Dynamic Preamble Subset Allocation for RAN Slicing in 5G Networks
Serdar Vural, Ning Wang, Senior Member, IEEE, Paul Bucknell, Gerard Foster, Rahim Tafazolli, Senior Member, IEEE, and Julien Muller

Abstract—The random access (RA) mechanism of Long Term Evolution (LTE) networks is prone to congestion when a large number of devices attempt RA simultaneously, due to the limited set of preambles. If each RA attempt is made by means of transmission of multiple consecutive preambles (codewords) picked from a subset of preambles, as proposed in [1], collision probability can be significantly reduced. Selection of an optimal preamble set size [2] can maximise RA success probability in the presence of a trade-off between codeword ambiguity and code collision probability, depending on load conditions. In light of this finding, this paper provides an adaptive algorithm, called Multi-preamble RA, to dynamically determine the preamble set size in different load conditions, using only the minimum necessary uplink resources. This provides high RA success probability, and makes it possible to isolate different network service classes by separating the whole preamble set into subsets each associated to a different service class; a technique that cannot be applied effectively in LTE due to increased collision probability. This motivates the idea that preamble allocation could be implemented as a virtual network function, called vPreamble, as part of a random access network (RAN) slice. The parameters of a vPreamble instance can be configured and modified according to the load conditions of the service class it is associated to.

Index Terms—LTE, mobile networks, random access, network slicing, RACH, congestion reduction, analysis, simulations, coded random access.

I. INTRODUCTION

In fourth generation (4G) mobile networks (as known as Long Term Evolution, LTE) [3], each time a connection is to be established, devices must first make a transition into the Radio Resource Control (RRC) “Connected” state [4] by sending a connection request message to the eNodeB. Devices must first be allocated with uplink frequency and time resources by the system, so that message transmissions from different devices do not cause collision at the eNodeB. To get allocated with an uplink resource, devices must indicate their intention to send a message to the eNodeB. In 4G networks, this is achieved by transmitting a random access preamble signal on the Random Access Channel (RACH).

S. Vural, N. Wang, G. Foster, and R. Tafazolli are with 5G Innovation Centre (5GIC), Institute for Communication Systems (ICS), Electronic Engineering Department, University of Surrey, Guildford, GU2 7XH, UK. E-mail: {s.vural, g.foster, n.wang, r.tafazolli}@surrey.ac.uk.
P. Bucknell and J. Muller are with Fujitsu Laboratories of Europe Ltd. E-mail: [Paul.Bucknell, Julien.Muller]@uk.fujitsu.com.
This study is partially funded by the 5G Innovation Centre (5GIC), and EU SoftFIRE project, Grant Number 687860.
Manuscript received XXX XX, 2017; revised XXX XX, 2017.

A. Preamble signals

Preamble signals are orthogonal, i.e. the eNodeB can detect and distinguish these signals when they are simultaneously transmitted by different devices. In other words, these signals could be considered as a set of orthogonal basis functions, and the eNodeB is able to detect different preambles when multiple are transmitted at the same time. However, there is a set of finite number of preamble signals (typically 54 in LTE), and devices randomly choose a single preamble from this set.

B. Collision events in the random access (RA) procedure

When multiple devices choose the same preamble and transmit it at the same RA subframe1, their messages collide as they use the same uplink resource allocated for that preamble by the eNodeB. Collision of the messages means that the access attempt made by these devices is unsuccessful, and hence they must make another attempt later. The more devices access the RACH at the same subframe, the more likely it is for a preamble to be picked by multiple devices, which leads to a higher chance of message collisions, as these devices are then allocated with the same uplink resource.

C. The massive access problem

As the number of devices expected to be connected to mobile communication networks has been rapidly increasing, the load on the random access channel (RACH) is also expected to grow [5]. Especially with the advent of emerging technologies [6], such as the Internet-of-Things (IoT) that envisions billions of connected devices in the world in the next decades, it is necessary to efficiently service a large number of devices accessing RACH to perform Random Access procedure. This scenario is in the scope of the use case massive machine type communications (mMTC) as defined for IMT 2020 in [7].

When more devices attempt RA within a small time interval, it is more likely to find more access attempts sharing the same set of preambles. Furthermore, IoT devices have regular traffic generation patterns (e.g. smart meters) [8], and may tend to establish network connection in a synchronised manner [9]. This issue, which is often referred to as the “Massive Access problem”, has been noted by the Random Access Network (RAN) study groups of the Third Generation Partnership Project (3GPP). According to 3GPP, in the worst case scenario,

1In LTE, time is divided into frames, each of which is further divided into smaller time units called subframes.
D. Multiple preamble transmissions

The root-cause of the massive access problem is the limited preamble space which cannot support a high level of device access. A simple but effective approach is to make multiple preamble transmissions back-to-back, and interpret each such preamble sequence as a codeword that represents an RA attempt from a single device [1] [16]. For instance, when codeword length is 2 (i.e. two consecutive preamble transmissions), the codeword space is expanded to 54^2, and hence collision probability is significantly reduced; i.e. it is far less likely for multiple devices to choose the same preamble sequence than it is to choose the same single preamble. This method is referred to as the Code-expanded Random Access (CeRA) [1].

On the other hand, there is a “code-ambiguity” issue to be addressed: the eNodeB does not exactly know which sequences of preambles are sent by individual devices, i.e. all that the eNodeB knows is a set of preambles in each preamble transmission time, yet not how to match the received preambles to form the exact sequences sent by devices. This causes ‘phantom’ codewords, which are those sequences interpreted by the eNodeB but not actually sent by any device. Hence, from the eNodeB’s perspective, there are too many possible combinations of preambles, which results in a large number of resources allocated to codewords not used by devices.

E. Use of preamble subsets

In LTE, devices are allowed to make a certain maximum number of RA attempts until they succeed (or eventually fail). In our previous study [2], mathematical analysis of the success probability of a single RA attempt is presented, when multiple successive preamble transmissions constitute an RA attempt, as in CeRA [1]. Results show that this technique is effective in high access intensity scenarios, and LTE is ideal for low access intensity scenarios. Furthermore, the subset size must be suitably adjusted for the technique to succeed. In different device access intensity scenarios, an optimal number of preamble subset size makes it possible to maximise the success probability of a single RA attempt and perform better than LTE. When CeRA’s preamble subset size is chosen according to the access intensity, we refer to this technique as Multi-preamble RA. In short, the finding in [2] is that, to support high RA intensity scenarios, it is necessary and sufficient to use a preamble subset (rather than the whole set of preambles).

Based on this finding, this paper proposes separation of preambles into preamble subsets, and then associating each subset with a different service class (hence different vertical markets), such as IoT, Human-to-Human (H2H), mobile broadband (MBB) etc. The scope of the study is the area covered by (and the devices attached to) a single LTE base station (eNodeB).

The concept of separation of preambles for different service classes has been mentioned for LTE in earlier studies [17] as well as industry standardisation meetings [18], with the purpose of isolating different service classes in RA, so that congestion from one class does not adversely affect others. However, with only a single preamble transmission in each RA attempt, as in LTE, limiting the preamble set size only intensifies the level of congestion in RA, which means separation of preambles in LTE makes RACH congestion even worse for those service classes with high access intensity. In contrast, as shown in [2], if each RA attempt uses a preamble sequence as a codeword, then the expansion of the orthogonal space makes it possible to reduce collision probability, hence making it practically possible to assign preamble subsets to different service classes, effectively isolating them in RA.

F. Adaptation of preamble subsets

The paper presents an adaptation algorithm based on a control feedback loop, in which the eNodeB associates a preamble subset to a service class and then adapts the subset size dynamically based on its estimation of the access intensity received from that class. Performance results in Section V demonstrate that the proposed algorithm not only makes it possible to use only a subset of the total available preamble set reserved for contention-based random access, but also enables support for a larger number of devices in RA than what LTE can support. Hence, the paper shows that it is practically possible to realise separation of preambles into mutually exclusive subsets, each associated to a different service class. The algorithm also uses uplink resources minimally, and adapts the number of resources based on the estimated device access intensity in RA.

