SUMMARY

Today, cellular networks rely on fixed collections of cells (tracking areas) for user equipment localisation. Locating users within these areas involves broadcast search (paging), which consumes radio bandwidth but reduces the user equipment signalling required for mobility management. Tracking areas are today manually configured, hard to adapt to local mobility and influence the load on several key resources in the network. We propose a decentralised and self-adaptive approach to mobility management based on a probabilistic model of local mobility. By estimating the parameters of this model from observations of user mobility collected online, we obtain a dynamic model from which we construct local neighbourhoods of cells where we are most likely to locate user equipment. We propose to replace the static tracking areas of current systems with neighbourhoods local to each cell. The model is also used to derive a multi-phase paging scheme, where the division of neighbourhood cells into consecutive phases balances response times and paging cost. The complete mechanism requires no manual tracking area configuration and performs localisation efficiently in terms of signalling and response times. Detailed simulations show that significant potential gains in localisation efficiency are possible while eliminating manual configuration of mobility management parameters. Variants of the proposal can be implemented within current (LTE) standards.

1. INTRODUCTION

Cellular networks need to keep track of where its users are located in order to relay incoming connections to the most suitable cell. Current implementations of this service use significant amounts of system and spectrum resources and require extensive manual configuration. We propose a mechanism that implements this service more efficiently than the current state of the art and autonomously adapts to changes in user mobility patterns.

1.1. Mobility management in current cellular networks

In LTE, mobility management is confined to associating terminals with their closest cell or a small group of adjacent cells. To this end user terminals—user equipment (UEs)—keep track of their closest cell at all times, whereas network nodes—mobility management entities (MMEs)—keep track of sets—tracking areas (TAs)—of possible cell associations. Figure 1(a) shows two typical adjacent and disjoint TAs.

The cells of the network are statically partitioned into TAs. Incoming connections are routed to the the MME which looks up a list of TAs associated with the requested UE, and initiates the localisation (paging) process. To find the exact cell at which a UE resides, the network broadcasts page messages in all cells of all TAs in the list associated with the UE and detects the cell from which a response is...
received. Figure 1(b) shows an example in which the three TAs to the left belong to the TAI list of the UE which are paged in parallel.

Cells broadcast their TA identifiers (TAIs), which allow the UEs to compare the TAIs of nearby cells to their current TAI lists. If a UE cannot find the TAI of its preferred cell in its TAI list, it performs a TA update to inform the MME of its new location and to obtain a new TAI list. Figure 1(c) shows a UE triggering an TA update as it passes into TA outside its TAI list.

Figure 1. Static tracking areas. (a) Two adjacent static tracking areas. (b) On an incoming connection a UE is paged in all cells of all TAs (the three leftmost ones) in its associated TAI list. (c) UE moving between several tracking areas belonging to its TAI list before detecting movement in a TA outside of it, and only then issuing a TA update.
Noting that large TAs and long TAI lists reduce the need for TA update messages while small TAs and short TAI lists reduce the need for paging messages, it is clear that optimal TAs and TAI lists are a matter of striking a balance between TA update messages (which imply large TAs and long TAI lists) and paging messages (which imply small TAs and short TAI lists). As noted by many authors [1–3], these choices can be critical to avoid problems like massive signalling at TA borders.

While the above description applies to LTE systems [4], similar concepts were used in 2G (GSM/GPRS etc.) and 3G (WCDMA/HSPA etc.) systems, where the network nodes and location sets are known as MSCs (mobile switching centres) and LAs (location areas) for circuit switching and SGSNs (serving GPRS support nodes) and RAs (routing areas) for packet switching, respectively. A major difference, however, is that the TAI lists in LTE correspond to single LAIs/RAIs in 2G/3G UEs. In our proposal, the TAI lists (or some similar facility) is essential to represent the dynamic local neighbourhoods we propose to use instead of static tracking areas for paging.

1.2. Mobility management overview

To solve the problem of efficiently managing UE mobility, several alternative update schemes and paging strategies have been proposed, but none takes the actual and dynamic collective mobility as observed online in the network into account as we propose. Some other updating schemes [5,6] propose that networks trace two and three static cell sets, respectively, for each UE. This is a very useful mechanism and a variant has been implemented in LTE in the form of TAI lists. We propose a completely self-organising mechanism that generalises this idea to arbitrary cell sets, construct, update and maintain them online, and a corresponding mechanism to page efficiently within them.

Other examples [1–3,7] propose static cell set construction from mobility or traffic data collected offline, or propose dynamic areas based on detected UE direction and velocity [8,9]. The former approach fails to adapt to dynamic changes in mobility patterns, while the latter takes no account of the influence of infrastructure and other geographical elements. Our proposal, in contrast, records the actual cells where UEs are successfully located and aggregates these statistics for large number of users.

A different approach is taken by several authors [9–14], where static cell sets are replaced by or combined with a maximum allowed distance from the last reported location. This requires the cells to broadcast global coordinates and modifications of UEs to use this information for update triggering, and has to our knowledge not been adapted in any current standards. In principle, such approaches suffer from the fact that distance alone is not always a good indicator of actual mobility, which is also heavily influenced by the presence of infrastructure and varying population densities, for example. Also, any proposal requiring extensive UE modifications will be hard to implement in practice once a system has gained momentum. There will always be older UEs that do not support the functionality introduced by such modifications.

Paging strategies based on ranking cells in a cell set has been proposed with various ranking mechanisms. Some authors [15–17] propose estimating the likelihood of a page response at a given location from UE-specific statistics. This could be achieved both statically and dynamically, but in both cases requires a large number of UE profiles to be collected and maintained. This type of statistic could also be considered sensitive w.r.t. privacy. Others [18–20] propose to combine or complement such profile-based approaches with traffic and network topology data, for example. The effort to collect such data from other sources and applying them to network management seems wasted when the data needed for estimating the local mobility patterns relevant to network management can, as we will show, be generated by the network itself, and with minimal impact on privacy.

Some authors [8,13] instead estimate the likelihood from current UE speed and/or direction, which require an online collection and dissemination mechanism, but is still based on individual mobility rather than collective. Such a mechanism could be more precise in some situations, but requires significant modifications to current protocols and UEs in addition to extensive user modelling. The management overhead required to implement such such schemes seems prohibitive.

1.3. Relevance and contribution

The motivation for this work is the increasing amount of work required to configure mobility management data in cellular networks. In particular, we note that, while this is a demanding task already in 2G and 3G
networks, it will be even more complex in 4G networks, which typically have more cells (due to smaller coverage), different types of cells (due to heterogeneous networks) and more complex functions (e.g., the TAI list concept). An equally important motivation is, however, the fact that mobility data are seldom used in this context although mobility patterns can have a significant influence on resource consumption and thus, potentially, on performance. We have, for example, seen cases where attempts at load balancing between MSCs have resulted in significantly increased load on all MSCs to the extent that almost half the load was related to location updates, and found that the reason for this was that the boundaries between the resulting, load-balanced location areas repeatedly intersected major roads.

The goal of our approach is thus to provide a completely self-organising solution that improves updating and paging in fourth-generation systems w.r.t. system efficiency and management complexity. Comparing LTE to the earlier proposals, it is noted that the two or three cell sets discussed elsewhere [5,6] have been adopted as the (now up to 16) cell sets allowed by the TAI lists in LTE, while the distance-, direction- and velocity-based approaches mentioned above have not been adopted at all. The use of UE-specific information for paging is not only complex to collect and maintain but also comes with significant privacy issues.

