of a coordinated time-domain eICIC framework based on ABSF in heterogeneous LTE-Advanced networks," Proc. IEEE GLOBECOM 2012, Anaheim, CA, 2012, pp. 5326-5331.

[30] G. Piro, L. Grieco, G. Boggia, F. Capozzi, and P. Camarda, "Simulating LTE Cellular Systems: An Open-Source Framework," IEEE Journal on Vehicular Technology, 2011, pp. 498-513.

[31] 3GPP TR 36.814, "Further advancements for E-UTRA physical layer aspects", Release 9, V.9.0.0, March 2010.

[32] A. Prasad, A. Maeder, C. Ng “Energy Efficient Small Cell Activation Mechanism for Heterogeneous Networks”, proc. Globecom 2013, pp. 754-759

[33] K. I. Pedersen, Y. Wang, S. Strzyz, F. Frederiksen, “Enhanced Inter-cell Interference Coordination in Co-channel Multi-layer LTE-Advanced Networks”, IEEE Wireless Communications, June 2013, pp. 120-127

[34] Energy Aware Radio and Nework Technologies (EARTH) project. Deliverable D2.3 - Energy efficiency analysis of the reference systems, areas of improvements and target breakdown. http://www.ict-earth.eu/

[35] ILOG CPLEX Software, http://www.ilog.com

[36] A. Virdis, G. Stea, G. Nardini, “SimuLTE: A Modular System-level Simulator for LTE/LTE-A Networks based on OMNeT++”, proc. SimulTech 2014, Vienna, AT, August 28-30, 2014

[37] SimuLTE webpage. http://www.simulte.com

[38] OMNeT++, http://www.omnetpp.org

[39] INET framework for OMNeT++: http://inet.omnetpp.org/

[40] M. W. P. Savelbergh, “Preprocessing and probing techniques for mixed integer programming problems”, ORSA Journal of Computing, Vol. 6, No. 4, pp. 445-454, Fall 1994

[41] L. Wolsey, Integer Programming, John Wiley and Sons, Inc, 1998

9. APPENDIX

Proof of Lemma 1

<u>*Lemma 1*</u>: *Problem* (1) *admits no more than* $2\cdot\left(T/3+1\right)^3$ *solutions.*

<u>*Proof*</u>*: The solution space for* (1) *is a set of tuples* $\langle t_1,t_2,t_3,b\rangle$. *Their number is twice as large as the number of triplets of non negative numbers* $\langle t_1,t_2,t_3\rangle$ *that verify constraints (i), i.e.* $t_x \le T_x \quad x\in\{L,N,I\}$. *These constraints are verified by* T_x+1 *non negative integers. However,* $T_L+T_I+T_N=T$ *, hence a bound on the maximum number of triplets* $\langle t_1,t_2,t_3\rangle$ *is the optimum of the following problem:*

$$\begin{aligned}&\max\left[\left(T_L+1\right)\cdot\left(T_I+1\right)\cdot\left(T_N+1\right)\right]\\ &s.t. \quad T_L+T_I+T_N=T\\ &\qquad\; T_x\in\mathbb{Z}^+\end{aligned}$$

The optimum of the continuous relaxation *of the above problem is equal to* $\left(T/3+1\right)^3$ *(from the Arithmetic Mean / Geometric Mean inequality). Therefore, an upper bound on the number of solutions of* (1) *is* $S=2\cdot\left(T/3+1\right)^3$.

■

Proof of Lemma 2

<u>*Lemma 2*</u>: *Problem* (2) *admits at most* $\left(T+1\right)\cdot\left(T+2\right)$ *feasible solutions.*

<u>*Proof*</u>*: The solution space for* (2) *is included in the set of tuples* $\langle T_L,T_I,T_N,b\rangle$, *that verify constraint (i), i.e.* $T_L+T_I+T_N=T$. *Therefore the number of feasible solutions of* (2) *is at most twice as large as the number of triplets of non-negative integers whose sum is T. This last number is:*

$$Q=\sum_{j=0}^{T}\left(j+1\right)=\left(T+1\right)\cdot\left(T+2\right)/2$$

since the number of couples of non-negative integers T_L,T_I *summing to* $j\le T$ *is equal to j*+1 *(they are* $\langle 0,j\rangle$, $\langle 1,j-1\rangle$, *….,* $\langle j,0\rangle$*), and for each of these there is a unique non-negative integer* $T_N=T-j$ *that sums up to T. Therefore, the number of feasible solutions is no larger than* $S=2Q$.

■