G. Dual mode of RA operation

The proposed adaptation algorithm considers a dual-mode of operation in RA: (i) Single-Preamble mode, which is the conventional LTE RA used when access intensity is low, and (ii) Multi-Preamble mode, which uses multiple preambles in each RA attempt, and runs the proposed adaptation algorithm to adapt preamble subset size when access intensity is high. The algorithm is applied to different service classes separately, and different service classes may independently be in either of these two modes at a time.

H. Preamble allocation as a new virtual network function

Network function virtualisation (NFV) is a recent technology which is envisioned to support 5G mobile networks by providing flexibility and scalability to network functions in the network core as well as its radio access components. Thanks to the softwarisation of mobile network functions, rather than the use of dedicated hardware equipment, it is now possible to instantiate different network functions as virtual network functions (VNF), which can be started and terminated as software
instances on-demand, in a much faster way than deploying physical hardware. Groups of VNFs form a network service, making it possible to run separate instances of virtual mobile networks, each dedicated to either different service classes (vertical network slicing) or as replicates for the same service class to meet the load demands coming from that service class (horizontal network slicing). Operators could simultaneously run multiple instances of a whole mobile network core, or radio access network functions (RAN slicing), or both.

The paper shows that the proposed algorithm is effective in supporting a large RA load from a single service class using only a preamble subset. In light of this finding, the paper proposes that the preamble allocation functionality in the random access network (RAN) protocol of a mobile network could be implemented as software. Then, each instantiation of this software could be run at the eNodeB for a different service class. Not all service classes may exist all the time within the operation area of a base station. Hence, this function could be virtualised and instantiated only when there is any emerging demand, i.e. when a new service class attempts communication with the eNodeB. Thus, preamble allocation functionality could be a new RAN virtual network function (VNF), called vPreamble, which can be instantiated as part of a RAN slice. The concept is illustrated in Fig. 1.

Whenever a new RAN slice is instantiated and configured specifically for a certain service class, the parameters of the new vPreamble VNF instance (which would be part of that RAN slice) need to be configured as well. The main parameters are (i) the starting index of the preamble subset to be used by the service class, and (ii) the preamble subset size. Network operators could also use any other parameters that they would like to assign differently for different classes. The vPreamble VNF runs a set of algorithms (such as Multi-preamble RA as proposed in this paper) to modify its parameters, e.g. the preamble subset size. Then, vPreamble delivers updated parameter information to another VNF of the same RAN slice, called vAnnouncer (see Fig. 1), which is responsible for initiating announcements to devices belonging to the service class associated with that RAN slice. Alternatively, vAnnouncer could be implemented as a separate network service that coordinates all announcements made by the eNodeB for all RAN slices.

I. Paper outline

In the rest of the paper, first, prior studies on the massive access problem are briefly mentioned in Section II. Section III summarises the previous findings in [2] on maximisation of collision probability and the choice of preamble subset size. Based on this, Section IV presents the proposed adaptation algorithm. Then, Section V presents performance gains as compared to LTE and CeRA [1] for a set of key performance indicators. Finally, Section VI concludes the paper.

II. RELATED WORK

The performance of LTE under high RACH load conditions has been evaluated both analytically [19] [20] [21] [22] and via simulations [23] [24] [25]. The prior art studies proposed so far to address the massive RACH access problem can be grouped into different categories. Some surveys [10] [26] [27] as well as research articles [28] [29] provide extensive reviews of existing approaches. A summary is provided in the following.

Allocation of RACH Resources: In this category, the eNodeB allocates additional RA slots to Machine-to-Machine (M2M) devices during times of congestion [13] [18] [30], or distributes the use of existing RA slots as proportional to the arrival rate of each traffic class [12]. Although effective, this requires more RA slots in a frame, leaving fewer subframes for data transmissions. Furthermore, the technique can quickly become non-scalable to the number of new arrivals to RACH. An alternative approach is to assign additional frequency blocks to RACH (frequency domain) in each LTE time frame; however this is also not desirable due to the limited and precious frequency spectrum. It may prove to be non-scalable as well, since the number of devices accessing the system at a time can be arbitrarily large, requiring a significantly large frequency block to be assigned to RACH.

Back-off Adjustment Schemes: In this approach, different back-off timers are assigned to different service classes in order to postpone their access attempts [30] [18] [11] [31] when devices of a class experience RACH access failure. However, this method cannot cope with peak congestion scenarios [28], and leads to large access delays.

Slotted Access: In this category, dedicated RA slots are allocated to each M2M device to access the eNodeB, where these slots are determined based on device identity and a parameter called RA cycle [9] [13], or based on service classes [15]. However, to support a large number of devices, a large RA cycle is needed, which leads to large access delays. In [32], devices that have experienced collision at RACH access are allocated with separate sets of preambles in their next transmission attempt, which may also be non-scalable when high congestion is experienced, leading to a large number of devices experiencing collision.

Distributed Queueing: With this technique, devices are provided with implicit knowledge as to when to make preamble transmissions based on (i) virtual distributed queues (DQ) and (ii) device locations at these queues [33] [14]. DQ is essentially a random access method in low load conditions, and a reservation access protocol in higher load cases.

Access Class Barring (ACB) [34] [25]: This method bans devices’ access to the eNodeB for a period of time, depending

Fig. 1. An example RAN slicing strategy, illustrated for 3GPP-defined use cases: massive-IoT (mIoT), Ultra Reliable Low Latency Communications (URLLC), and enhanced Mobile BroadBand (eMBB)
on congestion conditions. In [4], 16 different service classes are defined, and the eNodeB delivers a probability factor and a barring timer assigned differently to these classes. At times of congestion, the eNodeB then controls how much barring will be applied to each service class. However, this method can cause arbitrarily high time delay when high congestion occurs, as the devices may be banned for a long time [28].

Different variations of ACB have also been proposed. Extended Access Barring (EAB) [15] [9] [18] is an extension of ACB, specifically applied for M2M devices [30] [15]. In [15] and [35], Dynamic Access Barring (DAB) is applied where device access to the eNodeB is barred based on network load conditions. In [36], the ACB parameter is modified based on available resource blocks and access load from M2M devices. Another related technique is Cooperative Access Class Barring [37], where ACB parameters of multiple eNodeBs are optimized based on congestion levels; however, this relies on each M2M device to be in the coverage of multiple eNodeBs at the same time. In [38], a joint use of ACB and Timing Advance (TA) is shown to reduce the number of RA slots necessary to serve M2M devices.

Priorised Random Access: This includes separation of Random Access resources to be assigned to different traffic classes, namely H2H and M2M [15]. Prioritisation is either in favour of M2M/H2H traffics [39], or by means of multiple levels of prioritisation, such as low priority, high priority, scheduled, and emergency [15]. The technique also includes dynamic access barring, which bars newly arriving M2M devices’ access to the eNodeB. Similarly, low-priority and high priority M2M service classes are assigned with different persistence levels when performing channel access [13]. This type of scheme is reported to be better than other EAB methods in terms of average access delay and probability of success; however it still requires banning a M2M device for a period of time.

The massive access problem has also been addressed and studied from various other perspectives. Resource allocation is the most prevalent one [40], with solutions such as distributed algorithms [41] and game theoretic division of resources [42]. Device grouping based approaches have been proposed in [43] [44] [45]; a resource coordination scheme where multiple devices with correlated access patterns are grouped in presented in [43], and in another approach, devices are clustered [44] based on their desired Quality of Service [45]. In [38], fixed-location devices use timing advance (TA) information and may decide to defer from message transmission to avoid potential collision. In [46], Machine-Type Communications (MTC) devices send a signature that is constructed using a Bloom Filtering method, which can carry device identity and connection cause; the technique is shown to provide shorter connection times than LTE. Authentication procedures can also be embedded in this solution [47]. Finally, energy consumption considerations in LTE random access are studied in [24].

