

進捗報告資料

安達智哉

to-adachi@ist.osaka-u.ac.jp

2019年7月31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、ペ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できないように、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を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そうな状況や変動が大きいような状況においても、どちらかのリソース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。

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、対応可能なり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當時 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。

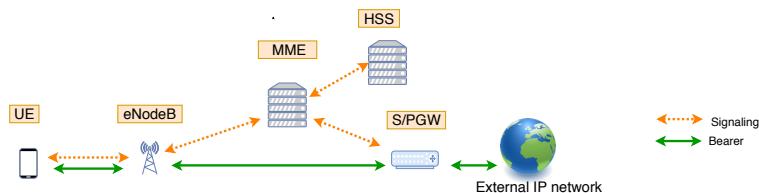


図1: LTE/EPC ネットワークモデル

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

従来のモバイルネットワークにおけるUEのステート遷移を図2に示す。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。

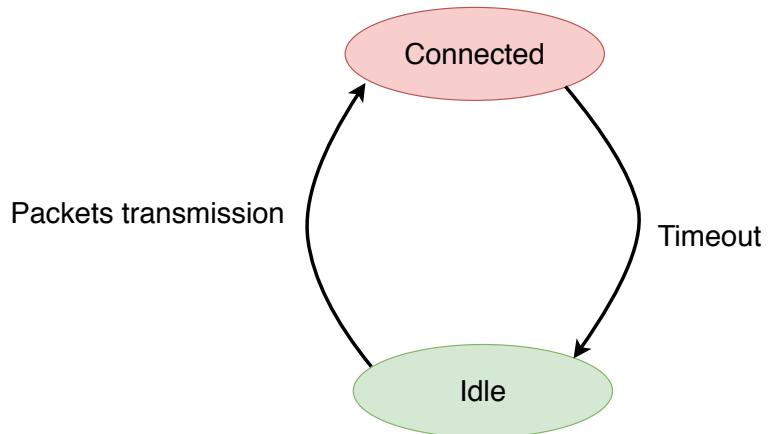


図2: 従来のモバイルネットワークにおけるUEのステート遷移

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

Connected Inactive を導入した場合における UE のステート遷移を図 3 に示す。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、送信データサイズが小さい場合は、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。

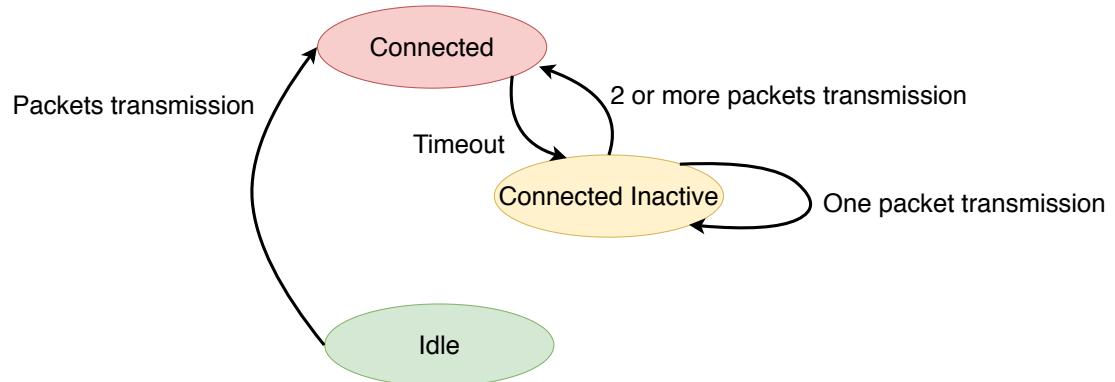


図 3: Connected Inactive を導入した場合における UE のステート遷移

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 にタイムアウト制御を適用した場合における UE のステート遷移を図 4 に示す。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。

