進捗報告資料

安達智哉
to-adachi@ist.osaka-u.ac.jp

2019 年 3 月 13 日

目 次

1　はじめに

MME および SGW、PGW などの EPC ノードは、主なリソースとして CPU とメモリを持っている。CPU は、アタッチやデタッチなどのシグナリング処理を実行するために必要とされるリソースである。一方メモリは、ベアラなどのセッション情報を保持するために必要とされるリソースである。これらのリソースは、モバイルネットワークにおける通信を可能にするために必須であるため、ネットワーク事業者は、どちらのリソースも枯渇することがないように、CPU とメモリをバランスよく割り当てる必要がある。

その一方で、近年は M2M/IoT 端末の急激な増加が注目されている。M2M/IoT 端末は通信特性において従来の端末 (携帯電話やスマートフォンなどのユーザ端末) とは大きく異なり、データの送信に周期性や間欠性を持つという特徴がある。そのため、データの送信ごとに idle 状態と connected 状態を遷移することが予想される。その結果、端末のネットワーク接続やデータ送信に必要なシグナリングに関する通信や処理を行う、制御プレーンの輻輳が悪化すると考えられる。また、M2M/IoT 端末は消費電力を抑えることが必要とされている。このような問題に対し、RRC Connected Inactive と呼ばれる M2M/IoT 端末の新たなのステートを導入することによって、消費電力およびシグナリングの削減を目標とする研究が行われている。RRC Connected Inactive とは、M2M/IoT 端末のコンテキストが端末及びネットワークに保存され、RAN-CN 間の接続がアクティブ状態で維持されている状態である。Connected Inactive 状態においては、一部の情報が保持されているため、connected 状態へ遷移する際に発生するシグナリングは、idle 状態と connected 状態を遷移することによって発生するシグナリングよりも小さくなることが予想される。さらに、シグナリングの削減に伴い、端末の消費電力削減も期待できる。実際、文献 [1] および [2] においては、RRC Connected Inactive を導入することにより、シグナリングオーバヘッドの削減および消費電力の削減が可能であることを示している。

上述の RRC Connected Inactive を M2M/IoT 端末に適応することは、CPU やメモリなどのサーバリソースの効率的な割り当てを難しくすると考えられる。なぜなら、RRC Connected Inactive はその特性から、端末がデータを送信していないタイミングにおいてもその端末情報をネットワークに保持するため、コアネットワークノードのメモリに対してこれまで以上の大きな負荷を発生させるためである。また、M2M/IoT 端末の接続台数の予測が難しいこともリソースの割り当てを難しくする一因である。M2M/IoT 端末は、スマートフォンのようなユーザ端末とは異なり、家電や自動車、電気メーター、センサなど様々な場所、様々な用途で使用される可能性があり、端末の台数およびその分布を予測することは困難であると考えられる。さらには、それらの端末の送信タイミングも把握することも容易ではない。このように、新たな状態の導入や、接続台数の予測が難しい IoT 端末の普及により、今後のネットワーク事業者は、コアネットワークノードへのサーバリソースの割り当てがより難しいくなると予想される。

上述のようなネットワーク (サーバリソース消費の予測が難しく、変動が激しいネットワーク) において、収容可能な端末の増加を目的とした既存研究には、スケールアウトの考え方を用いたものが多い。これらの研究では主に稼働するサーバやインスタンスの数をリソースの需要に応じて変動させることにより、ネットワークの変動に対応している。しかし、この方法では、本来必要とされているリソース量 (需用量) よりも多くのリソースが供給される、オーバープロビジョニングが発生する問題がある。なぜなら、これらの研究では、サーバやインスタンス一台あたりのリソース量は一定であることを前提にした研究が多く、細かい粒度でリソースを制御できないためである。また、必要とされる CPU とメモリのリソース比があらかじめ分かっていることを前提とした研究が多く、必要とされるリソース比が未知の場合はリソースの効率的な利用ができないからである。

例えば、CPU のリソース不足を解消するためにケールアウトを行った場合、CPU リソースと同時にメモリリソースも増加する。しかし、メモリは元々ボトルネックにはなっていないため、新たに追加されたメモリはオーバープロビジョニングされたことになる。

