進捗報告資料

安達智哉
to-adachi@ist.osaka-u.ac.jp

2019 年 3 月 22 日

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、EPC ノードにおける 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、server disaggregation にはいくつかの課題がある。まず、CPU とメモリのリソースを分離するためには、大きなコストがかかる点が挙げられる。文献 [4]、[5] では、CPU とメモリを分離するためには、両者を結ぶための新しい高帯域ネットワークが必要になると述べている。また、両者の物理的な距離が増加することによって発生する遅延も考慮する必要があるため、新たなメモリアーキテクチャの構築が必要であると述べている。文献 [6] では、メモリを分離するためには、低遅延かつ高帯域のネットワーク接続が必要となるが、それを実装するためにコストは従来と比較して大幅に増加すると述べている。実際、文献 [7] では、Disaggregated Server に基づくインテルのラックスケールアーキテクチャを示しているが、このモデルでも CPU とメモリの分離はできていない。課題の 2 つ目として、短いタイムスケールでの制御が難しいという問題が挙げられる。例えば、ストレージをモジュール化してサーバと分離する手法について述べられている文献 [8] では、時間スケールの細かいストレージ制御を行った場合、ストレージの再割り当て処理に伴うオーバーヘッドが大きくなると述べている。また、頻繁にリソース構成を変化させることは、コスト面や消費電力の面でも不利である (注:まだ根拠はない)。

このように、server disaggregation を用いたリソース制御では、リソース制御に伴うオーバーヘッドの発生が避けては通れない課題となると予想される。特に、モバイルネットワークのように、突発的なトラヒックの増加が発生し、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ネットワークにおいては細かいタイムスケールでの効率的なリソース制御が求められる。

そこで、本研究ではモバイルネットワークに特化した、柔軟かつ効率的な、EPC ノードにおける CPU とメモリ間の負荷のオフロード方法を考案する。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。この時間の再設定処理は、数分単位のオーダーで可能でありかつ、それに伴い発生するオーバーヘッドは、server disaggregation と比較して僅かである (注:提案手法のオーバーヘッドの大小に関する記述は、今後の研究結果によって修正します)。また、既存のシグナリングアーキテクチャに変更を加えることが可能であれば、秒単位の時間スケールでの制御も可能であると考えられる。

しかし、提案手法には限界もある。それは、対応可能なリソース需要に制限があることである。なぜなら、提案手法では、限られた CPU およびメモリのリソースを効率的に利用することは可能であるが、双方のリソースが過負荷になるようなリソース需要には対応できないからである。特に長期的かつ大規模なリソース需要の変化に対しては、server disaggregation を用いたリソース制御の方が適している。また、提案手法を用いなかった場合には常時 Connected Inactive 状態であった UE が、提案手法を用いることにより Idle 状態へと遷移する可能性があるため、データを送受信にかかる遅延時間が増加するなど、QoS の低下が発生する可能性がある。しかし、電気メータや気温計のようなリアルタイム性を必要としない IoT 端末であれば、これらの QoS の低下は無視できると考えられる。

そのため、提案手法と server disaggregation やスケールアウト/スケールインを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。例えば、長期的かつ大規模なリソース制御は server disaggregation を用いて行う一方で、server disaggregation で対応できないような短いタイムスケールでのリソース制御は提案手法を用いて行う。もしくは、リソース需要の大きな変動に対してはスケールアウト/スケールインを用いて対応した上で、細かなリソース需要の変化に対しては提案手法を用いて対応する。上述ように、server disaggregation やスケールアウト/スケールインと提案手法組み合わせ、双方のデメリットを補うことで、モバイルネットワークのリソース制御をより良く実現できると考えられる。

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。

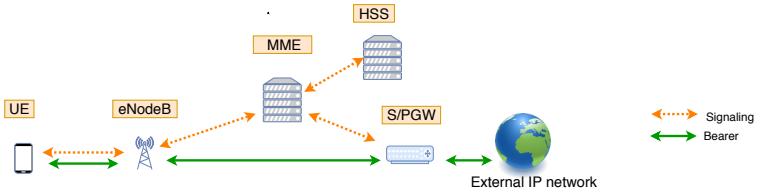

図 1: LTE/EPC ネットワークモデル

2.2　UE のステート遷移

2.2.1　従来のモバイルネットワークにおける UE のステート遷移

UE はデータ送信のタイミングで Idle 状態から Connected 状態へ遷移する。その後、一定時間データの送受信が発生しなければ、再び Idle 状態へ遷移する。図 2