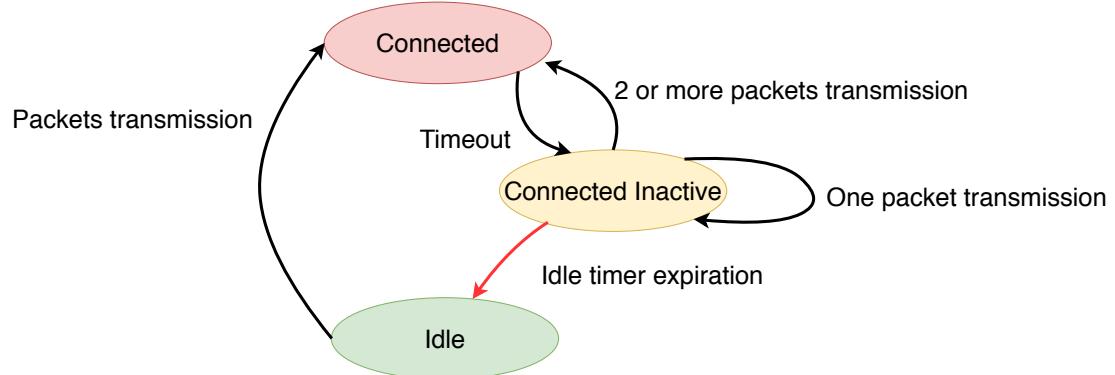


図 4: Connected Inactive にタイムアウト制御を適用した場合における UE のステート遷移

3 MME の負荷の算出方法

MME の負荷はシグナリングを処理することによって発生する CPU 負荷と、UE のコンテキストを保持することによって発生するメモリ負荷に分けられる。CPU 負荷とメモリ負荷のバランスは、UE の状態遷移を制御することにより変えることが可能である。具体的には、UE が最後にデータを送信したあと、Idle 状態に遷移するまでの時間 (T^i) を設定することで負荷のオフロードが可能である。例えば、CPU が過負荷である場合は、 T^i を長く設定することにより、メモリの負荷が増加し、CPU の負荷が低下する。またその逆に、メモリが過負荷である場合は、 T^i を短く設定することにより、CPU の負荷が増加し、メモリの負荷が低下する。

負荷を算出するにあたって以下のように条件を設定した。

- UE ごとに通信周期は固定であり、途中で変化することはない。
- データ送信にかかる時間は UE の通信周期と比べて十分小さいものとし、送信が失敗することはないと仮定する。
- UE の総数は有限かつ一定である。
- 算出する負荷は、UE の通信周期が理想的に分散しており、毎秒均等な負荷が MME に対して発生する場合の値である。現実的なモデルを考える場合は、UE の通信の同期などにより、MME に対して発生する負荷は時間的に変動することを留意する必要がある。
- MME を含めた全ネットワークノードのリソースおよびリンク帯域は十分大きいと仮定する。つまり、どのネットワーク要素もオーバーロード状態になることはない。

3.1 CPU 負荷の算出

CPU 負荷は 1 秒あたりに発生させるシグナリングの数を基に算出する。 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 \mathbf{U} を、 $\mathbf{U} = \{u_1, u_2, \dots, u_{N_{\text{UE}}}\}$ と定義する。ある UE u_h ($u_h \in \mathbf{U}$) が 1 秒あたりに発生させるシグナリングの数 (=CPU 負荷) を c_h と定義する。Connected Inactive 状態では、小さいデータ量であれば、コントロールプレーンを使ってデータ送信を完了させることができる。この場合、Connected 状態への状態遷移は発生しない。UE u_h がデータ送信を行う際に、そのデータサイズが状態遷移が発生しない送信が可能であるデータサイズの上限を超える割合を 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^{ci} 、最後のデータ送信から Idle 状態へ遷移するまでの時間を 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、 c_h は以下の式 (1) で表せる。ここで留意すべきは、 T_h が T^{ci} 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、 c_h は 0 と定義したことである。また、私の研究では T^i を制御するのであるが、この値が T^{ci} 以下になるような設定はしないことを前提にしている。そのため以下の式でも $T^i >= T^{\text{ci}}$ が常に成り立つものとしている。

$$c_h = \begin{cases} \frac{1}{T_h} \cdot s_{\text{MME}}^{\text{c} \rightarrow \text{c}} & \text{if } T_h \leq T^{\text{ci}} \\ \frac{1}{T_h} \cdot (s_{\text{MME}}^{\text{ci} \rightarrow \text{c}} + s_{\text{MME}}^{\text{c} \rightarrow \text{ci}}) \cdot d_h + \frac{1}{T_h} \cdot s_{\text{MME}}^{\text{ci} \rightarrow \text{ci}} \cdot (1 - d_h) & \text{if } T^{\text{ci}} < T_h \leq T^i \\ \frac{1}{T_h} \cdot (s_{\text{MME}}^{\text{i} \rightarrow \text{c}} + s_{\text{MME}}^{\text{c} \rightarrow \text{i}} + s_{\text{MME}}^{\text{ci} \rightarrow \text{i}}) & \text{otherwise} \end{cases} \quad (1)$$