III. PRELIMINARIES

Our previous study [2] demonstrates that making random access based on multiple preamble transmissions, as introduced in CeRA [1], can achieve support for a much larger number of simultaneous RA attempts than what LTE can support, but only if the number of preambles in the preamble subset is suitably chosen, i.e. the preamble subset size. With a suitably chosen preamble subset size, CeRA is referred to as Multi-preamble RA. The study also provides a mathematical method to estimate this optimum preamble subset size based on the load received (in high load conditions), which is the number of new arrivals, i.e. the average number of devices starting their RA procedure per RA time.

In the following, first, the concepts of RA time and RA intensity are introduced, for both single and multiple preamble based random access. Then, the expression for probability of success in RA in a single attempt, as derived in [2] is presented. Finally, the concept of preamble set separation, i.e. preamble subsets, in the context of multiple preamble based RA is introduced.

A. Random Access intensity and RA subframes

RA load, i.e. access intensity, is defined as the number of devices making a new RA attempt per RA subframe; in LTE PRACH configuration 6 (PRACH 6), subframes 1 and 6 are RA subframes. For instance, with PRACH 6, if the access intensity is 10, this corresponds to 20 new RA attempts per frame, and 200 new RA attempts per second.

In the following definitions, both CeRA and Multi-preamble RA perform two preamble transmissions as a codeword over two successive subframes. Devices may perform a new RA attempt starting at subframes 1 or 6 (labelled as the First RA subframe, ‘F’); whereas subframes 2 and 7 are used for the transmission of the second preamble of the preamble sequence that a device chooses (labelled as the Second RA subframe, ‘S’). This is illustrated in Figure 2. For instance, if a device decides to make an RA attempt during subframe 8 of a frame, it waits until subframe 1 of the next frame and picks one preamble from the preamble subset of size a, then transmits this preamble. Then, at subframe 2, it makes another random preamble choice out of the same preamble subset of size a, and then transmits this second preamble.

For the sake of fairness, LTE is provided with additional two RA subframes in PRACH 6, i.e. subframes 2 and 7. Devices may perform an RA at one of the subframes 1, 2, 6, or 7. This is illustrated in Figure 3.
RA, it denotes a number of consecutive RA subframes during which devices make transmission of the sequence of their chosen preambles, i.e. codewords.

For CeRA and Multi-preamble RA with codeword length 2, an RA time is a sequence of 2 RA subframes marked as F and S, as in Figure 2. For LTE, and RA time is equivalent to one of the RA subframes at which a device makes its own preamble transmission, which is marked as P in Figure 3.

B. Probability of RA success in a single attempt

In [2], the probability of RA success achieved by a device in a single RA attempt using Multi-preamble RA is derived as:

$$P_s(a, m, n, R) = \begin{cases} f_1 = \left(1 - \frac{1}{a^n}\right)^{m-1}, & \text{if } R \geq \sqrt[n]{N} \\ f_2 = \left(1 - \frac{1}{a^n}\right)^{m-1} \frac{R}{N}, & \text{if } R \leq \sqrt[n]{N} \end{cases}$$

(1)

In (1), variable $a$ denotes the preamble subset size, $m$ is the number of new arrivals (RA intensity) per RA time, $R$ is the number of uplink resources available per RA time, $n$ is the codeword length, and $N$ is the average number of observed unique preambles by the eNodeB at a single preamble transmission subframe. In high load conditions, the probability of success expression in (1) is a piecewise probability distribution function with two sub-functions $f_1$ and $f_2$, meeting at a point $a = a^*$, such that $\sqrt[n]{R} = \sqrt[n]{N} = a^*(1 - (1 - 1/a^*)^m)$. The conclusion of the study in [2] is that when there is high RA load, the maximum probability of RA success in a single RA attempt can be achieved if the preamble subset size is chosen to be around $a = a^*$; the expression is valid for a sufficiently large RA intensity $m$, which is observed when $N = a^*$, (the number of observed unique preambles $N$ is equal to the preamble subset size $a^*$), i.e. all available $a$ preambles are chosen by at least one device. As a quick reference for the rest of the paper, Table I below lists the mathematical symbols in use.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Mathematical symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>Preamble set size</td>
</tr>
<tr>
<td>$m$</td>
<td>Number of new arrivals per RA time</td>
</tr>
<tr>
<td>$M$</td>
<td>Total number of devices at an RA time</td>
</tr>
<tr>
<td>$n$</td>
<td>Codeword length</td>
</tr>
<tr>
<td>$R$</td>
<td>Number of uplink resources available per RA time</td>
</tr>
<tr>
<td>$N$</td>
<td>Average number of observed unique preambles at each single preamble submission time</td>
</tr>
<tr>
<td>$P_s$</td>
<td>Probability of success in a single RA attempt</td>
</tr>
</tbody>
</table>

In Figure 4 (as presented in [2]), the case of single-shot random access with a single preamble (where codeword length is $n = 1$, as in LTE) and the case of single-shot random access performed by Multi-preamble RA (referred to as Multi($n$), with $n = 2, 3, 4$) are compared for their success probability. The analytical expression in (1) has been verified by Monte-Carlo simulations, providing average results of 10000 repetitions. The arrival rate to the system (new RA attempts) is $m$ devices per RA subframe. The figure demonstrates examples of the optimum value of preamble subset size $a = a^*$ where the RA success probability in a single RA attempt is the largest, for two high-load scenarios of $m = 50$ and $m = 100$ devices attempting RA per RA time. In other words, instead of using the whole preamble set of 54 preambles that is typically reserved for contention-based RA in LTE, the use of a much smaller subset of preambles provides the highest RA success probability. This motivates the use of a preamble subset (rather than a large set of preambles) when making RA with multiple preamble transmissions as in Multi-preamble RA.

C. Capability to separate preambles into subsets

The results presented in Section III-B imply that if a certain service class, e.g. IoT devices, are given a preamble subset, it is then possible to provide a separate subset to another service class, e.g. mobile broadband. As a result, in Multi-preamble RA, devices can use a specific preamble subset based on their device profiles and traffic characteristics, without sacrificing collision probability. For instance, periodically occurring IoT traffic generated by smart meters and smart home devices could use a single preamble subset, whereas smart phones that initiate Human-to-human (H2H) traffic can be given another
subset [30] [18] [17]. Eventually, preamble set separation could effectively be achieved when Multi-preamble RA is used. This motivates the use of Multi-preamble RA, in order to isolate RA attempts from different vertical markets, which is necessary for defining RA as a RAN network function and for further defining it as a virtual network function (VNF). An example of preamble set separation is illustrated in Figure 5.

IV. OPERATION OF MULTI-PREAMBLE RA

Based on the analysis result and the motivation in the previous section, this section presents the proposed RA technique, Multi-preamble RA. It provides feedback control based adaptation of: (i) the number of preambles \( a \) (preamble subset size), and (ii) the minimum number of resources \( R \) required, so that dynamically changing access intensity conditions can be supported for each service class. In doing so, Multi-preamble RA aims to achieve high RA success probability so that a large number of simultaneous RA attempts generated by a service class can be effectively supported.