このような背景から、CPU とメモリのリソース消費の予測が難しような状況や変動が大きいような状況においても、どちらかがボトルネックにならずに、効率的にリソースを活用するアーキテクチャを考えることは重要である。実際、CPU とメモリのリソースを効率よく活用する研究は、データセンタなどの分野では行われている。文献 [3] では、server disaggregation の考えをデータセンタに適用し、CPU やメモリなどのリソースをモジュール化し、需要に合わせて自由に組み替えることを可能にすることにより、リソースの効率的な利用が可能であることを示している。しかし、需要に合わせてリソース構成を変化させる方式は、短いタイムスケールでの制御は難しいという問題がある。特に、モバイルネットワークおいては、突発的なトラヒックが発生することもあるため、数分以下のオーダでリソース量の制御を行う必要がある。

そこで、本研究ではモバイルネットワークに特化した、CPU とメモリのリソースのオフロードを可能にする仕組みを考案する。具体的には、ネットワークの負荷に合わせて、UE の状態を制御することにより、メモリおよび CPU に与える負荷のバランスを変化させる。UE の状態の制御は、UE が最後にデータを送信したあと、Connected Inactive 状態から Idle 状態に遷移するまでの時間を設定することで実現する。この方法により、CPU が過負荷である場合は、UE が最後にデータを送信したあと、Connected Inactive 状態から Idle 状態に遷移するまでの時間を長く設定することにより、メモリの負荷を増加させる代わりに CPU の負荷を削減することが可能である。またその逆に、メモリが過負荷である場合は、この時間を短く設定することにより、CPU の負荷を増加させる代わりにメモリの負荷を削減できる。この方法では、従来手法のように需要に合わせてリソースを変化させるのではなく、リソース量に合わせてリソースの需用量を変化させることが特徴である。

この仕組みにより、CPU とメモリのリソース消費の予測が難しい場合や、変動が激しいネットワークであっても、短いタイムスケールで各リソース負荷の制御が可能である。そして、CPU およびメモリ双方のリソース利用率の向上により収容可能な端末の増加が期待できる。また、将来的には、server disaggregation やスケールアウトの考え方と組み合わせることにより、より効果のあるリソース管理が可能であると考えられる。