表 1: state signaling

Source	Destination	The number of signaling occurrences
Connected	Connected	$s_{\text{MME}}^{\text{c} \rightarrow \text{c}}$
Connected Inactive	Connected Inactive	$s_{\text{MME}}^{\text{ci} \rightarrow \text{ci}}$
Connected	Connected Inactive	$s_{\text{MME}}^{\text{c} \rightarrow \text{ci}}$
Connected Inactive	Connected	$s_{\text{MME}}^{\text{ci} \rightarrow \text{c}}$
Connected Inactive	Idle	$s_{\text{MME}}^{\text{ci} \rightarrow \text{i}}$
Idle	Connected	$s_{\text{MME}}^{\text{i} \rightarrow \text{c}}$

ここで、今回の評価では T^i をパラメータとして変化させることを想定している。そこで、式(1)を T^i に関する関数として捉えることにする。 T^i を表すパラメータを t と置いた時、式(1)は以下の式(2)に示す、 t に関する関数として表せる。

$$c_h(t) = \begin{cases} \frac{1}{T_h} \cdot s_{\text{MME}}^{\text{c} \rightarrow \text{c}} & \text{if } T_h \leq T^{\text{ci}} \\ \frac{1}{T_h} \cdot (s_{\text{MME}}^{\text{ci} \rightarrow \text{c}} + s_{\text{MME}}^{\text{c} \rightarrow \text{ci}}) \cdot d_h + \frac{1}{T_h} \cdot s_{\text{MME}}^{\text{ci} \rightarrow \text{ci}} \cdot (1 - d_h) & \text{if } T^{\text{ci}} < T_h \leq t \\ \frac{1}{T_h} \cdot (s_{\text{MME}}^{\text{i} \rightarrow \text{c}} + s_{\text{MME}}^{\text{c} \rightarrow \text{ci}} + s_{\text{MME}}^{\text{ci} \rightarrow \text{i}}) & \text{otherwise} \end{cases} \quad (2)$$

T^i を t とした時に、ネットワーク全体で 1 秒毎に発生する CPU 負荷を $C(t)$ と定義する。 $C(t)$ は $c_h(t)$ を用いて以下の式(3)で表せる。

$$C(t) = \sum_{h=1}^{N_{\text{UE}}} c_h(t) \quad (3)$$

3.2 メモリ負荷の算出

まず、UE u_h が Connected 状態である時間割合を τ_h^{c} 、Connected Inactive 状態である時間割合を τ_h^{ci} 、Idle 状態である時間割合を τ_h^{i} と定義し、これらを求める。これらの値は、 T_h および T^i 、 T^{ci} を用いて以下の式(4)、(5)、(6)で表せる。

$$\tau_h^{\text{c}} = \begin{cases} 1 & \text{if } T_h \leq T^{\text{ci}} \\ \frac{T^{\text{ci}}}{T_h} \cdot d_h + \frac{0}{T_h} \cdot (1 - d_h) & \text{if } T^{\text{ci}} < T_h \leq T^i \\ \frac{T^{\text{ci}}}{T_h} & \text{otherwise} \end{cases} \quad (4)$$

$$\tau_h^{\text{ci}} = \begin{cases} 0 & \text{if } T_h \leq T^{\text{ci}} \\ \frac{T_h - T^{\text{ci}}}{T_h} \cdot d_h + \frac{T_h}{T_h} \cdot (1 - d_h) & \text{if } T^{\text{ci}} < T_h \leq T^i \\ \frac{T^i - T^{\text{ci}}}{T_h} & \text{otherwise} \end{cases} \quad (5)$$

$$\tau_h^{\text{i}} = \begin{cases} 0 & \text{if } T_h \leq T^{\text{ci}} \\ 0 & \text{if } T^{\text{ci}} < T_h \leq T^i \\ \frac{T_h - T^i}{T_h} & \text{otherwise} \end{cases} \quad (6)$$

ここで、今回の評価では T^i をパラメータとして変化させることを想定している。そこで、式(4)、(5)および(6)を T^i に関する関数として捉えることにする。 T^i を表すパラメータを t と置いた時、これらの式は以下の式(7)、(8)および(9)に示す、 t に関する関数として表せる。

$$\tau_h^c(t) = \begin{cases} 1 & \text{if } T_h \leq T^{ci} \\ \frac{T^{ci}}{T_h} \cdot d_h + \frac{0}{T_h} \cdot (1 - d_h) & \text{if } T^{ci} < T_h \leq t \\ \frac{T^{ci}}{T_h} & \text{otherwise} \end{cases} \quad (7)$$

$$\tau_h^{ci}(t) = \begin{cases} 0 & \text{if } T_h \leq T^{ci} \\ \frac{T_h - T^{ci}}{T_h} \cdot d_h + \frac{T_h}{T_h} \cdot (1 - d_h) & \text{if } T^{ci} < T_h \leq t \\ \frac{t - T^{ci}}{T_h} & \text{otherwise} \end{cases} \quad (8)$$

$$\tau_h^i(t) = \begin{cases} 0 & \text{if } T_h \leq T^{ci} \\ 0 & \text{if } T^{ci} < T_h \leq t \\ \frac{T_h - t}{T_h} & \text{otherwise} \end{cases} \quad (9)$$