Our approach instead constructs dynamic local neighbourhoods based on collective mobility patterns as observed through successful localisation attempts logged in the mobility management entities (MMEs) of the network. The neighbourhoods, which can be unique for each cell, are used to form and maintain the TAI lists distributed to UEs. Aggregated location update rates in this approach depend directly on the joint probability represented by the cells denoted by the neighbourhood in our model. For any neighbourhood, multi-phase paging sequences are derived at cell level with the object of balancing the expected number of page messages against increases in response times. The resulting method, ‘zero configuration adaptive paging’ (zCap), is implementable in present standards with no UE modifications and fairly conservative changes to the mobility management mechanisms, while offering a self-tuning and distributed method to obtain efficient TAI lists and efficient paging of the cells denoted by the neighbourhoods. Experiments based on an advanced prototype and mobility simulations show the potential for a radical reduction of the number of page messages required to locate UEs. A short outline of the results presented here paper appears elsewhere [21].

1.4. Presentation layout

Each aspect of the approach will be described in more detail in the following sections. After a brief overview in Section 2.1 we discuss in Section 2.2 how to maintain distributed traces of cells recently associated with specific UEs; we describe in Section 2.3 how to disseminate information of successful UE localisations along the traces, and how to update counters to support local estimates of UE mobility; we show in Section 2.4 how such counters can be used to maintain, for each cell and time frame, a Bayesian estimate of the probability of UEs being localised at other cells, and how to discount older data at a rate which captures both stable and dynamic properties of local mobility patterns. In Section 2.5 we use the mobility estimates to form local neighbourhoods which are transferred as TAI lists to UEs and maintained by the MME of the cell. In Section 2.6 we show how to compute optimal paging sequences for any given neighbourhood, time frame and current estimate. Section 2.7 finally details the connection set-up and paging mechanism itself. Section 3 provides a summary of experimental results obtained from an advanced prototype implementation of the method in two mobility and connection simulations, after which we conclude with a short summary and discussion in Section 4.

2. THE ZCAP APPROACH TO MOBILITY MANAGEMENT

2.1. Overview

The main components of zCap are the mechanism to collect mobility statistics online, the probabilistic model of the local mobility in the neighbourhood of each cell, and the mechanism to estimate and use it to efficiently manage the services of the network. The mobility statistics collection mechanism we describe is fully distributed and scales from managing the statistics on the level of individual cells, up to, and between, the more centralised mobility management entities (MMEs) of current systems.
The collected statistics consists, for each cell $i$, of counters $n_{ij}$ of the number of successful localisations at cells $j$, of UEs associated with $i$ within a certain limited time from the UE’s initial association with $i$. This information is used to estimate the probability of locating a UE in each observed cell and to maintain a list of the most likely UE destinations around each cell. We refer to this list as the *local neighbourhood* (or simply *neighbourhood*) of each cell $i$ and update it dynamically as the estimate evolves over time using a Bayesian estimation scheme in which the estimates of earlier periods are used as a prior for the current estimate of the distribution. Scaling the window size of the estimator and updating a small number of parallel estimators $n_{ij}$ allows us to capture both stable and dynamic aspects of the local mobility, e.g. variations over the day or week. The stored representation of the estimate is very compact.

We propose to replace the static tracking areas of current systems with these local neighbourhoods such that, when a UE registers with the network, it receives the current local neighbourhood of its preferred cell as a list of identifiers of other cells. The most straightforward way to implement this in current systems is to use the tracking area identifier (TAI) lists which are already supported by the user equipment [4]. As the local neighbourhoods evolve and are updated with new statistics, the network nodes need to keep track of the associations between UEs and neighbourhoods for up to a period as long as the periodic location update (PLU) timeout (normally a few hours). Figure 2 shows how two neighbourhoods local to the two central cells $R$ and $G$ overlap and how location updates are triggered when a UE associated with $R$ leaves its current neighbourhood and becomes associated instead with the cell $G$ and its current neighbourhood. From the UE perspective the update triggering mechanism remains unchanged compared to current procedures, but in an LTE implementation the TAI list will be used to represent our overlapping, local and dynamically updated neighbourhood.

Figure 3(a) illustrates how the trace is built up during a sequence of location updates, connections and handovers and how information about successful localisations is propagated back to cells previously associated with the UE.

At an incoming connection, when the UE needs to be localised, the network node responsible for the cell where the UE last resided retrieves the neighbourhood associated with the UE and requests the cells in that neighbourhood to broadcast paging messages just as in current systems. However, since the estimates of our model also tells us in which cells we are most likely to find the UE, we can choose to try the most likely cells first.

For this we divide the neighbourhood into a number of phases, the first of which requests paging in only the most likely cells, and so on in probability order. Since each phase takes a significant amount of time, we can only afford a small number of phases (two to four), which means that how we partition the cells of the neighbourhood into phases becomes important for the efficiency of the paging scheme. Different partitions give rise to a typical expected number of pages and response times for the average UE. However, given a preferred balance between paging cost and response time, how to partition the cells of the neighbourhood

![Location update between local neighborhoods](image-url)

Figure 2. Overlapping neighbourhoods local to each cell are formed using mobility estimates. Location updates are triggered when the UE leaves the neighbourhood of its resident cell, just as in the current system. No UE modifications are necessary.
efficiently becomes an optimisation problem, for which we propose a constraint programming solution. As a refinement, and in order to capture the dependency of UE location at the time since it last resided at a cell $i$, we maintain separate sets of counters $n_{ij}^{l}$ for a small number of distinct time frames $l$. Figure 3(b) illustrates
how the local neighbourhood of a central cell is paged in three phases for a case similar to what we would expect around a major road.

In summary, to solve the mobility management problem we propose to:

1. construct and maintain, for each cell, a dynamic local neighbourhood, i.e. a list of cells where recently registered UEs are most likely to be found;
2. track UEs to the level of recorded neighbourhoods and page efficiently within them using estimates of local mobility based on mobility statistics collected online.

To achieve this we:

- record for each connection, location update and handover of an UE
- the cell where it occurred;
- where the UE was previously registered;

this constitutes a sparse distributed trace of the movement of each UE;

1. record the cells of successful localisations and distribute counts/cell along the trace for a limited duration, after which the individual trace is purged;
2. estimate for each cell the probability of an arbitrary UE being localised at each observed, neighbouring cell;
3. do this for several disjoint time frames, to approximate the dependency of the distribution at the time since the UE last resided at some cell of the network;
4. as the resulting estimates are themselves time dependent, we maintain several parallel and overlapping estimators to obtain a dynamically updated and self-adapting Bayesian estimate for each time frame.

The result is an efficient, robust and dynamically self-adaptive approach to mobility management that solves one of the the most important configuration problems in cellular network management. Only two high-level configuration parameters remain. The first is the goal probability mass represented by each neighbourhood which directly influences average update rates, and the second a weight which expresses the relative cost of page messages against increases in response time. We expect both of these to be global parameters for a given network.