Multi-preamble RA is based on estimation of the access load, i.e. the number of devices making RA attempt, and then adjusting the preamble subset size based on this estimation. The new preamble subset size is then announced to devices, which make preamble selections out of a new subset of preambles. Figure 6 demonstrates this feedback control loop. In this figure, codeword length is 2, i.e. devices make two back-to-back preamble transmissions. The number of distinct preambles that the eNodeB observes in the first and second RA subframes are denoted by \( N_1 \) and \( N_2 \), respectively. Using the average value, \( N = (N_1 + N_2)/2 \) and the value of preamble subset size \( a \), the eNodeB then estimates the RA load as \( M_{est} \), which is the number of devices that have made an RA attempt (these devices have chosen \( N_1 \) and \( N_2 \) distinct preambles over two consecutive RA subframes). Then, using a cached recent history of estimation values, the next value of RA load \( M_{next} \) is predicted. This is then translated into a prediction for the next value of the number of observed distinct preambles \( N_{next} \). Using \( N_{next} \), a new value for the preamble subset size is determined as \( a = a_{new} \), which is then announced to devices, to be the preamble subset size for the upcoming RA time.

A. Estimating the access load, \( M \)

As seen in Figure 6, Multi-preamble RA requires capability to perform the following operations:

1) Estimation of the number of devices attempting RA (access load), using preamble subset size \( a \) and the number of observed distinct preambles at eNodeB \( N \leq a \),
2) Setting the preamble subset size according to the current access load conditions,
3) Prediction of the next value of access load, based on a cached history of access load conditions,

In the following sections, the procedures to perform these operations are explained in detail. Codeword length is considered to be \( n = 2 \).

As an essential feature of Multi-preamble RA is estimation of the current access intensity level (RACH load). The eNodeB can make such an estimation based on a simple observation: the number \( N \) of distinct preambles it receives [48] [49], which can be denoted by \( N_1 \) and \( N_2 \) for the two consecutive preamble transmission RA subframes (when codeword length is 2). As the same process is carried out independently from each other and back to back, \( N_1 \) and \( N_2 \) are two independent samples of the same experiment. Hence, the eNodeB simply computes \( N = \lceil (N_1 + N_2)/2 \rceil \) to take the average of the two.

The estimation method is based on a maximum likelihood approach, which maximises the probability of having \( M \) devices making an attempt, given that \( N \) distinct preambles have been observed out of \( a \) available preambles, which is denoted by \( P(M|N,a) \). This probability is equal to the following:

\[
P(M|N,a) = \frac{P(N|M,a)}{\sum_{k=1}^{\infty} P(N|k,a)}.
\] (2)

In this equation, the denominator is the same for all possible cases of \( k \in \{1,2,3,\ldots,\infty\} \). Hence, maximising \( P(M|N,a) \) in Eqn. 2 is equivalent to maximising \( P(N|M,a) \), which is given by the following expression:

\[
P(N|M,a) = C(N \choose a) \left( \frac{N}{a} \right)^2 P_{cov}(M,N),
\] (3)

where \( P_{cov}(X,Y) \) is the probability that \( X \) users end up choosing all \( Y \) preambles (\( Y \) distinct preambles appear), when each user makes a random preamble choice out of the \( Y \) available. In other words, it is the probability that \( X \) users “cover” \( Y \) preambles. \( P_{cov}(X,Y) \) is computed by:

\[
P_{cov}(X,Y) = \sum_{k=1}^{X - Y + 1} C \left( \frac{X}{k} \right) \frac{1}{y} 1 - \frac{1}{y} X - k P_{cov}(X-k,Y-1),
\] (4)
which has the following boundary conditions:
\[ P_{\text{cov}}(k, 1) = 1, k \geq 1; P_{\text{cov}}(x, x) = \frac{x!}{x^x}; P_{\text{cov}}(x, y) = 0, \text{ if } x < y. \] (5)

The values of \( P_{\text{cov}}() \) are computed offline, for value ranges of \( 1 \leq a \leq a_{\text{max}} \) and \( 1 \leq M \leq M_{\text{max}} \), where \( a_{\text{max}} = 54 \) and \( M_{\text{max}} \) is some maximum value set for \( M \) (e.g. \( M_{\text{max}} = 500 \)). With this, a static table is obtained for \( P_{\text{cov}} \), with size \( a_{\text{max}} \times M \); a simplified version is shown in Fig. 7.

![Fig. 7. The static \( P_{\text{cov}} \) referral table.](image)

The combinatorics term \( C(N, 2) \) in Eqn. 3 is common for all \( M \), and since \( P(N|M, a) \) is compared for different \( M \), this term is dropped and not computed.

**Algorithm 1** Estimation of number of RACH access attempts, based on number of observed distinct preambles

1. Procedure ESTIMATE NUMBER OF ATTEMPTS(\( N_1, N_2, a \))
2. \( N = \frac{N_1 + N_2}{2}; N_{\text{max}} = \text{max}(N_1, N_2); \)
3. \( r = N/a; \)
4. \( P = 1; P_{\text{max}} = 1; M_{\text{est}} = N_{\text{max}}; \)
5. for \( M = N_{\text{max}} : M_{\text{max}} \) do
6. \( p_{\text{cov}} = \text{fetch}(P_{\text{cov}}(M_{\text{max}}, N)); \)
7. \( P = P_{\text{cov}} \times P; \)
8. if \( P > P_{\text{max}} \) then
9. \( M_{\text{est}} = M_{\text{max}}; P_{\text{max}} = P; \)
10. end if
11. end for
12. return \( M_{\text{est}}; \)
13. end procedure

To be able to accurately estimate the number of access attempts \( M \), the observed number of distinct preambles \( N \) should be less than the available number of preambles \( a \); otherwise there could be an arbitrarily large number of access attempts \( M \), which makes it impossible to estimate \( M \) with a maximum likelihood estimator. Hence, for the estimator to be operational, \( a > N \) must hold.

The algorithm that estimates the number of access attempts \( M \) based on this analysis is provided in Algorithm 1. The algorithm has two inputs: the two numbers of distinct preambles \( N_1 \) and \( N_2 \) that are observed over the two consecutive preamble transmissions, and the currently announced preamble set size ‘\( a \).’

Here, function \text{fetch} denotes a fetch operation from the stored static \( P_{\text{cov}} \) matrix, which has dimensions of \( M_{\text{max}} \) and \( N_{\text{max}} \) (see Eqn. 4 and Fig. 7). The algorithm computes Eqn. 3 for different \( M \) (Note that the term \( C(N, 2) \) is dropped, as explained in Section IV-A), and returns \( M_{\text{est}} = M_{\text{max}} \) that provides the highest probability.

The algorithm is run off-line, and has a time complexity of \( O(M_{\text{max}}) \). Using different combinations of \( N \) and \( a \) (\( N \leq a \)), an estimation matrix \( E_{a_{\text{max}} \times N_{\text{max}}} \) is computed and stored. At each RA time, upon reception of the two sets of preambles, the estimate \( M_{\text{est}} \) of the number of attempts is determined by fetching the corresponding value from this matrix for the values of \( a \) and \( N \) of that RA time, i.e. \( M_{\text{est}} = E(N, a) \).

**B. Choosing the number of preambles \( a \) in varying load conditions**

The analysis result \( a = a^* \) is found to be the best value for the preamble subset size that would maximise the probability of RA success in a single RA attempt when there is high load on RACH [2]. On the other hand, in a system with varying levels of system load, it is necessary to test if such a simple strategy would provide continuous effective performance, even for low load cases and when the load varies over time. Towards this, in this new exercise presented below, the intention is to analyse how the preamble subset size \( a \) should be adjusted when the access intensity level changes over time. Hence, this is a numerical study to observe what the value of \( a \) should be for a given intensity level \( m \).

The strategy followed for setting the preamble subset size \( a \) can be outlined as follows. Picking a large preamble subset size \( a \) reduces collision probability (probability that a code-word is picked by multiple devices). However, if the preamble subset size \( a \) is too large, this results in resource non-allocation events: the eNodeB falsely concludes that there is a much larger set of possible preamble sequence combinations than what is actually selected by devices, and randomly assigns its available resources to a subset of this large set of combinations. This results in many resources to be allocated to those combinations that actually have not been picked by devices. Resource non-allocation events occurs particularly when there is a large number of devices attempting random access, and the resulting observed number of distinct preambles \( N \) is larger than \( \sqrt{R} \), where \( R \) is the number of uplink resources available at each RA time. For more details on non-allocation events, please see [2]. In summary, there is essentially a trade-off between collision probability and resource non-allocation, as outlined in [2].