2　モバイルネットワークアーキテクチャ

2.1　ネットワーク構成

図 1 にモバイルネットワークを示す。モバイルネットワークは以下のノードから構成される。

- UE
- eNodeB
- MME
- S/PGW
- HSS

UE および eNodeB は複数台、その他のノードは 1 台づつ存在する。

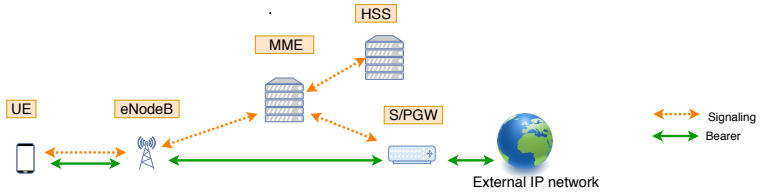

図 1: LTE/EPC ネットワークモデル

2.2　ステート遷移

従来のモバイルネットワークアーキテクチャにおける、ネットワークの状態遷移図を図 2 に示す。

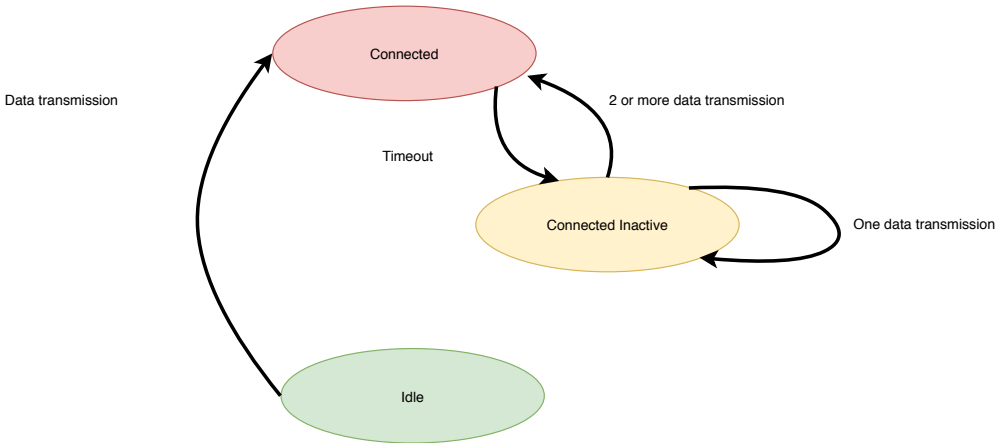

図 2: 従来の状態遷移モデル

2.2.1 Connected 状態

データの送受信が可能である状態。UE およびネットワークに UE のコンテキストが保存されており、ベアラも確立されている。

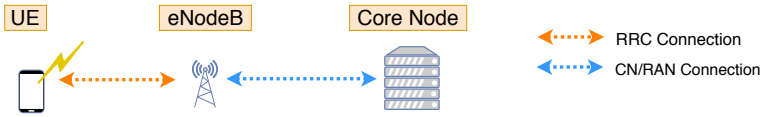

図 3: Connected

2.2.2 Idle 状態

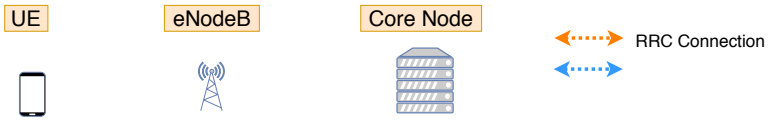

図 4: Idle

2.2.3 Connected Inactive 状態

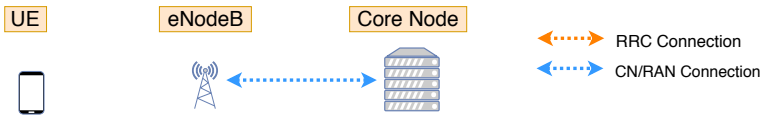

図 5: Connected Inactive

3　提案手法

本研究では、M2M/IoT 端末が最後にデータを送信した後に、Idle 状態に遷移するまでの時間を動的に制御することにより、CPU とメモリのリソース利用率を最適化し、収容可能な端末の台数を最大化する方法を提案する。

3.1　Idle timer の設定

提案手法では、M2M/IoT 端末が最後にデータを送信した後に、Idle 状態に遷移するまでの時間 (Idle timer) を CPU およびメモリリソースの負荷状況に応じて動的に決定する。Idle timer の設定によってコアノードのリソースの需要が変化する。提案手法では、ユーザ端末と M2M/IoT 端末の存在割合に応じて、収容可能な UE 台数を最大化するような Idle timer を計算によって導出する。処理のフローを図 6 に示す。常に CPU とメモリのリソース負荷を監視し、CPU が過負荷であれば Idle timer を増加させ、反対にメモリが過負荷であれば Idle timer を減少させる。

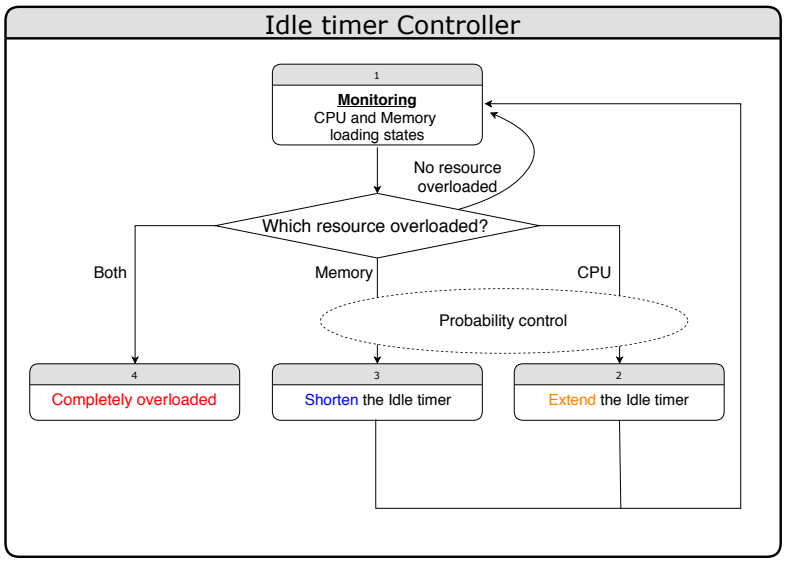

図 6: The proposed method

3.2　ステート遷移

提案手法を適用した場合の状態遷移図を図 7 に示す。従来の状態遷移に対して、Connected Inactive 状態から Idle 状態への遷移が追加されている。