We present in the following sections a detailed, completely general and fully distributed version of our proposal. We assume for the purpose of the presentation that we have a single management entity for each cell. From the point of the proposal the number of cells managed by any given MME is immaterial, but assuming that MME and cell can be mapped 1:1 has advantages for generality and clarity of presentation. In a real implementation, a more efficient algorithm running on MMEs managing many local cells can be used, and if records of the mobility of UEs between MMEs is desired the message-passing mechanism between cells presented below can be implemented between the involved MMEs. One could also choose to implement the scheme only within each MME, in which case it becomes a centralised solution for managing standard configuration parameters in LTE, which allows mixing nodes implementing the scheme with older ones that do not.

Due to the limit (of 16) on the size of the TAI list in current versions of LTE [4], we propose that for a practical implementation one should use small groups of (for example) adjacent cells as static TAs, and use TAI lists based on these to represent neighbourhoods. Implemented in this way, our proposal becomes a way to dynamically configure the TAI lists of LTE. This will yield neighbourhoods consisting of up to a few hundred cells but which may include some cells with low likelihood if they belong to highly likely TAs. In future standards, this can be avoided by using longer lists of single cell i.d.s as neighbourhoods. However, in the following presentation we will, for generality, ignore this complication and assume that TAs correspond to single cells and that neighbourhoods are directly implemented as TAI lists with no hard limit on size.

2.2. Distributed trace and neighbourhood records

In order to maintain the estimates used to compute and update the neighbourhoods, we construct a short distributed trace per UE of the cells with which the UE has been associated through location updates,
connections and handovers. At each such event, the time and previous cell of the UE is recorded. The current neighbourhood of the new cell is also associated with and transferred to the UE as a TAI list.

This ‘chain’ of previous association records constitutes a distributed, short-term location trace of the UE which, after a time related to the periodic location update timeout, can be discarded. From a privacy (and management overhead) perspective, it is noted that neither long-term individual associations nor any profiling of individual UE behaviour are required in our approach.

Algorithm 1 implements three routines in the cell: ‘initialise’ sets up the cell parameters and statistics counters. The parameters of the cell initialisation is summarised in Table 1. ‘regRequest’ is used by a UE to register at the cell, which then updates its current local neighbourhood LN, stores the association of the UE with LN and, if necessary, sets up paging sequences for any previously unknown LN. If the registration is part of a location update, the previous cell is notified by an ‘unReg’ message and a record of the transfer is set up before acknowledging the UE of the registration with a copy of LN in the form a TAI list.

Algorithm 1 Cell initialisation, registration and de-registration routines

\begin{verbatim}
Algorithm 1 Cell initialisation, registration and de-registration routines

on event < initialise i> do
  PagePhases ← 3; PLUtimeOut ← 120 minutes;
  MaxL ← 5; t ← [0, 2, 5, 15, 45, PLUtimeOut];
  (# paging phases, PLU timeout and time frame parameters in minutes)
  N ← 200; M ← 2; {estimator size & number}
  for (l = 0; l ≤ MaxL; l++) do
    ηl ← 0; n_{l(l+1)} ← 1; n_{ii(l+1)} ← 1; {index and zero knowledge prior}
    for (k = 0; k < M − 1; k++) do
      n_{l(l+k) mod M} ← 0; n_{ii(l+k) mod M} ← 0;
    end for
  end for
end {initialize counters}
end for {each time frame l}
end {initialize cell i}

on event < regRequest Ue PrevTime PrevCell> do
  LN ← (set of most likely destination cells) ; {see Section 2.5}
  RegTime(Ue) ← now;
  if PrevCell = "none" then
    PrevReg(Ue, PrevTime) ← PrevCell; {set up user trace}
    trigger < unReg Ue RegTime(Ue) Self > @ PrevCell
  end if
  if \[NUNeighbourhoods(U) = LN\] then
    for (l = 1; l ≤ MaxL; l++) do
      trigger < pageSequenceUpdate l LN > @ Self {see Algorithm 3}
    end for {each time frame l}
  end if
  Neighbourhoods(Ue) ← LN; {store association}
  trigger < regAck Self RegTime(Ue)
    PLUtimeOut TAIlist(LN) > @ Ue
end {registration of user U}

on event < unReg Ue NextTime NextCell > do
  forget RegTime(Ue); forget Neighbourhoods(Ue);
  if \[NUNeighbourhoods(U) = LN\] then
    for (l = 1; l ≤ MaxL; l++) do
      forget PageSequence(l LN);
    end for {all time frames}
  end if {last user for LN}
end
\end{verbatim}
Algorithm 2 shows the corresponding event handlers in the UE: ‘ initialise ’ initiates the registration at a given cell \( C_l \). Upon receiving a ‘ regAck ’ from the MME of \( \text{PrevCl} \), it records the connection data, and the received TAI list \( T_l \) as its own. On timeout, or when the UE moves closer to a new cell, ‘ locationUpdate ’ is used to initiate a location update on the same criteria as in current implementations.

**Algorithm 2 UE initialisation, registration and location update routines**

```plaintext
on event < initialise \( \text{Cell} \) do
  trigger < regRequest Self 0 “none” > @ Cell
end

on event < regAck Cell RegTime PLUtimeOut Tl > do
  RegCell \leftarrow \text{Cell}; \text{CellRegTime} \leftarrow \text{RegTime};
  ReRegTime \leftarrow \text{now} + \text{PLUtimeOut};
  TAIlist \leftarrow \{\text{store registration data}\}
end

on event < locationUpdate Cell > do
  if \( (\text{Cell} \notin \text{TAIlist}) \) —— \{neighbourhood exit or time trigger\}
    \( (\text{now} > \text{ReRegTime}) \) then
      trigger < regRequest Self CellRegTime RegCell > @ Cell
  end if \{initiate network location update\}
end \{UE cell level update\}
```

2.3. Observation propagation

UE mobility data are collected and distributed through the relevant parts of the network using a distributed algorithm as follows. At each successful localisation of a UE \( U \) at cell \( C_j \), each cell \( C_i \) in the distributed trace for \( U \) starting at \( C_j \) is recursively informed of the observation. The MME of any cell \( C_i \) in the trace, i.e. where \( U \) has previously resided at time \( t_i \), on receiving the observation \( (t_j, C_j) \) updates, if \( t_j - t_i \leq \text{PLUtime} \), counters \( n_i^l \) and \( n_i^{l,j} \) for \( (t_j - t_i) \in l \), where \( l \) belongs to a partitioning of \( (0, \text{PLUtime}) \] into distinct time frames (duration intervals). That is, for each cell \( i \) we maintain, for each time frame \( l \), one counter \( n_i^l \) and one \( n_i^{l,j} \) for each (known) cell \( j \), including \( i \) itself. With a periodic location update timeout of 2 hours we may, for example, use \( l \in \{(0,2],[2,5],[5,15],[15,45],[45,120]\} \) minutes.

Periodic location update timeouts; the number and duration of time frames can be local to each cell and, if desired, to broad categories of UEs. If periodic location update timeouts differ between cells, some care needs to be taken to account for this when purging cell/UE association records. Otherwise the only overhead associated with this scheme is memory and fairly minimal and easily automated management.