In the following analysis, three different strategies are considered:

1. Choose the preamble subset size \( a \) such that the probability of success \( P_5 \) (see Equation 1) is maximised (denoted by \( a_{\text{max}}(P_5) \)),
2. Set \( a = N + 1 > N \) to ensure that it is always possible to estimate the load \( m \), and
3. Set \( a \approx a^* = \sqrt{R} \), which is the estimation valid in high load conditions, as found in [2].

To evaluate the performance of these three different strategies for varying access load conditions, the access load \( m \) over time is modelled a time-varying curve based on a beta probability distribution function with parameters of \( \alpha = 3 \), \( \beta = 4 \), within a 10-second time window. Such choice of access load is due to the recommendation made by 3GPP for simulations of access load [9]. The average arrival rate

---

2169-3536 (c) 2018 IEEE. Translations and content mining are permitted for academic research only. Personal use is also permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
of devices making RA attempt for the first time is chosen as ≈ 65 new RA attempts per RA time.

Fig. 8 demonstrates the resulting values of the preamble subset size \(a\) obtained with these three different strategies, as well as the resulting observed number of unique preambles \(N\) chosen by devices out of a total of \(a\) preambles, i.e. \(N \leq a\). As seen in the top plot of Figure 8, the strategy to maximise \(P_s\) (this curve is represented as \(a_{\text{max}}(P_s)\)) results in \(a = 54\) initially (the maximum \(a\) possible) when the access intensity is low. By setting a large \(a\), collision probability is minimised, and there are no non-allocations at this time, since \(N < \sqrt{R}\). This region is marked by A in the figure. As the access intensity increases, \(N\) starts to grow (bottom plot in Fig. 8) due to the increasing intensity, but it is still less than \(\sqrt{R}\) while \(a = 54\).

Then, after a certain access intensity (encircled point marked by P1), ‘\(a\)’ gets modified (region B) such that \(N\) does not exceed \(\sqrt{R}\) and non-allocations do not occur, but \(a\) is kept as large as possible to keep the collision probability low, which results in \(N \approx \sqrt{R}\). This is observed in the bottom plot, and continues up to a certain access intensity point (encircled point marked by P2), after which \(a = \sqrt{R}\) is observed (region C), as this provides the maximum possible \(P_s\) (further decrease in \(a\) would dramatically increase collision probability). Note that, for high access load conditions, \(a = \sqrt{R} = a^*\) is the estimation for the best value of the preamble subset size \(a\) [2].

The reverse events occur when the access intensity decreases; the corresponding regions A’ and B’ are marked in the figure. The bottom curve in this figure also shows the observed number of distinct preambles \(N\) by the eNodeB. The figure shows values for the third strategy, which sets \(a = a^*\) (which would correspond to a straight horizontal line in Figure 8, and not shown for clarity reasons).

Fig. 9 shows interesting trends in the resulting values of \(P_s\) (probability of success at a single RA attempt) with the three strategies. Setting ‘\(a = N + 1\)’ so that estimation of the access load \(M\) is always possible (as mentioned as a condition, \(a > N\), in the previous section, Section IV-A) always results in the smallest obtained value of \(P_s\) among the three strategies. As expected, the \(a_{\text{max}}(P_s)\) strategy (that selects \(a\) to maximise \(P_s\)) consistently provides the highest \(P_s\) among the three strategies.

The critical point P2 of Fig. 8 appears to be a threshold for the estimation of \(M\) in the lower plot of Fig 9. Note that when \(a = N + 1\) strategy is followed, it is possible to estimate \(M\) throughout, but only up to point P2 if the \(a_{\text{max}}(P_s)\) strategy is used. The reason why point P2 represents a threshold is the following: Region B in Fig. 8 represents \(N \approx \sqrt{R}\), i.e. \(a = \left(1 - \left(1 - \frac{1}{a}\right)^{\frac{1}{2}}\right)\) ≈ \(\sqrt{R}\). After point P2, \(a = \sqrt{R}\) is observed; hence point P2 is where \(N \approx \sqrt{R} = a\), which means the access intensity starts to become high enough to “saturate” \(a\), such that \(a \approx N\), and it is no longer possible to estimate \(M\) (recall that \(a > N\) is required for estimation of \(M\)). Note that in [2] this corresponds to the finding that \(a = a^* = \sqrt{R}\) maximises the probability of RA success in a single attempt, where there exists a sufficiently high access load.

The second strategy always sets \(a = N + 1\) (the least value of \(a\) to ensure that it is possible to estimate the cumulative load \(M\)). At point P2, this is slightly above what Region B of the \(a_{\text{max}}(P_s)\) curve provides: \(N \approx \sqrt{R} = a\); hence the two curves have their intersection almost at P2 (slightly before P2 and then slightly after P2'). Based on this, point P2 can be approximately defined as the point where the curve \(a = \left(1 - \left(1 - \frac{1}{a}\right)^{\frac{1}{2}}\right)\) approximates \(\sqrt{R}\) and \(a = N + 1\) intersect, which gives:

\[
M^* = \frac{\ln(n + 1)}{\ln(n + 1) - \ln(n)}.
\]

Based on the findings from Fig. 8 and Fig. 9, a simple strategy to set the number of preambles \(a\) is defined as follows:

- When \(N < a\), set \(a = N + 1\). This is sufficient, because the obtained success probability \(P_s\) with the strategy \(a = N + 1\) is not too far off from those obtained with the
strategies $q_{\text{max}}(P_b)$ and $a = a^*$. This choice provides a minimal preamble subset size $a$, which is desired.

- Once $N \approx a$ has been reached, set $a = \sqrt{N} = a^*$. At this point, the probability of success $P_b$ achieved by both of the strategies $q_{\text{max}}(P_b)$ and $a = a^*$ make this same selection for $a$ (region C). On the other hand, the strategy $a = N + 1$ has a significantly low probability of success $P_b$. This provides minimal $a$ at low and moderate access intensity levels (regions A and B) and maximal $P_b$ when access intensity is high and when the cumulative load $M$ can no longer be estimated.

C. Prediction of upcoming access intensity

The third capability that Multi-preamble RA requires is to predict the access intensity to occur at the upcoming RA time. The Multi-preamble RA technique keeps a history of its past estimates of access intensity made over the most recent $W$ RA times, and then predicts the intensity to occur in the next RA time [21]. Hence, based on this, the preamble subset size ‘$a$’ to be made available at the next RA time is determined.

Algorithm 2 outlines the procedure, which first computes averages of each consecutive estimation value-pair in the set of $W$ recent values, and then replaces the original set with these averages, and finally re-iterates the same operation until only two values remain in the set, at which point a linear estimation is made. The algorithm has a time complexity of $O(W \log(W))$, and $W = 10$ has been used in performance evaluations (Section V). Note that this is a custom estimator, and can easily be replaced with a much more accurate and/or simpler one.