次に、各状態におけるメモリ負荷をそれぞれ表2のように定義すると、 T^i を t とした時に、UE u_h が MME に与えるメモリ負荷の平均 $m_h(t)$ は以下の式(10)で表せる。

表 2: 各状態における UE1 台当たりの MME のメモリ負荷

state	load of MME memory
Connected	m_{MME}^c
Connected Inactive	m_{MME}^{ci}
Idle	m_{MME}^i

$$m_h(t) = m_{\text{MME}}^c \cdot \tau_h^c(t) + m_{\text{MME}}^{ci} \cdot \tau_h^{ci}(t) + m_{\text{MME}}^i \cdot \tau_h^i(t) \quad (10)$$

T^i を t とした時に、MME に対してネットワーク全体で発生する平均的なメモリ負荷の合計を $M(t)$ と定義する。 $M(t)$ は $m_h(t)$ を用いて以下の式(11)で表せる。

$$M(t) = \sum_{h=1}^{N_{\text{UE}}} m_h(t) \quad (11)$$

MME が許容できるメモリ負荷および CPU 負荷の最大値をそれぞれ C^{\max} 、 M^{\max} とする。この時、以下の条件(12)に示した 2 つの条件を同時に満たすような t の値が存在するならば、Idle Timer を t に設定することによって、与えられた UE は全て収容可能であると言える。

$$\begin{aligned} C(t) &\leq C^{\max} \\ M(t) &\leq M^{\max} \end{aligned} \quad (12)$$

4 パラメータ設定

4.1 シグナリング数の調査

文献 [9] および [2] を調査することにより、状態遷移に伴うシグナリングの発生数が明らかになった。図 5 に示す状態遷移図と共に、状態遷移に伴って発生するシグナリングに関する情報を、表 3 に示す。

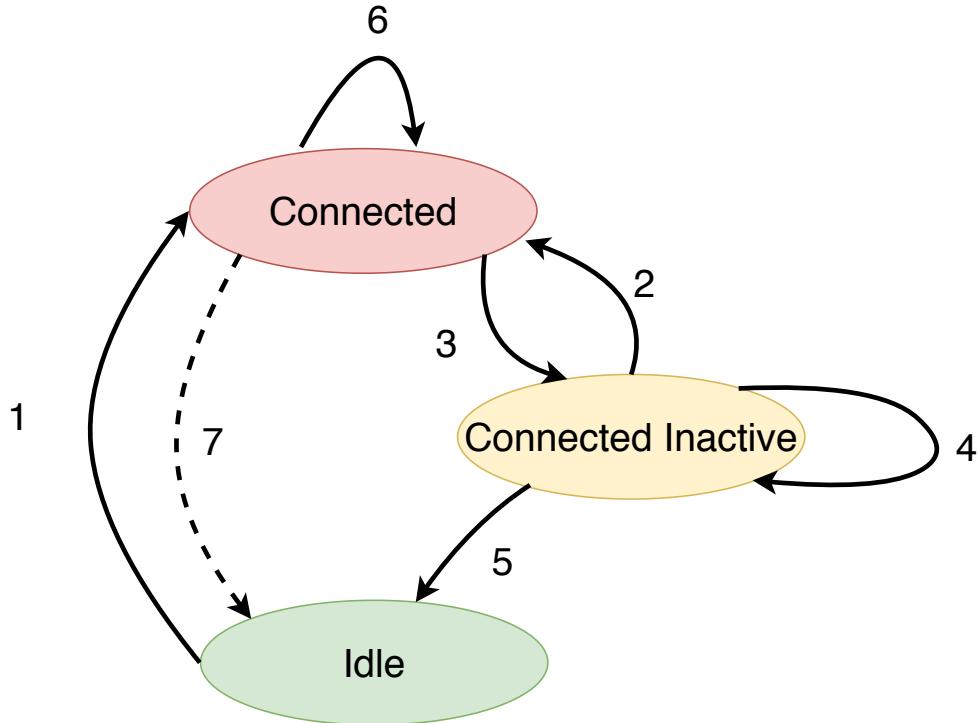


図 5: state transition

表 3: Signaling Load

遷移 ID	シグナリング処理数				遷移条件
	UE	RAN	MME	SGW	
1	9	12	5	2	Packets transmission
2	5	5	0	0	2 or more packets transmission
3	1	1	0	0	Connected timer expiration
4	4	4	0	0	One packet transmission
5	0	3	5	2	Connected Inactive timer expiration
6	0	0	0	0	Packets transmission
7	1	4	5	2	Connected timer expiration