In more detail, note how in Algorithm 1 a distributed ‘ trace ’ of the movement of each UE is accumulated in the vectors \( \text{PrvReg} \) at each re-registration. This trace is used to distribute information of observed mobility patterns between the cells involved at connections, location updates and during handovers. This is achieved through recursively triggering an ‘ observe ’ action at the registered cell on each successful localisation. The mechanism involved is formalised in Algorithm 3, which also gives a high-level indication of the paging sequence update mechanism, described in more detail in Section 2.6. Refer to Figure 3(a) for an overview of the mechanism. The trace can also be used, in combination with a timeout mechanism to safely clear UE data from the MME after its use in constructing the collective mobility patterns is served, although the exact mechanism could be implemented in several ways and is not shown here.

Algorithm 3 is the core of the proposed mechanism since here, first of all, the estimates are evolved, and the statistics counters \( n_i^{l,j} \) and \( n_i^l \) for the relevant time frame \( l \) and current estimator index \( \eta_l \) are updated as indicated in Section 2.4.
The complexity of the estimate update in a single cell is $O(1)$, since the counter update operation is $O(M)$ for small constant $M$, and the estimate evolution which is linear in the number of previously observed cells is performed only once every $N$ observations, for $N$ typically larger than the number of observed cells. The recursion on the $\text{PrevReg}$ links does create a small cascade of updates in the cells recently visited by the UE but in practice these are seldom longer than a few tens.

Algorithm 3 Cell mobility statistics gathering routines

```
on event <observe Ue Cell ObsTime RegTime> do
  if $\text{ObsTime} - \text{RegTime(Ue)} \leq \text{PLUtimeOut}$ then
    if $n_i^{0\text{no}} \geq N(\text{MaxL} - 1)$ then
      trigger <evolveEstimate $0$ @ Self
    end if {observation count reached for LN estimate}
  end if
  for $l$ s.t. $l[l - 1] \leq \text{ObsTime} - \text{RegTime(Ue)} < l[l]$ do
    if $n_i^{ln} \geq N$ then
      trigger <evolveEstimate $l$ @ Self
    end if {observation count reached for $l$}
    for $(k = 0; k < M - 1; k++)$ do
      $n_i^{(n_i - k \mod M)}++$; $n_i^{(n_i - k \mod M)}++$ {update all estimators except prior}
    end for {where $i = \text{Self} \land j = \text{Cell}$}
    for all $\text{Ngh}$ s.t. $\exists(U)\text{Neighbourhoods}(U) = \text{Ngh}$ do
      if $p^{'(\text{Ngh})} = \{p^{ln}_{ij} : j \in \text{Ngh}, i = \text{Self}\}$ differs sufficiently from the support for corresponding stored paging sequence then
        trigger <pageSequenceUpdate $l$ Ngh @ Self
      end if {update paging sequence}
    end for {all stored neighbourhoods}
  end for {in the locally relevant time frame}
  end if {within PLU timeout limit}
  if $\exists(\text{PrevTime} < \text{RegTime})$ s.t. $\text{PrevReg(Ue, PrevTime)}$ is recorded then
    for the latest such $\text{PrevTime}$ do
      trigger <observe Ue Cell ObsTime PrevTime> @
      $\text{PrevReg(Ue, PrevTime)}$
    end for {use only latest to avoid loops}
  end if {propagate observation}
end

on event evolveEstimate $l$ do
  for all cells $j$ recorded at $\text{Self}$ do
    $n_i^{ln} \leftarrow p^{ln}_{ij}$; $n_i^{ln} \leftarrow 1$; {store current estimate as new prior}
    $n_i^{(n_i+1 \mod M)} \leftarrow 0$; $n_i^{(n_i+1 \mod M)} \leftarrow 0$; {clear old prior}
  end for
  $\eta_i \leftarrow \eta_i + 1 \mod M$; {increment estimator index $\eta$}
end

on event <pageSequenceUpdate $l$ N> do
  <calculate a new paging sequence $\text{PageSequence}^l(N)$ w. support $p^l(N) = \{p^{ln}_{ij} : j \in N, i = \text{Self}\}$>
  <replace any sequence stored for $N$
    and time frame $l$, with $\text{PageSequence}^l(N)$>
  <store $p^l(N)$ as the support for $\text{PageSequence}^l(N)$>
end {See Section 2.4 for definition of $p^{ln}_{ij}$}
```

The complexity of the estimate update in a single cell is $O(1)$, since the counter update operation is $O(M)$ for small constant $M$, and the estimate evolution which is linear in the number of previously observed cells is performed only once every $N$ observations, for $N$ typically larger than the number of observed cells. The recursion on the $\text{PrevReg}$ links does create a small cascade of updates in the cells recently visited by the UE but in practice these are seldom longer than a few tens.
What does affect scalability is the update of the paging sequences for each current neighbourhood in the relevant time frame which scales with both neighbourhood size and number of phases. The solution space for this problem, for large parameters, is huge:

$$O\left(\frac{n - 1}{k}\right)$$

for environment size $n$ and number of phases $k$, but the problem is in practice tractable for small $n$ and $k$, and with the use of approximations—even moderately large ones. In Section 2.6 we present one particular solution to this problem that gives optimal solutions within a few seconds for $k = 3$ and $n \lesssim 90$ (see Figure 5(b) for full scaling results). On the other hand, the mechanism used to maintain the estimates and neighbourhoods is orthogonal to how well we solve this problem. Almost any partition will result in fewer pages than most current practices and, if very large neighbourhoods are desired, simple approximations or even fixed sized partitions can be used, at the cost of higher expected page counts.

We filter observation events from the paging sequence update mechanism by storing the support with each computed sequence and requiring that the current estimate has changed sufficiently before re-triggering the sequence generation. The criterion for a sufficiently large change is checked by measuring the Kullback-Leibler divergence [22] between the current estimate, and that used as support for the stored paging sequence. The K-L test has complexity roughly linear in the size of the neighbourhood.

Depending on the duration between the observation and the registration time at the cell, the previous cell in the user trace is also notified of the observation and so on until the duration between the local record for the UE at the current cell and the time of observation becomes so long that it is no longer relevant to the cell, normally after the time at which the periodic location timeout for the UE would have triggered. At this point the observation propagation is terminated. There are some intricacies involved here, i.e. when periodic location update times differs between cells, and to take care of situations where the UE has been registered several times at a single cell.

### 2.4. Robust estimation and representation of UE mobility patterns

To create local neighbourhoods and optimal paging sequences (see Section 2.6 below), we estimate the conditional UE location distribution, given where and when the UE was registered and other locally available data. The mechanisms for user mobility management and data collection presented above ensure that each cell is informed about where a previously registered UE is located at the time of a successful localisation, meaning that both first- and higher-order Markov models of user mobility can be estimated (estimating the location distribution conditional on where the UE is registered and conditional on where the UE is and has been registered, respectively).