### Algorithm 2 Estimation of upcoming access intensity, $M_{next}$

1. $T[1,\ldots,W]$; Recent discrete RA time instances (up to $W$)
2. $e[1,\ldots,W]$; Recent access intensity estimates (up to $W$)
3. $\text{numel}(e)$: Number of elements operator
4. $\text{procedure}$ PREDICTACCESSINTENSITY($e$, $T$)
5. $\text{end procedure}$
6. $\text{procedure}$ PREDICTACCESSINTENSITY($e$, $T$)
7. $\text{while}$ numel($e$) > 2 $\text{do}$
8. $\text{for}$ $j = 1$; $j \leq \text{numel}(e)$; $j = j + 1$ $\text{do}$
9. $e_{\text{new}}[i] = (e[j] + e[j + 1]) / 2$
10. $T_{\text{new}}[i] = (T[j] + T[j + 1]) / 2$
11. $\text{end for}$
12. $\text{end while}$
13. $\text{return}$ $M_{next}$
14. $\text{end procedure}$

D. Setting the new value of the preamble subset size, $a$

Multi-preamble RA adapts the preamble subset size to be used in the upcoming RA time, using its recent history of estimated access intensity values (which have been predicted with the recent history of observed average $N$ and the current preamble subset size $a$). Algorithm 3 outlines the procedure, where the strategy to update $a$ depends on whether $N$ has “saturated” $a$ or not (i.e. $N = a$). Definitions of the variables used in this algorithm are provided at the beginning of the algorithm.

In the saturation case, i.e. $N = a$ (line 25), if this saturation event has happened for a Limit$_{N=a}$ instances, then $a$ is exponentially increased (line 30), otherwise it is linearly incremented ($a = N + 1$, line 28).

In the non-saturation case ($N < a$) (line 3), after updating the set of estimates $e[]$ by entering the most recent one ($M_{est}$) (lines 4 to 10), the next access intensity level $M_{next}$ is predicted (Alg. 2). Based on $M_{next}$ and the current value of $a$, the algorithm then predicts the number of distinct preambles to be observed $N_{est}$ (Lines 13 and 15), by $N_{est} = a(1 - (1 - 1/a)M_{\text{next}})$ as in [2]). If $N_{est}$ is close to $a$, which is represented as $N \geq 0.99a$, then a new value of $a$ that would provide around $N_{est} \approx 0.9a_{\text{new}}$ is chosen. The choice of 0.9 is arbitrary and provides a large enough $N$ which does not saturate $a$ completely. In the algorithm, the term $M_{\text{next}}\sqrt{a}$ is computed off-line, and stored as a hash-map that is keyed by different values of $1 < M < M_{\text{max}}$.

If a saturation condition is not predicted, but the number of observed distinct preambles is predicted to decline (line 20), as compared to the previously predicted value $N_{estprev}$, then $a_{\text{new}}$ is set to $N_{est} + 1$, which is a value just enough to avoid saturation; and $N_{estprev}$ is updated with its new value.
Finally, if $a_{\text{new}}$ exceeds $a_{\text{max}}$, it is set to this maximum. The value of $a_{\text{max}}$ depends on the resource allocation speed $R$; which is dynamically adapted as outlined next in Section IV-E.

The algorithm has a time complexity of $O(\log(W)) + O(M_{\text{max}})$, where $1 \leq M \leq M_{\text{max}}$ is the access intensity and $W$ is as in Section IV-C.

Figure 10 shows a flowchart representation of Algorithm 3 which outlines the adaptation of preamble subset size $a$.

![Fig. 10. Detailed flowchart of the algorithm for setting preamble subset size in Multi-preamble RA.](image)

### E. Adaptation of the number of uplink resources $R$

Based on the estimated access intensity ($M_{\text{est}}$ in Alg. 1), Multi-preamble RA adjusts its number of uplink resources $R$, which is to be reserved for a given service class. By definition, $R$ can take values up to an absolute maximum of $R_{\text{max}}$, which stands for the maximum possible number of resources that the system can ever support at a time for that service class. A minimum is also considered, which is $R_{\text{min}} = 1$.

The procedure is simple, and aims to achieve at least a minimum required success probability $P_{\text{suc}}^{\text{req}}$ at a single RA attempt. The number of all possible preamble combinations $(N \times N_2$, as observed by the eNodeB) is compared to what can actually be allocated at a time ($R$). Based on this, the achievable success probability is estimated as $P_{\text{suc}}$ for the current value of $R$, and as $P_{\text{suc}}^{\text{req}}$ for a decremented value $R - 1$. $R$ is either incremented to be $R + 1$, or decremented to be $R - 1$, so as to meet the $P_{\text{suc}}^{\text{req}}$ requirement. The algorithm has a time complexity of $O(M_{\text{max}})$, and $1 \leq M_{\text{est}} \leq M_{\text{max}}$.

The target success requirement $P_{\text{suc}}^{\text{req}}$ is chosen such that an average of 99% RA success probability can be achieved in a maximum of $K$ successive attempts. According to repeated Bernolli trial principles, the success probability expression is derived as $P_{\text{suc}}^{\text{req}} = 1 - KL^{K}_{0.01}$.

### F. The main algorithm: Operation of Multi-preamble RA

The main adaptation algorithm of Multi-preamble RA is outlined in Algorithm 5, which runs at each RA time (consisting of two consecutive preamble transmissions). Here, $\alpha'$ represents the preamble subset size that has been announced to the devices of the service class, where the maximum possible preamble set size is denoted by $a_{\text{max}}$. In other words, although a maximum of $a_{\text{max}}$ is considered at a time, only part of it is announced to devices, and adjusted over time.

If the traffic load is high, Multi-preamble RA may need to adjust the set size $a_{\text{max}}$ up to a limit of $a_{\text{limit}}$ (e.g. if there are no other services classes present, then $a_{\text{limit}} = 54$ in LTE, otherwise $a_{\text{limit}} < 54$), which is triggered by a need for increasing the number of uplink resources $R$ either in non-saturation conditions ($N < a$) (line 5) or when the maximum set size has been used for a consecutive Limit$_{\text{limit}}$ times (line 11). The adaptation of the available number of preambles $a$ within the set $a_{\text{max}}$ is performed once in every adaptation time interval, $T_{\text{adapt}}$ (line 24), i.e. $a$ is adapted once in every $T_{\text{adapt}}$ RA times. Initial values are chosen as $R = 3$ and $a = 3$. The algorithm has a time complexity of $O(\log(W)) + O(M_{\text{max}})$, where $W$ is as defined in Section IV-C, and $1 \leq M \leq M_{\text{max}}$.

### G. Dual-mode of operation

In Multi-preamble RA, devices operate in one of the following two modes of operation:

1. **Single-preamble (SP) mode**: A device makes a randomly picked single preamble transmission. This is the mode of operation LTE devices operate at.

2. **Multi-preamble (MP) mode**: A device makes a sequence of single preamble transmissions, each selected randomly out of a preamble subset of size $a$, where $a < 54$ considering the whole set of preambles reserved for contention-based RA is 54.

While in the SP mode, devices of a service class make a single preamble transmission at each of their RA attempts, as in LTE. The eNodeB monitors the observed number of distinct preambles $N$. If $N = a$ for a number of $X$ consecutive times, then the eNodeB decides that RACH is congested for this service class for their allocated preamble subset size $a$. Once
Algorithm 5 Multi-preamble RA main algorithm.

\[ \alpha_0: \text{Initial preamble subset size allocated for this service class} \]
\[ N_1, N_2: \text{The number of distinct preambles chosen at the two preamble transmissions} \]
\[ \text{(out of the announced set size of } a) \]
\[ a_{max}: \text{Maximum preamble set size for this service class} \]
\[ a < a_{max}: \text{Announced number of preambles to this service class} \]
\[ Count_a = a_{max}: \text{Number of times that the preamble subset size has reached} \]
\[ \text{the current value of the maximum preamble subset size allowed} \]
\[ Limit_{a_{max}}: \text{The necessary consecutive number of times that } a = a_{max} \]
\[ \text{happens before incrementing } R. \]