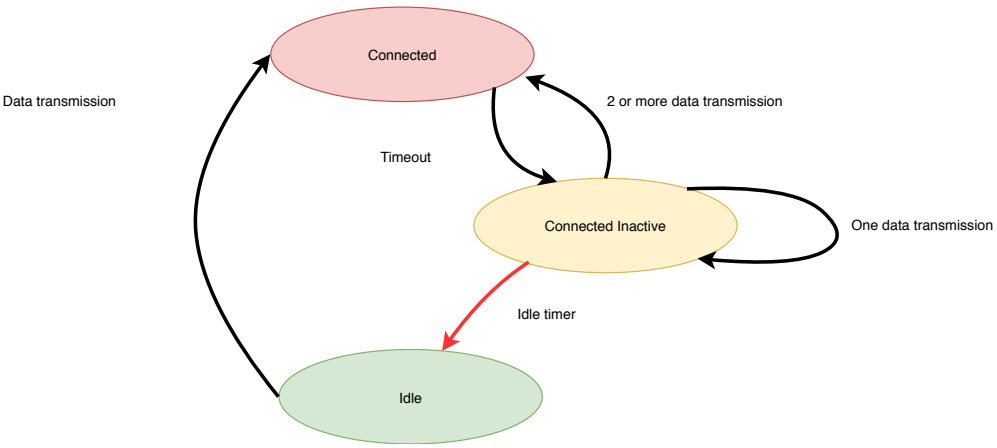
Connected
Data transmission
2 or more data transmission
Timeout
Connected Inactive
One data transmission
Idle timer
Idle

図 7: ステート遷移

4　解析モデル

4.1　状態遷移モデル

UE が最後にデータを送信した後、Connected Inactive 状態に遷移するまでの時間を T_c、Idle timer を T_i と定義する。ある UE におけるデータの送信間隔を T_h と定義した時、この UE の状態遷移を図 8 に示す。まず、UE はデータを送信すると、Connected 状態へ遷移する。データ送信後 T_c 時間 Connected 状態を維持したのち、Connected Inactive 状態へ遷移する。そして、データ送信後 T_i 後に Idle 状態へ遷移する。

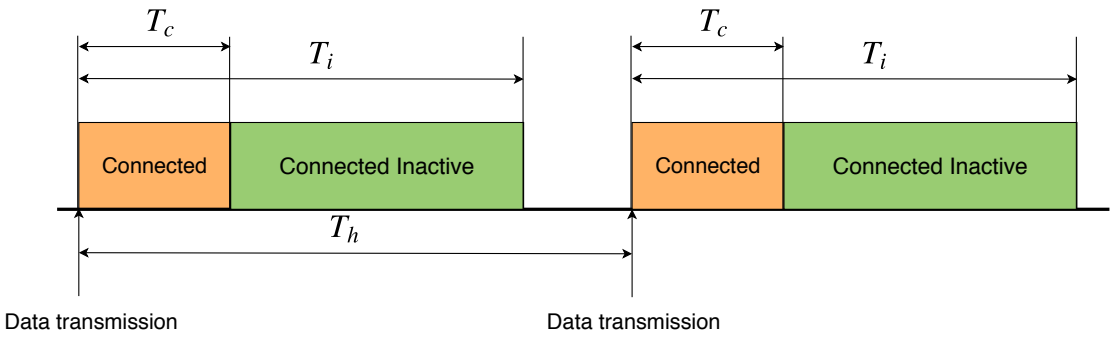

図 8: 状態遷移モデル

4.2　解析環境

今回の研究はシミュレーションを用いてネットワークの性能評価を行う。シミュレーションのタイムステップは 1 分から 5 分程度を想定している。各タイムステップでは、UE の通信特性と Idle timer に基づき EPC の負荷を算出する。

EPC の負荷の算出のイメージ図を図 9 に示す。UE の通信特性と Idle timer を入力とし、EPC の負荷を出力とする。EPC のメモリ負荷の導出のために上野さんの実験結果を用いる。