Here, we choose a base model using a first-order Markov approximation. To account for the fact that the distribution over cells typically is also highly dependent on the time since UE registration, we extend the conditional to account for this. The entropy of this distribution

---

**Table 1. Cell initialisation parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PagePhases</td>
<td>Number of page phases</td>
</tr>
<tr>
<td>PLUtimeOut</td>
<td>Interval between periodic location updates (in minutes)</td>
</tr>
<tr>
<td>MaxL</td>
<td>Number of discrete time frames</td>
</tr>
<tr>
<td>$t$</td>
<td>Time frame intervals, i.e. 0...2, 2...5...minutes after a connection</td>
</tr>
<tr>
<td>$l$</td>
<td>Time frame index, from 1...MaxL</td>
</tr>
<tr>
<td>$N$</td>
<td>Estimator window size (see Section 2.4)</td>
</tr>
<tr>
<td>$M$</td>
<td>Number of parallel estimators employed</td>
</tr>
<tr>
<td>$\eta_i^0, \eta_i^1$</td>
<td>Estimator indices, initialised to 0</td>
</tr>
<tr>
<td>$n_i^l, n_i^0$</td>
<td>Counters for observations at cell $i$ for time frame $l$</td>
</tr>
<tr>
<td>$(\eta_i+1)mod M$</td>
<td>Refers to the prior of an estimator $p_{ij}$</td>
</tr>
</tbody>
</table>

Copyright © 2013 John Wiley & Sons, Ltd.


DOI: 10.1002/nem
typically increases with time, as it is more uncertain where the UE is located when it has had a longer time to move through the network. To simplify the representation and estimation of the conditional we introduce a number of fixed time frames indexed by a variable \( l \). These intervals are fixed for each cell, and can typically be set to 0–2, 2–5, 5–15 and 45–120 minutes, for example, as in Algorithm 1 above. In practice, this means that we estimate and store one conditional for each time frame at each cell.

Additionally, we need to account for long-term development of usage and mobility patterns. We manage this by using multiple overlapping estimators \( E_\eta \) for the UE location distribution, and the latest complete model as prior to the next model as illustrated in Figure 4. The estimation scheme is circular using \( M = N/T \) models, each based on \( N \) observations and degree of overlap \( T \). The degree of overlap directly affects the temporal properties, i.e. how fast older historical data are discounted. By using the previous model as a prior for the following model, a smooth transition between models is achieved, while older mobility patterns have a smaller impact on the current parameter estimates, offering adaptation to new network mobility regimes. Note that only \( M \) sets of counters need to be kept in memory at any one time, which means that in a practical implementation we circulate between \( M \) model representations, as indicated in Figure 4 and in Algorithm 3.

Using Bayesian inference where all probabilities are estimated using means over the posterior, we can write the \( \eta \)th estimate of the probability that a UE is located at cell \( j \) given that it was last registered at cell \( i \) in time interval \( l \) and for a current estimator index \( \eta \) as

\[
P_{ij}^{\eta_l} = \frac{p_{ij}^{(\eta-M)}}{\alpha + n_{ij}^{\eta_l}}
\]

where \( p_{ij}^{(\eta-M)} \) represents our last complete estimate and \( n_{ij}^{\eta_l} \) the number of successful localisations of a UE to cell \( j \) during the time interval \( l \) since this UE resided at cell \( i \). \( \alpha \) controls the equivalent sample size of the prior, i.e. how much we trust the prior compared to the new observations. Using a prior based on zero knowledge of the network topology, our initial estimate only includes the current cell and can be written as

\[
(\forall \eta < M) p_{ij}^{\eta_l} = \begin{cases} 
1 & \text{if } i = j \\
0 & \text{if } i \neq j
\end{cases}
\]

Again, note that although in the expressions above model \( \eta \) increases infinitely, only \( M \) models and sets of counters need to be kept in memory at once. For smoother transitions between estimates, a larger value of \( M \) can be used, but for most practical purposes \( M = 2 \) should give adequate performance. For fast adaptation, \( N \) can be chosen to be of the order of about 200 samples, but can be set to a larger value if stability of the estimates is prioritised. To reduce the sensitivity to temporary fluctuations in mobility patterns, we typically set \( \alpha = N \). Note also that in the algorithms \( p_{ij}^{(\eta-M)} \) is stored as \( n_{ij}^{(\eta-M)} \).

![Figure 4. Overlapping estimators. Each \( E_i \) contains counters for each neighbouring cell, and \( E_1 \) is used as prior for a Bayesian estimate based on the counters in \( E_M \). Intermediate estimators \( E_2 - E_{M-1} \) are updated in parallel with \( E_M \) for a smooth transition between estimates, performed at each \( T \) observations. Each estimator contains the statistics for \( N = TM \) observations](image)
2.5. Local neighbourhoods

To assign neighbourhoods, we use an estimate of the mobility patterns within all possible time intervals:

\[ p_{ij}^\eta = \frac{p_{ij}^{\eta-M} + n_{ij}^\eta}{\alpha + n_{i}^\eta} \]  

(3)

which is similar to the expression above for one time frame, using overlapping models in the same manner. Here, as above \( p_{ij}^{\eta-M} \) represents our previous estimate and \( n_{ij}^\eta \) the number of successful localisations of a UE to cell \( j \) during any time interval since this UE resided at cell \( i \), i.e. \( n_{ij}^\eta = \sum_k n_{ij}^{k\eta} \), and similarly for \( n_i^\eta \). Note that in the algorithms \( n_{ij}^\eta \) is explicitly represented and updated as \( n_{ij}^{\eta_0} \).

We can use two parameters to control and limit the size of neighbourhoods created from this distribution estimate: a maximum size \( K \), which cannot be exceeded, or a cut-off \( c \) on the joint probability mass represented by cells included in the neighbourhood. The TAI list is then created by

1. sorting the one-dimensional conditional probability vector over all cells, where UEs once residing at the current cell have later been observed;
2. starting with the largest probability, adding all corresponding cells/TAs to the TAI list until we reach size \( K \) or \( c \) is exceeded.

This assignment is performed each time a UE registers with a cell.

The initial prior is, as for each time frame above, set to

\[ (\forall \eta < M) p_{ij}^\eta = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \]  

(4)

which leads to the expected behaviour for cell neighbourhood construction. The neighbourhoods will initially only contain the current cell but gradually grow as more statistics are collected. However, if prior knowledge on network topology and mobility patterns is available, this could be encoded in the prior for faster convergence.

A possible extended approach to TAI list assignment is to create TAI lists as described above for each time frame. If the network and handset implementation would allow for multiple TAI lists with corresponding time frames, these could be sent directly to the handset, which would switch between TAI lists according to the duration since its last TA update, allowing for greater precision within UE location management.

2.6. Optimal paging phase partitioning

At an incoming connection, we would like to use the most suitable partitioning of the neighbourhood associated with the requested UE into a set of consecutive paging phases (a ‘paging sequence’). ‘Suitable’ here should be understood in terms of the current estimate of UE mobility and a given relative weight between the cost of the expected number of page messages per localisation and the disadvantage represented by increased response times on quality of service. In order for the paging sequences to be ready at connection time, we propose to construct them as soon as we come across an unrecorded neighbourhood, and recalculate them as required when our estimates evolves.

At connection time, the precomputed paging sequence is thus used to page all the cells of the first phase in parallel, proceeding with the next phase in the case of no response, and so on. That is, upon receiving a localisation request for a UE \( U \) at time \( t \) which last resided at cell \( C_i \) at \( t_i \), the MME of \( C_i \) retrieves the paging sequence \( S_{nu}^l \) for the neighbourhood \( n_{U} \) associated with \( U \) and \( l \) s.t. \( t - t_i \in l \), then sends page messages in each cell \( C_j \in n_{U} \) in a sequence of phases \( P_{\in S_{nu}^l} \).
Noting that paging all cells in a single phase would minimise response time (since success in an early phase eliminates the need for further paging) but would increase the number of page messages broadcast (since all cells in the neighbourhood are affected). We thus prefer, for any given number $H$ of phases, to page in the most likely cells first in order to reduce the expected total number of page messages required for localisation.