1: procedure MULTI-PREAMBLE RA(\(a_0\))
2: \hspace*{1em} At each RA time, eNodeB observes \(N_1\) and \(N_2\).
3: \hspace*{1em} if \(N_1 \neq 0 \text{ AND } N_2 \neq 0\) then
4: \hspace*{2em} \(N = \text{Round}\left((N_1 + N_2)/2\right)\);
5: \hspace*{2em} slots = slots + 1;
6: \hspace*{2em} if \(N < a\) then \( \triangleright \) Estimation possible
7: \hspace*{2em} \(R = \text{AdapterResources}(m_{est}, N, a)\); \(\triangleright \) Algorithm 1
8: \hspace*{2em} \(a_{new} = \sqrt{R}\);
9: \hspace*{2em} else if \(a = a_{max}\) then \( \triangleright \) Reached maximum allowed preamble subset size
10: \hspace*{3em} \(Count_{a_{max}} = Count_{a_{max}} + 1;\)
11: \hspace*{3em} if \(Count_{a_{max}} = Limit_{a_{max}}\) then
12: \hspace*{4em} \(R_{new} = R + 1;\)
13: \hspace*{4em} if \(R_{new} > R_{max}\) then \(\triangleleft \) Algorithm 4
14: \hspace*{4em} \(R_{new} = R_{max};\)
15: \hspace*{3em} end if
16: \hspace*{3em} \(Count_{a_{max}} = 0;\)
17: \hspace*{3em} \(a_{max} = \sqrt{R_{new}};\) \(\triangleright \) Adjust maximum allowed preamble subset size
18: \hspace*{3em} end if
19: \hspace*{3em} end if
20: \hspace*{3em} if \(a_{max} > a_{limit}\) then
21: \hspace*{4em} \(a_{max} = a_{limit};\)
22: \hspace*{4em} end if
23: \hspace*{3em} if \(\text{rem}(\text{slots}, T_{adapt}) == 0\) then
24: \hspace*{4em} \(a_{new} = \text{SelectNumPreambles}(a, N, a_{max}, m_{est}, e)\); \(\triangleright \) Algorithm 3
25: \hspace*{4em} end if
26: \hspace*{3em} end if
27: end procedure

H. Practical realisation of Multi-preamble RA

In the following, some practical considerations are explained when a multiple preamble based random access technique is to be implemented in a mobile system.

1) Service Class ID: To realise separation of preambles into preamble subsets, each service class is to be assigned to a subset of preambles, and each service class is to be referred to using a service class ID (SCID). Service Class IDs are to be known by devices as well as the eNodeB. Preamble subsets are to be mutually exclusive. For example, a service class for delay-tolerant MTC and another another class for mobile broadband are to have different preamble subsets. This isolates the random access attempts made by different services; as a result, congestion in one service does not diversely affect others.

2) Adaptation of system parameters: The eNodeB is to adapt the number of preambles assigned to a service class, periodically, and then announce the preamble subsets to be used by different service classes. Modification is to be made only when necessary, and not all preamble subset sizes are to be necessarily modified each time. Announcements are to be made periodically in an SIB. For each preamble subset size to be modified, the eNodeB announces the following information triplet:

- Service Class ID (SCID)
- First index of preamble subset (FIPS)
- Preamble subset size (\(a\))

For instance, for 54 preambles reserved for contention-based RA (Preambles index as 1 to 54), the delay-tolerant MTC service class (say SCID=2) may receive 10 preambles, starting at preamble index 15, ending at preamble index 24. Then, the SIB is to include SCID=2, FIPS=15, \(a=10\).

To be able to modify system parameters when needed, the eNodeB is to continuously observe the number of distinct preambles (\(N\)) received at each RA subframe time for each service class, regardless of the RA mode that service class is operating on (SP or MP modes). For instance, if preambles 2, 5, 6, and 7 are received at an RA subframe (where these preambles belong to the subset used by a certain service class), then this means that four distinct preambles are received at that RA subframe. At each mode, the eNodeB is to keep a running average of the number of distinct preambles over a window of consecutive RA subframes, separate for each service class. When in the MP mode (i.e. at congestion times), the eNodeB must also estimate the total number of devices that made preamble transmissions belonging to that service class,
based on: (i) the number of distinct preambles received from the service class, and (ii) preamble subset size used by the class. Estimations can be made at every RA subframe, or less frequently but periodically.

The eNodeB is to modify the preamble subset to be assigned to a service class, based on the estimated number of devices attempting RA in that class. The modification of the preamble subset size is to consider maximisation of RA success probability for the given load on RACH from that service class, the current preamble subset size, and the uplink resources available at the time, and is based on Algorithm 5. Adaptation of preamble subset size can be made as frequently as desired, yet the announcements to devices are to be made once at each SIB period. Announcements are to include the information triplet (SCID, FIPS, \( a \)).

Since there is a fixed total number of preambles (i.e. 54), if the change of preamble subset size of a service class affects the starting index or size of other service classes, then the announcement is to include modification for affected classes as well. Adaptation of the preamble sizes of different classes are to be coordinated by the eNodeB; and different algorithms and restrictions can be applied when an expansion of a preamble set size is deemed not possible due to the subset sizes used by other classes. This aspect is considered as a topic of further research.

V. PERFORMANCE EVALUATION

In this section, the performance of Multi-preamble RA is evaluated and compared to those of LTE and CeRA.

A. Simulation settings

Performance evaluations are performed using a simulation environment developed in MATLAB [50], which keeps device states and follows the steps of random access (RA) defined by 3GPP [9], with typical RA parameter values as outlined in [51].

The codeword length in Multi-preamble RA and CeRA is \( n = 2 \). Besides sub-frames 1 and 6 as in PRACH configuration 6, LTE is given two additional RA subframes (subframes 2 and 7), so that it has an equal number of RACH opportunities as CeRA and Multi-preamble RA do.

Each device performs random access (RA) procedure once, and device RA time instances are randomly assigned to be one of the 2000 subframes\(^2\) in a 10-second time window; the arrival time distribution follows a stationary Poisson process with a mean inter-arrival time whose reciprocal is equal to the average access intensity. An RA procedure allows a device to make up to \( K = 10 \) RA attempts, after which RA is considered as a failure. In Multi-preamble RA and CeRA, devices do not perform back-off before re-attempting upon failure of a previous RA attempt.

Various access intensity levels are considered in simulations, ranging from 2 up to 100 new devices starting their RA procedure per RA subframe. A new device (hence a new RA procedure) is referred to as a new arrival event. PRACH Configuration 6 is considered, resulting in an RA periodicity of 5 subframes; as a result, for instance, for 20000 new arrivals accessing RACH within a 10 sec time window, the average access intensity is 10 per RA time. In LTE, devices make an RA attempt at the first upcoming RA subframe (see Fig. 3), whereas in Multi-preamble RA and CeRA, RA attempts are made at the first upcoming ‘F’ subframe, as shown in Fig. 2 in Section III-A.

In CeRA and Multi-preamble RA, each preamble transmission is completed within 1 subframe time (Format 0 with around \( \approx 800 \) \( \mu \text{sec} \) preambles), hence 2 consecutive transmissions take 2 subframes. There is a guard time interval of 2 subframes after the completion of the second preamble transmission (a 3-subframe wait until the Random Access Response (RAR) window starts [51] [4]). The RAR window is \( t_{\text{RAR}} = 5 \text{ms} \). During a RAR time window which follows the last preamble transmission in the sequence of preamble transmissions, one RAR is sent per subframe. RARs arrive at allocated devices at random subframes within the RAR window. Upon reception of a RAR, a processing delay of 5 ms is considered at devices [52].

In order to support the RA load, the number of uplink resources made available to devices per RA time is chosen to be equal to the access intensity level, in all three techniques. Hence, for an arrival rate of \( m \) devices per RA time, the number of resources \( R \) allocated to each RA time is \( R = m \), so that there is one resource for each device.

An HARQ time delay of 2 ms in considered, which includes around 1 ms delay for RRC Connection Request message transmission (Message 3) and an average of 0.8 ms for retransmissions [52]. The processing time delay before sending an RRC Connection Setup message (Message 5) is 12 ms [52]. Each message transmission between a device and the eNodeB takes 1 ms (one subframe time).