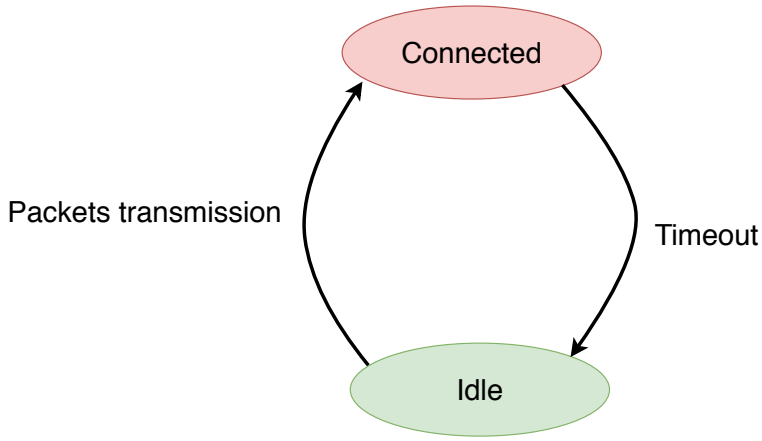

図 2: 従来のモバイルネットワークにおける UE のステート遷移

2.2.2 Connected Inactive を導入した場合における UE のステート遷移

Idle 状態の UE はデータ送信のタイミングで、Connected 状態へ遷移する。その後、一定時間データの送受信が発生しなければ、Connected Inactive 状態へ遷移する。Connected Inactive 状態の UE は、データ送信のタイミングで、Connected 状態へ遷移する。しかし、送信データが 1 パケットのみであった場合は、Connected 状態への遷移を必要としない。また、Connected Inactive 状態の UE は、例外的な処理が発生しない限り、Idle 状態へは遷移しない。図 3

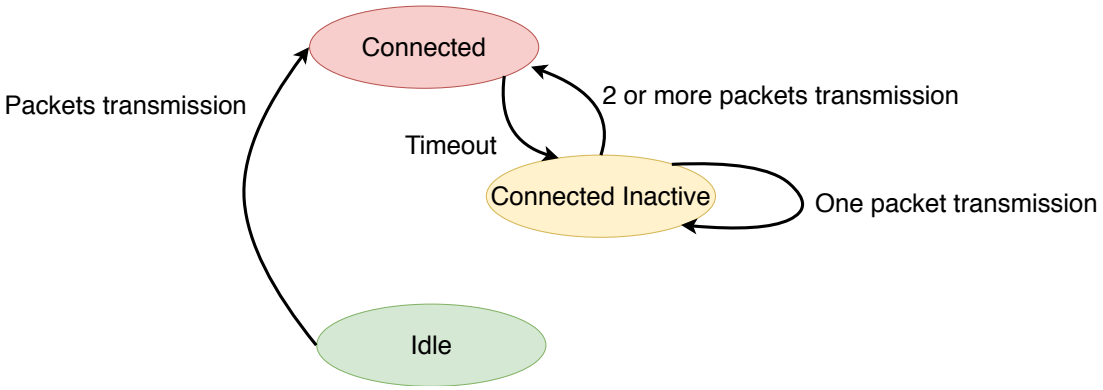

図 3: Connected Inactive を導入した場合における UE のステート遷移

2.2.3 Connected Inactive にタイムアウト制御を適用した場合における UE のステート遷移 (提案手法)

Connected Inactive 状態から Idle 状態への状態遷移を追加したモデルである。図 4

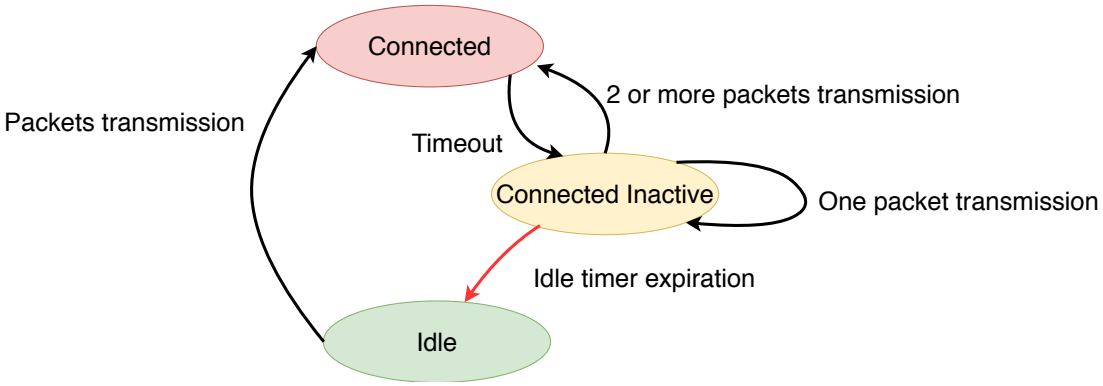

図 4: Connected Inactive にタイムアウト制御を適用した場合における UE のステート遷移

2.3 シグナリング

現在調査中である。

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 5 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。