For each cell in any given neighbourhood and estimated UE location distribution, there is a trade-off between paging it in an early phase, thus contributing to the expected page message count for the connection, and postponing it to a later phase, thereby increasing response time, but reducing the probability of having to send a page message in that cell at all. These two objectives are encoded in our set partitioning formulation as delay cost and page cost, respectively. This paging phase generation mechanism produces a partitioning of the cells in the neighbourhood that is optimal w.r.t. a given weighted sum of these two cost components.

In more detail, we note that the probability of success in a phase $h \in \{1, \ldots, H\}$ is a function of the probability mass of all cells paged in $h$ and the probability of reaching phase $h$ is the probability mass of all cells $h, \ldots, H$ not yet paged. Since we can order the cells of the neighbourhood according to their individual probability, the mass of each phase depends directly on the number of cells in each phase.

We currently use a constraint programming formulation to solve this optimisation problem, and the constraints and objectives as they occur in the formulation below are transformed in our model into a formulation in which the sum of objectives for each phase depend on the probability mass, or equivalently on the number of cells in that phase. The phase size is captured in our implementation by a global cardinality constraint [23] that iteratively and efficiently prunes the solution space as we search for increasingly better solutions and eventually reach an optimal one.

Associating costs for page messages and response time increases, optimal partitioning of cells into phases can be formulated as follows. Express the cost of a partition of the cells in an neighbourhood $n_t$ and time frame $l$ in terms of boolean variables $q_{jh}$ representing the fact that the cell $C_j$ is paged in phase $h$. Note, first of all, that since each cell $C_j \in Lnl_U$ is paged exactly once:

$$\sum_{0 \leq h \leq H} q_{jh} = 1 \quad (5)$$

for each $j$ and $l$. We then express the cost of delaying the pages of cells to phase $h$ as

$$\sum_{j: C_j \in L_h} w_h q_{jh} p_{ij}^{lh} \quad (6)$$

where $p_{ij}^{lh}$ is the estimated probability that $U$ is in $C_j$ at time $t$ and $w_h$ is a weight specific to phase $h$. $w_h$ should grow with $h$ to reflect our estimation of the cost increased response times represent in terms of connection failures and general system latency. The delay cost for a complete partition is the sum over all phases.

We also express a discounted page cost for a phase $h$ as the number of page messages $c_h$ sent in that phase minus a discount based on the probability of having successfully located $U$ in an earlier phase:

$$c_h = \sum_{1 \leq g < h} \sum_{j: C_j \in Lnl_U} q_{jh} p_{ij}^{lh} \quad (7)$$

and summed over all phases for the complete partition. The cost function for the set partition problem is the sum of these two components (equations (6) and (7)) under the constraint equation (5) for all $j$ and Booleans $q_{jh}$. This cost can be expressed as a fixed cost for the localisation of an arbitrary UE minus a sum depending only on probabilities of each cell in the estimate and the number of cells in each phase.

Note that this optimisation problem takes into account only the expected number of page messages required to locate a user $U$ and the response time increase. The balance between these
opposing objectives is determined by the choice of weights $w_k$ but does not take into account any cost for location updates. The number of location updates, instead, directly depends on the probability mass represented by the cells in the neighbourhood, which is orthogonal to the optimisation problem.

For a small number of phases and medium-sized neighbourhoods the solver provides proven optimal solutions in reasonable time (see Figure 5b). We note that $H$ should in any case be kept small, of the order of 2–4, say, since each phase takes a significant amount of time and contributes to the total response time. For larger neighbourhoods, approximations of the optimal paging sequence can be used. In the paging stage we are not restricted to paging cells in groups corresponding to any static TAs used to allow larger neighbourhoods in current LTE standards, but can base the sequence on individual estimates for each cell.

Re-computation of paging sequences is still fairly costly computationally, and for this reason we filter triggering events (observations of successful localisation) by requiring a sufficiently large (Kullback–Leibler) divergence of the current estimate as compared to the estimate supporting the current paging sequence for each neighbourhood and time frame. Accuracy and timeliness of the estimate can thus be balanced against computational complexity.

### 2.7. Connection set-up and paging

#### Algorithm 4 Cell connection set-up and paging routines

```plaintext
on event <connect Ue> do
    Location(Ue) ← “unknown”;
    for (Phase ← 0; Phase < PagePhases; Phase++) do
        if Location(Ue) = “unknown” then
            for the least $l$ s.t. $t[l] > \$\text{now} − \text{RegTime}(Ue)$ do
                for all Cell in PageSequence($l, LN(Ue)) [Phase]$ do
                    trigger <pageRequest Ue Self > @ Cell
                end for {all cells in phase}
            end for {the relevant time frame $l$}
        else
            trigger <locationUpdate Ue Location(Ue) > @ Self
            <tell Network to set up connection through Location(Ue)>
            trigger <observe Self Cell > @ Location(Ue) {see Algorithm 3}
        end if {location still unknown}
        sleep PhaseTimeOut;
    end for {each phase} {report failure if location still unknown}
end {localisation process}
```

```plaintext
on event <pageResponse Ue Cell> do
    Location(Ue) ← Cell;
end {response received from UE’s current cell}
```

```plaintext
on event <pageRequest Ue RqCell> do
    RqCell(Ue) ← RqCell;
    repeat
        signal page request in cell of Self;
        until PqTimeOut
        RqCell(Ue) ← “none”;  
    end {broadcast page message in ”own” cell}
end {UE response received}
```

2.7. Connection set-up and paging
We can now formalise the localisation, connection set-up and paging mechanisms in Algorithm 4. Whenever an incoming connection is initiated for a UE $U_e$, a precomputed sequence ‘$PageSequence(l, LN(U_e))$’ of phases associated with the UE’s current neighbourhood is applied. Stepping through each phase, all its cells are sent ‘pageRequest’ messages. Upon receiving a ‘pageRequest’, each cell in that phase broadcasts page messages over the radio network and awaits a response from the UE. If the UE is indeed in one of the cells (say $C_i$) of that phase, it acknowledges the page with a ‘pageAck’ message back to $C_i$, and $C_i$ can reply with a ‘pageResponse’ message to $C$. If no ‘pageResponse’ is received during a phase, $C$ continues, after a timeout, with the next phase until a ‘pageResponse’ message is received from $C_h$, at which it sets up the connection (through $C_h$) and initiates a chain of ‘observe’ messages at $C_h$.

The UE side of the paging process remains unchanged w.r.t. current implementations.

3. EXPERIMENTAL RESULTS

To verify the correctness and scalability of the proposed method, we have undertaken a series of experiments using a detailed prototype implementation of the method and two different mobility and call scenarios.
3.1. The zCap prototype

The zCap prototype as well as the network traffic and the two mobility simulators reported on are all implemented in SICStus Prolog [24]. The network part of the implementation is very close to the algorithms presented in Section 2, with the estimation, neighbourhood and paging sequence creation and updates all implemented as event handlers and message passing between the nodes, i.e. as a distributed system.