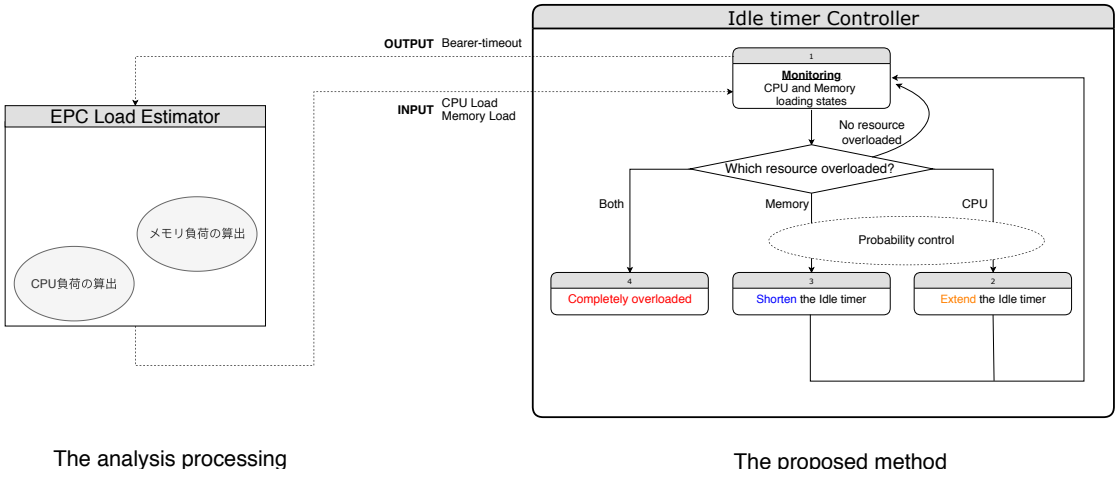

図 9: 解析環境

4.2.1 CPU 負荷の算出

ネットワーク全体におけるシグナリングの発生レートを求める。N_{UE} 台の UE がネットワークに存在すると仮定した時の UE の集合 $\boldsymbol{U}$ を、$\boldsymbol{U} = \{u_1, u_2, \ldots, u_{N_{UE}}\}$ と定義する。ある UE u_h ($u_h \in \boldsymbol{U}$) が 1 秒あたりに発生させるシグナリングの数を s_h と定義する。UE u_h が 2 パケット以上のデータ送信を行う確率を d_h とする。最後のデータ送信から Connected Inactive 状態へ遷移するまでの時間を T_c、Idle timer を T_i、u_h の通信周期を T_h とする。また、状態遷移に伴うシグナリングの発生回数をそれぞれ表 1 のように定義すると、s_h は以下の式 (1) で表せる。ここで留意すべきは、T_h が T_c 以下であるような UE は、一度 Connected 状態に遷移すると、常時 Connected 状態を維持し、状態遷移によるシグナリングが発生しないため、s_h は 0 と定義したことである。

表 1: state signaling

Source	Destination	The number of signaling occurrences
Connected	Connected Inactive	$n_{c\to i}$
Connected Inactive	Connected	$n_{i\to c}$
Connected Inactive	Idle	$n_{i\to d}$
Idle	Connected	$n_{d\to c}$

$$s_h = \begin{cases} 0 & \text{if } T_h \leq T_c \\ \frac{1}{T_h} \cdot (n_{i\to c} + n_{c\to i}) \cdot d_h & \text{if } T_c \leq T_h \leq T_i \\ \frac{1}{T_h} \cdot (n_{d\to c} + n_{c\to i} + n_{i\to d}) & \text{otherwise} \end{cases} \tag{1}$$

1 秒毎にネットワーク全体で発生するシグナリングの合計を S と定義する。S は s_h を用いて以下の式 (2) で表せる。

$$S = \sum_{h=1}^{N_{UE}} s_h \tag{2}$$

この S より、CPU にかかる負荷を算出する。

4.2.2 メモリ負荷の算出

まず、Connected 状態および Connected Inactive 状態、Idle 状態それぞれの UE 数を求める。ある時刻において、UE u_h が Connected 状態である確率を b_h^c、Connected Inactive 状態である確率を b_h^i、Idle 状態である確率を b_h^d と定義する。するとこれらの値は、T_h および T_i、T_c を用いて以