Finally, preamble detection probability at the eNodeB is \( 1 - \frac{1}{ek} \), where \( 1 \leq k \leq K \) is the trial number [9].

Simulation parameters are summarised in Table II.

B. Performance Results

The following key performance indicators (KPI) are evaluated for LTE, CeRA, and Multi-preamble RA:

- **Probability of RA success**: The probability that a device performing random access (with up to \( K = 10 \) attempts) succeed and is allocated with an uplink resource that no other device is allocated with.
- **RA throughput**: Number of successful devices per RA time.
- **N**: Average number of unique preambles observed by the eNodeB.
- **Preamble utilisation**: Average ratio of the observed unique preambles \( N \) to preamble subset size \( a \).
- **Random access time delay**: The time period from when a device decides to perform RA until it is successfully allocated with an uplink resource uniquely.

Performance results for these KPIs are presented separately. Results are the averages of 40 simulations in each simulation case.
Multi-preamble RA has been simulated for different values of its adaptation time interval, i.e. how often the preamble subset size is adapted, which are time intervals of 40, 80, 160, and 320 ms. In general, it is observed that adapting the preamble subset size more frequently achieves better performance in RA; yet of course this has a trade-off with how often devices need to be informed of the change in their allocated subset. In the following, the KPI results are presented in more detail.

1) Probability of RA success: Fig. 12 illustrates the comparison of Multi-preamble RA, LTE, and CeRA for various values of the access intensity level, which is the number of new RA procedures started by new devices per RA subframe time (new arrival density). It is observed that Multi-preamble RA can support higher access intensity levels, while LTE and CeRA cannot. LTE simply does not have sufficient code space (up to 54) which would have sufficiently low logical collision probability when devices pick their preambles. On the other hand, CeRA suffers from too much ambiguity, as it blindly uses all 54 preambles available to devices, while also using multiple preambles. In contrast, Multi-preamble RA adapts the number of preambles to be used, based on the estimated access intensity on the random access channel, by observing the number of unique preambles received at the eNodeB side. This makes it possible to reduce code ambiguity as much as possible, while still decreasing code collision probability.

2) RA throughput: Number of successful devices per RA time: Fig. 13 demonstrates that Multi-preamble RA achieves a higher throughput in random access, i.e. the number of devices succeeding in RA per attempt is higher than LTE and CeRA. The random access throughput increases linearly with respect to linearly increasing access intensity level, for the demonstrated set of access intensity levels.

3) Average number of observed distinct preambles by the eNodeB: Fig. 14 below shows that Multi-preamble RA uses much fewer preambles (increasing to around up to 15 preambles for the set of access intensity levels evaluated), whereas CeRA increments its preamble subset size all the way to 54 preambles. LTE uses preamble subset size (the average number of preambles picked by devices per attempt) also increases and is higher than Multi-preamble RA. Please note that LTE makes all the 54 preambles available devices, whereas Multi-preamble RA announces only a subset.

4) Preamble utilisation: As observed in Fig. 15, LTEs utilisation of preambles is only up to around 50-60% of the total 54, whereas Multi-preamble RAs utilisation is over 90%, i.e. the used preambles of up to 20 (as shown in Figure 5). This result shows that Multi-preamble RA is significantly efficient in its use of preambles. The implication of this result
is that different service classes can be allocated with mutually exclusive preamble subsets and can achieve high RA success.

5) Random access time delay: The time delay until a device successfully completes a random access procedure is another performance metric evaluated by simulations (The device gets an uplink resource, and delivers its Message 3 successfully, sends Message 4 and then receives Message 5.). Results are shown in Fig. 16. Time delay exponentially increases for LTE for increasing access intensity, as devices make more and more attempts before success. This then starts to decrease, which is an effect of having a very low proportion of devices actually achieving random access success; such devices are those that succeed in their first few attempts, as it becomes less likely for re-attempts to succeed, due to the accumulation of devices over time. The same is observed for Multi-preamble RA (yet with much higher success probability, as in Fig. 12), which can achieve an average time delay of around 40 ms, whereas CeRA has around 100 ms.

6) Minimum required resources to support a given access intensity: As mentioned in Section ??, when a sufficient resource allocation speed is provided, CeRA can achieve high probability of success. Accordingly, this section provides results on how fast the system should be able to provide resources (recall that \( R \) is the number of allocations per subframe time during a RAR window), so that a minimum of 90% success in RA is achieved. This is shown in Fig. 17. Multi-preamble RA adapts the preamble subset size every 40 ms. As observed in the figure, CeRA requires a significantly higher resource allocation speed, as compared to Multi-preamble RA. In other words, the adaptive allocation of resources in Multi-preamble RA makes it a far less resource-demanding scheme than CeRA. In the figure, the flat curve shown for LTE is due to the fact that higher \( R \) is redundant for LTE, as there is at most 54 resources required at each RA time (the full preamble set size is 54).
Multi-preamble RA adapts two system parameters of received distinct preambles at the eNodeB side at each RA load-aware scheme is driven by observations of the number of devices making RA attempts at a time. This concept proposed in CeRA and also adjusts preamble subset size, making it practically possible to isolate them in the access intensity that can no longer diversely affect others at times of heavy load, such as simultaneous connection requests made by a large number of IoT devices that have regular traffic service class, making it a suitable virtual network function, vPreamble, in future mobile networks. The paper introduces a concept in which each vPreamble instance can be associated with a service class and later modified according to traffic conditions of that service class, making it a suitable virtual network function (VNF) in a random access network (RAN) slice.

VI. CONCLUSION

A recently proposed technique CeRA [1] uses multiple preamble signals as a sequence so as to address the emerging issue of congestion on the random access channel (RACH) when multiple devices make simultaneous access attempts. In our previous work [2], we show that, for CeRA to be effective, the preamble set size must be chosen suitably, and provide a mathematical method to calculate the optimum set size that should be used in high access load conditions. The implication of this finding is that, when CeRA is used, different service classes may be associated with mutually exclusive preamble subsets, making it practically possible to isolate them in the random access (RA) procedure. Hence, congestion from one service class can no longer diversely affect others at times of heavy load, such as simultaneous connection requests made by a large number of IoT devices that have regular traffic generation patterns.

In this paper, based on the finding in [2], an adaptive RA scheme called Multi-preamble RA is presented, which uses the concept proposed in CeRA and also adjusts preamble subset size for a service class based on that class’ RA load, i.e. the number of devices making RA attempts at a time. This load-aware scheme is driven by observations of the number of received distinct preambles at the eNodeB side at each RA instance. Multi-preamble RA adapts two system parameters dynamically, based on RACH load conditions: (i) preamble subset size, (ii) the number of uplink resources. Performance results are obtained using system emulations that implement: (i) LTE random access procedure with back-offs, (ii) CeRA with no adaptation of preamble subset size, and (iii) Multi-Preamble RA. Results demonstrate that significant gains can be achieved by adapting these parameters in terms of: high success probability in RA for system loads as high as 100 attempts per RA time whilst using less than 20 preambles, a stable time delay of less than 40 ms even for high system load cases, and efficient use of the associated preamble subset via high utilisation.

Based on the presented results, it is concluded that the proposed adaptive scheme can effectively provide preamble separation among different service classes and support high RA load. This finding motivates virtualisation of the RAN preamble allocation functionality as a separate virtual network function, vPreamble in future mobile networks. The paper introduces a concept in which each vPreamble instance can be instantiated and later configured for a different vertical market (service class), such as IoT and mobile broadband. Parameters of a vPreamble instance can be associated with a service class and later modified according to traffic conditions of that service class, making it a suitable virtual network function (VNF) in a random access network (RAN) slice.

REFERENCES