The neighbourhoods are implemented directly as TAI lists, with a cut-off on either size or on probability mass. The UE part emulates LTE UEs w.r.t. paging and mobility, but without any hard restriction on TAI list size.

Paging sequences are constructed using a constraint program with primarily global cardinality [23] for partition size counting and indexing constraints for the assignment problem as implemented in SICStus. Solutions are produced using branch and bound search with an option to use a limit on execution time for each sequence computation, providing the best solution found within that limit.

Everything except the optimisation mechanism and the simulator used for the experiments can easily be ported to any modern programming environment. Porting it to run on MMEs serving many nodes would allow many code optimisations but also take more serious effort. Restricting the system to run within a single MME would simplify it.

3.2. Single cell simulation with constant UE destination distribution

The first scenario is based on UEs registering at a single cell A and then being observed exactly once at any cell B out of a given number of surrounding nodes according to a given stationary distribution. The point of this simplified mobility simulation is to study how the proposed mechanism, based on samples from the stationary distribution, builds up mobility estimates and neighbourhoods up to a first periodic location update, and using zero prior information of cell proximity and UE mobility.

An example of the type of distributions used is illustrated in Figure 5(a), where A is in the centre and B can be anywhere. The stationary distribution is designed to capture the dependence on distance from the first node A with two directions having a slightly higher probability than others, as expected, e.g. around a major road. For the sake of simplicity we assume here that all of this happens within one single time frame l and that the neighbourhood has a maximum allowed size of 64 cells.

Figure 6(a) illustrates the convergence of the estimated distribution built up inside the central node where each UE in the simulation is registered, as compared to the stationary distribution used to generate the ‘mobility’ of the experiment. The measure used is Kullback–Leibler divergence. The proposed estimation method converges nicely and for this sample set size (91 cells) errors become insignificant after around 300 observations (successful localisations).

As observations accumulate, the node gradually updates its neighbourhoods to include the most probable cells that its UEs are likely to be located in. This means that, as information about the distribution is built up, the neighbourhood will first grow, in this case to a fixed maximum size, and then fluctuate slightly as the estimate of the stationary distribution converges. Figure 6(b) illustrates how the probability of encountering a new neighbourhood decreases from 1 to about .2 over a run of 999 observations. A residual probability cut-off of 0.2 would have resulted in a similarly sized neighbourhood for this stationary distribution.

As mentioned above, paging sequence computations are computationally costly and hence new sequences for already stored neighbourhoods are recomputed only when the Kullback–Leibler divergence between current estimate and the one used to compute a previously stored sequence exceeds a certain threshold. Figure 6(b) plots the probability of such updates for a suitable cut-off divergence, while Figure 5(b) shows the runtime in seconds for optimising typical paging sequences of a single neighbourhood for some neighbourhood sizes and different numbers of phases (on a consumer-grade mobile CPU). It is seen that if we want the optimal paging sequence computation to terminate within a few seconds on this type of hardware, the
neighbourhood size should be kept below around 60 cells for four phases, and below a few hundred cells for three phases, unless we are content with approximations. Since timescale is logarithmic, the complete mechanism (as implemented) scales exponentially, which is to be expected given the complexity of the general problem. The size of problem we can solve however, is practically useful.

The kind of gains that can be expected by implementing the proposed method can be illustrated in several ways. In Figure 7(a) we show the number of cells included in each paging phase and in Figure 7(b) the sum of (estimated) probabilities of the cells included in each phase. To read the graphs, note that, for example, at 750 observations phase one (bottom) includes nine cells (Figure 7a) and the probability of finding the user in these cells is 43% (Figure 7b), phase two (middle) contains 24 nodes (Figure 7b) and the probability of finding the user in this cell is 23%, etc. Similarly, it can be seen that the 64 cells included in the neighbourhood of the centre cell (Figure 7a) in total cover about 84% of the users (bottom), corresponding to a residual probability of 16% for users who reside at cells outside the neighbourhood of the centre cell and hence would have made one or more location updates to obtain new TAI lists.

Another illustration of possible gain is to plot the expected number of page messages in order to locate a randomly selected UE from the sample set. In Figure 7(c) we see that, for this example, the expected number of page messages for a random incoming connection stabilises at around 26.5 (out of the 64 maintained in the neighbourhood). This should be contrasted to the currently common situation where paging takes place in all 64 cells included in the neighbourhood (TAI list). This means that our method reduces the number of page messages for this example by 58%. For variants (not shown in the figure) with two phases (shorter set-up time) and four phases (longer response time)
we save 48% and 63%, respectively. As we will see in the next section, these measures can be improved even further in a more realistic setting.

3.3. Multi-cell simulation over the Cologne mobility micro-simulation trace

In a second series of experiments we obtained mobility data from the TAPAS project [25]. These data provide a trace of a micro-simulation of the inhabitants in the city of Cologne during morning rush hour. It is based on a very detailed geography and well-researched origin/destination and timing information [26], covers an area of roughly 24 × 31 km² and contains 76 million individual movements as performed by 121,164 agents over 2 hours.

Over this area we impose a grid of up to 3000 uniformly placed square ‘network cells’ with a side of 500 m and preserve only those movement events resulting in the UEs crossing cell borders. In the largest instances this gives us 2.6 million cell crossing events in up to 1.725 active ‘cells’. A network activity simulation was set up for the UEs residing in each studied area based on a Poisson process with a connection rate of 3 connections per hour and user. Of these, 0.5 are assumed to be ‘voice call’ which generate both connect and disconnect handover events.

Figure 7. Paging performance for a case with three phases over 999 samples. Input is single stationary distribution over 91 cells and a maximum neighbourhood size of 64. (a) The number of cells in each phase. (b) Probability of each phase. Note how the smallest phase accounts for the greatest mass. (c) Expected number of page messages for a random connection.
Figure 8. Two-hour 80-cell simulation. (a) Average number of location updates per cell crossing and neighbourhood size. (b) Average number of pages and page phases (−1) needed per localisation.

Figure 9. Eighty-minute 317-cell simulation. (a) Average number of location updates per cell crossing and neighbourhood size. (b) Average number of pages and page phases (−1) needed per localisation.
whenever a cell crossing occurs during a call. The remaining 2.5 are considered ‘data connections’ and are assumed to be momentary, generating a single connection event and no handovers. A periodic location update process with a timeout of 2 hours is used to trigger UE-generated location updates in the absence of other connections. At the start of the simulation, and when arriving from outside the studied area, UEs sample their next connection time from the residual of the connection inter-arrival time distribution and a next PLU time as the start/arrival time plus a time selected uniformly in \(0 \ldots PLU\text{timeOut}\).

All of the events above generate location information used to record UE traces, but only .5 of the ‘calls’ and .2 of the ‘data’ connections are considered ‘incoming’ and are used to simulate localisation requests, and thus observations for the mobility estimates. Several discrete event simulations using these data were performed using subsets of the movement data generated by selecting increasingly larger areas around in the town centre, and simulating UE arrivals and departures from the studied area.