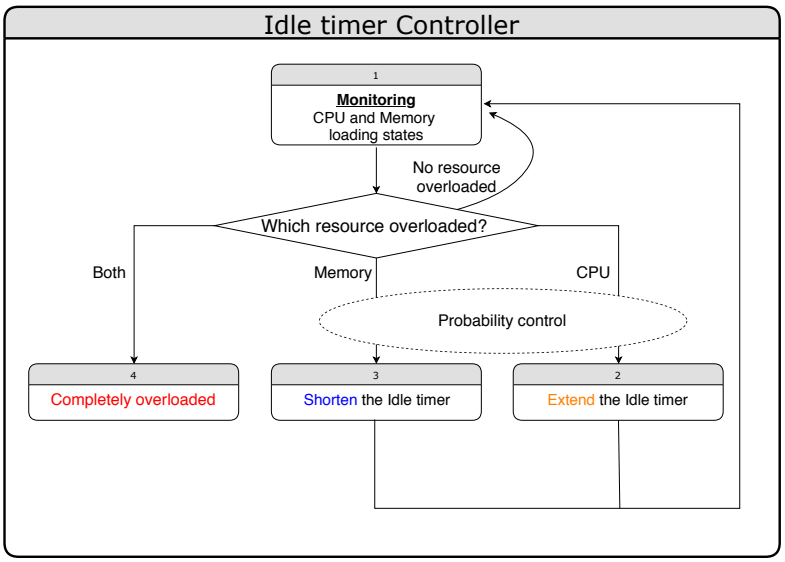

図 5: The proposed method

3.2　ネットワークの状態

3.2.1　Connected 状態

データの送受信が可能な状態である。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。

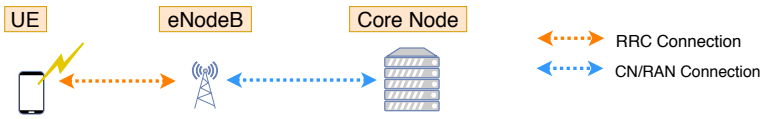

図 6: Connected

3.2.2　Idle 状態

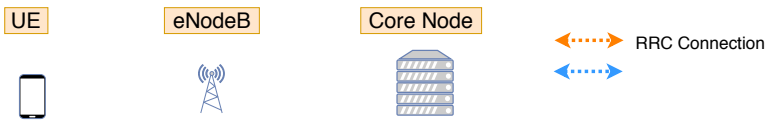

図 7: Idle

3.2.3　Connected Inactive 状態

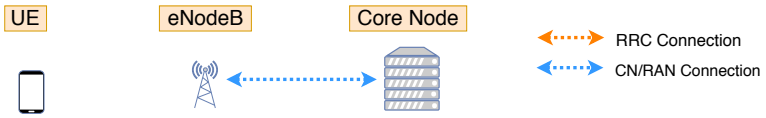

図 8: Connected Inactive

4　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 [9] 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 [10] 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。

5 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] “Intel’s Disaggregated Server Rack,” Moor Insights Strategy, Technical Report (TR), Aug. 2013. [Online]. Available: http://www.moorinsightsstrategy.com/wp-content/uploads/2013/08/Intels-Disagggregated-Server-Rack-by-Moor-Insights-Strategy.pdf

[5] C. Devaki and L. Rainer, “Enhanced Back-off Timer Solution for GTP-C Overload Control,” Feb. 2016. [Online]. Available: http://www.freepatentsonline.com/y2016/0057652.html

[6] B. Abali, R. J. Eickemeyer, H. Franke, C. Li, and M. Taubenblatt, "Disaggregated and Optically Interconnected Memory: When will it be cost effective?" *CoRR*, vol. abs/1503.01416, 2015. [Online]. Available: http://arxiv.org/abs/1503.01416

[7] "Disaggregated Servers Drive Data Center Efficiency and Innovation," Intel Corporation, Technical Report (TR), Jun. 2017. [Online]. Available: https://www.intel.com/content/www/us/en/it-management/intel-it-best-practices/disaggregated-server-architecture-drives-data-center-efficiency-paper.html

[8] S. Legtchenko, H. Williams, K. Razavi, A. Donnelly, R. Black, A. Douglas, N. Cheriere, D. Fryer, K. Mast, A. D. Brown, A. Klimovic, A. Slowey, and A. Rowstron, "Understanding Rack-Scale Disaggregated Storage," in *Proceedings of 9th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 17)*. Santa Clara, CA: USENIX Association, 2017. [Online]. Available: https://www.usenix.org/conference/hotstorage17/program/presentation/legtchenko

[9] 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.

[10] 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.