下の式 (3)、(4)、(5) で表せる。

$$b_h^c = \begin{cases} 1 & \text{if } T_h \leq T_c \\ \frac{T_c}{T_h} & \text{otherwise} \end{cases} \tag{3}$$

$$b_h^i = \begin{cases} 0 & \text{if } T_h \leq T_c \\ \frac{T_h - T_c}{T_h} & \text{if } T_c \leq T_h \leq T_i \\ \frac{T_i}{T_h} & \text{otherwise} \end{cases} \tag{4}$$

$$b_h^d = \begin{cases} 0 & \text{if } T_h \leq T_c \\ 0 & \text{if } T_c \leq T_h \leq T_i \\ \frac{T_h - T_i}{T_h} & \text{otherwise} \end{cases} \tag{5}$$

Conneted 状態である UE の平均数および Connected Inactive 状態である UE の平均数、Idle 状態である UE の平均数を B_c、B_i、B_d と定義すと、それらは、は b_h^c および b_h^i、b_h^d を用いて以下の式 (6)、(7)、(8) で表せる。

$$B_c = \sum_{h=1}^{N_{UE}} b_h^c \tag{6}$$

$$B_i = \sum_{h=1}^{N_{UE}} b_h^i \tag{7}$$

$$B_d = \sum_{h=1}^{N_{UE}} b_h^d \tag{8}$$

この B_c、B_i、B_d より、メモリ負荷を算出する。なおその際、上野さんの実験結果を参考にする。

5 先行研究調査

サーバのリソース分離に関する文献 [3] を調査した。この文献では、データセンタのコストを削減することを目的し、サーバリソースの分離を提案している。そして、サーバごとにリソース構成が固定されている従来のデータセンタと、CPU とメモリのリソース構成をサーバから分離したデータセンタとの比較を行っている。評価の結果、リソースを分離することによって、CPU とメモリのオーバープロビジョニングを削減できることを示している。そして、コスト面では（アプリケーションにも依存するが）最大 40%の削減が期待でいると結論づけてる。

文献 [4] では、モバイルネットワークにおいてハンドオーバに関わるシグナリングの発生回数を減らすことを目的とし、Delay Time Algorithem を提案している。このアルゴリズムでは、従来の仕組みであればハンドオーバ処理を行うような状況であっても、あえてハンドオーバ処理を行うタイミングを遅らせることによって、シグナリングの発生を削減することを可能としている。

文献 [5] では、UE が Cennected 状態から idle 状態へ遷移するまでの時間 (inactive timer) を制御することによって、シグナリング発生回数と UE の消費電力を最適化する研究を行っている。inactive timer を大きくすることによって、UE の状態遷移に伴うシグナリングの発生回数を抑えることが可能である一方、UE が長い時間 Connected 状態に止まるために消費電力が増加することを明らかにしている。

文献 [1] では、RRC Connected Inactive state と呼ばれる状態を新しく導入することによって、シグナリングの発生回数及び UE の消費電力を削減できることを示している。

6　今後の課題

- デタッチの処理負荷を求めるため、OAI に基づくデタッチ処理のプログラム行数を調査する。
- UE の通信特性の分布を決定する上で、根拠となるデータを見つける。
- メモリの負荷が二次関数的な増加を示している理由を調査する。調査方法としては、UE 数および eNodeB 数のどちらか一方を固定し、もう一方を変化させつつベアラ確立の実験を行い、メモリ負荷を観測する。もしくは、OAI のソースコードを読みデータの保持に関する実装を調査する。

参考文献

[1] S. Hailu, M. Saily, and O. Tirkkonen, "RRC State Handling for 5G," *IEEE Communications Magazine*, vol. 57, no. 1, pp. 106–113, Jan. 2019.

[2] I. L. Da Silva, G. Mildh, M. S ä ily, and S. Hailu, "A Novel State Model for 5G Radio Access Networks," in *Proceedings of 2016 IEEE International Conference on Communications Workshops (ICC)*, May 2016, pp. 632–637.

[3] M. Mahloo, J. M. Soares, and A. Roozbeh, "Techno-Economic Framework for Cloud Infrastructure: A Cost Study of Resource Disaggregation," in *Proceedings of 2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Sep. 2017, pp. 733–742.

[4] C. Lee and P. Lin, "Modeling Delay Timer Algorithm for Handover Reduction in Heterogeneous Radio Access Networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 1144–1156, Feb. 2017.

[5] Q. Liao and D. Aziz, "Proceedings of Modeling of Mobility-Aware RRC State Transition for Energy-Constrained Signaling Reduction," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2016, pp. 1–7.