We present here the results from two simulations—one in the very centre of the town—using an 81-cell grid covering some 20 km\(^2\) and consisting of 441.730 cell arrivals and departures and 61.214 periodic location update events performed by 15.779 UEs over 2 hours. The second simulation, over a larger area consisting of 317 cells covering around 80 km\(^2\), consists of 627.125 movements and 105.336 periodic location update events performed by 11.622 UEs over roughly 80 minutes. Each network cell is initialised with a prior containing up to four surrounding cells, with .5 probability of remaining in the original cell. Estimates are built gradually as the state and history of the network evolves. A more realistic prior would probably include a much larger selection of neighbourhood cells, but we chose not do so in this experiment in order to explore this minimal prior knowledge case.

The graphs in Figures 8 and 9 illustrate four key measures over the two simulation periods:

1. update rate, as the fraction of cell crossings resulting in location updates, ignoring arrivals and departures to the area and periodic location update events (left scale);
2. neighbourhood size as an average over localisation events (right scale);
3. delay, or contribution to response time, as an average over localisation events, i.e. the number of phases (after the first one) required to locate the UEs (right scale);
4. hit rate, i.e. the average number of page messages per localisation event.

We see in Figure 8(a) how the update rate falls from .67 to .31 while the average neighbourhood (at paging time) grows from 4.7 to around 28.5. In Figure 8(b) the paging efficiency, meanwhile, improves from 2.38 to 1.42 page messages per localisation, while the response time stays extremely low, fluctuating between 1 and 1.06 phases.

In the second run, we see a similar slow reduction in update rate (Figure 9a), going from .66 to .32, while the neighbourhood size grows from 6.3 to 51.0. Convergence of these two measures is fairly slow, and we used a very conservative cut-off (99.9\%) on neighbourhood probability mass to speed up the growth of neighbourhood size and reduction of updates. We can see that the optimisation struggles a bit with keeping response time down and paging efficiency up. Here (Figure 9b) the page messages per localisation actually grows from around 2.1 to around 3.8, while the response time rises steadily from 1 to 1.08 phases, which still represents exceptional paging efficiency, with an average neighbourhood size up to a sixth of the total area size, and insignificant average delay. Over time, as the estimate improves even further, we can choose either to let the neighbourhood grow even larger to further reduce the update rate, or maintain higher hit rate per page message and lower response time, at the cost of slightly smaller neighbourhoods. In the limit, this choice is directly controlled by the residual cut-off, but in a dynamic setting fast convergence of update rate may have to be taken into account as well.

4. CONCLUSIONS

The motivation for this work is that mobility patterns are not sufficiently accounted for today’s networks due to the difficulty and cost of estimating local mobility and configuring network mobility.
parameters accordingly. We propose a complete method (zCap) to autonomously and dynamically handle mobility management and configuration for LTE and related cellular networks. In the proposed solution, only two main parameters remain:

1. goal probability mass cut-off for cell inclusion in local neighbourhood;
2. relative cost of response times versus number of page messages per localisation.

The first one directly influences update rate, while the second one balances the required number of page messages per localisation against increased response time, both of which could be considered business decisions.

zCap makes the following contributions:

- a distributed method to collect spatial page response statistics;
- Bayesian estimation methods for managing dynamics and timeliness of local mobility models;
- a method to construct dynamic neighbourhoods of cells according to the current state of a local mobility model;
- a method to obtain an optimal sequence of paging phases for any neighbourhood and localisation request.

We show with a set of runs of an advanced prototype that the method is sound and has huge potential for autonomously managing the network resources and services efficiently while eliminating most of the manual configuration effort. We argue that the method is implementable within current standards, although the limit on TAI list length in LTE requires an additional first-level (TA) aggregation. The potential of the method lies in radically reduced rate of paging messages per localisation at a modest cost in response time, and in eliminating the need for manual configuration of the most important mobility parameters.

A few open questions remain. If LTE TAI list length restriction remains, methods of constructing aggregates of cells suitable for single TAs, based, for example, on long-term UE mobility statistics, could pose an interesting sub-problem. Managing both cell and TA levels in zCap would be a straightforward extension of the algorithms, as shown here, but a bad choice of large first-level TAs could still produce suboptimal results. Increasing the LTE TAI list length restriction from 16 to 256, for example, would otherwise seem a minor, and in any case, very useful extension. UEs limited by the current TAI list maximum length could keep using static TAs in the interim.

Our estimates capture precisely the spatial distribution of successful UE localisations in several time frames. Other mobility network events, e.g. outgoing connections, handovers, neighbourhood-level movements and time-triggered location updates, are used to build the trace that allows distribution of observation statistics in the network, but currently not for the mobility estimate. Further study is needed to determine if, and with which type of skewness, use of these statistics could improve, for example, the convergence of the estimates. Outgoing connections would seem the least controversial.

To obtain full self-configuration it is required that the method be supported in all eNBs (nodes) of a network. This in turn requires single-vendor radio networks (a quite common case in practice) or multi-vendor compatibility (i.e. international standardisation and widespread implementation). In a multi-vendor network where the mechanism is only partially implemented, the extent to which TAI lists can be optimised would be limited by the boundaries of MME service areas since all TAs in a TAI list must belong to the same MME. In practice, MMEs, and in particular pooled MMEs, can cover relatively large areas and hence we do not expect this to be a significant problem. Moreover, one can envisage ways to optimise these areas based on data collected in the present proposal, although the details of a practically useful method remain an open problem. Finally, we note that that we have not analysed the extent to which data messages between eNBs would add load to the access backhaul, although we note that this problem appears to be small, in particular given the current trend to transfer most of the processing to clouds.

The zCap method and several variants of its implementation are patented [27] by Ericsson AB.
ACKNOWLEDGEMENTS

The research leading up to the results presented here was funded by Vinnova, Sweden’s Innovation Agency, through the DiSC project and by Vinnova and SSF, the Swedish Foundation for Strategic Research, through the SICS Centre for Networked Systems, and to some extent by the MocNet project, also funded by Vinnova. The contribution of Åke Arvidsson was funded through his employment by Ericsson AB.

REFERENCES


AUTHORS’ BIOGRAPHIES

Per Kreuger is a senior research scientist and has been employed at the Swedish Institute of Computer Science (SICS) since 1986. He has a background in computational logics, combinatorial optimisation, constraint programming, distributed algorithms and statistical modelling. He received his PhD in computer science at Chalmers University of Technology in 1995. His research interests currently include application of probabilistic modelling and optimisation to problems in networking and network management, mobility modelling and distributed data analysis.

Daniel Gillblad is a senior researcher and director for the Industrial Applications and Methods laboratory at SICS. He has a background in statistical machine learning and data analysis, and has extensive experience of applying such methods in industrial systems. He holds an MSc in electrical engineering and received his PhD in computer science in 2008, both from the Royal Institute of Technology in Stockholm. He has been with the Swedish Institute of Computer Science (SICS) since 1999, where he currently leads the network management effort within the SICS Centre for Networked Systems and several development projects for advanced management techniques directly for industrial partners. His research interests are currently focused around probabilistic methods for network management, diagnostics, data mining and mobility modelling.

Åke Arvidsson obtained his MSc and PhD degrees in electrical engineering from Lund University, Sweden, in 1982 and 1990, respectively. He has worked with several consultancy companies and held various academic positions in Sweden and Australia. In 1998 he joined Ericsson as Technical Expert in the area of Traffic Theory and since 2008 he has been with Ericsson Research. His current research interests include content delivery, quality of experience and mobility modelling.