4.2 MMEにおけるメモリ負荷の算出

OAIのソースコードの調査結果と文献 [2] で示されている、RRC Connected Inactive 状態から Connected 状態へ遷移する際のシグナリング図(図 6)を参考にすることにより、各状態にある UE が MME に与えるメモリ負荷を推定することができた。結果を表 4 に示す。

図 6 を見ると、RRC Connected Inactive 状態から Connected 状態へ遷移する際には UE-RAN 間のシグナリングが 5 回発生している一方、コアネットワーク側にはシグナリングは発生していないことがわかる。よって RRC Connected Inactive 状態と Connected 状態では MME の状態には変化がなく、メモリ負荷も同じであると言える。

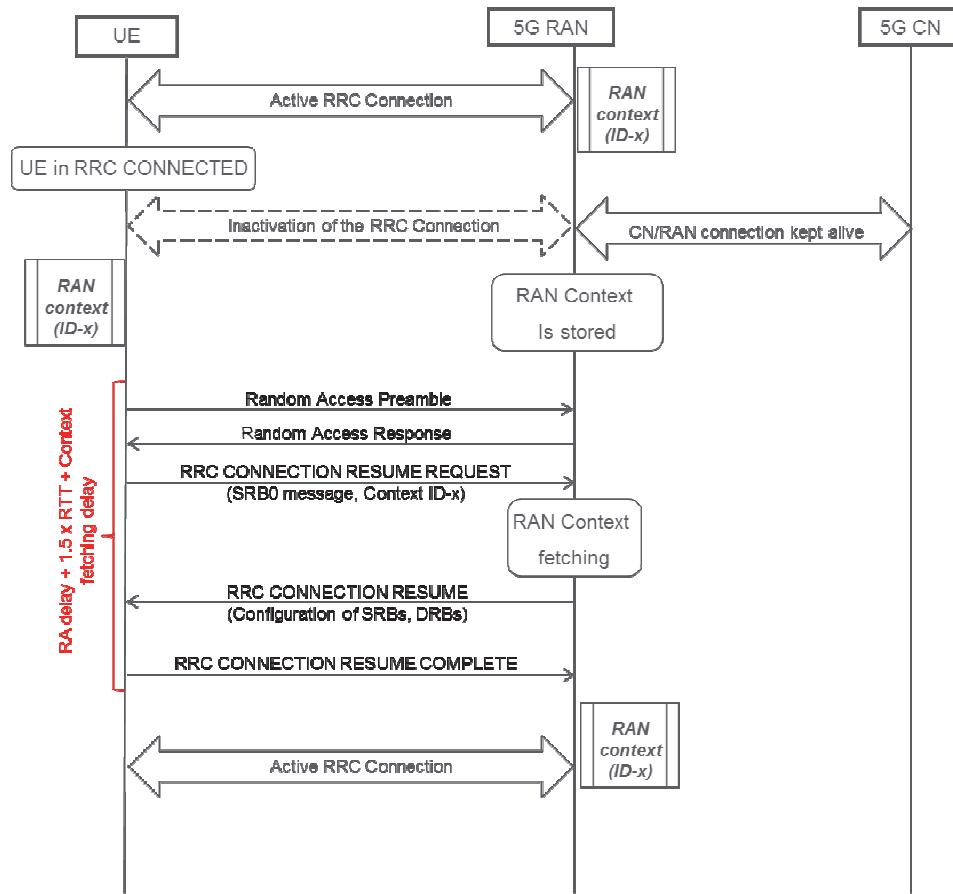


図 6: Signaling for the RRC CONNECTED INACTIVE to RRC CONNECTED transition for the novel state model

表 4: MME が保持する情報

UE のステート	情報名	情報量 (bit)
Connected	<code>ue_description_s</code>	408
	<code>ue_context_s</code>	17470
Connected Inactive	<code>ue_description_s</code>	408
	<code>ue_context_s</code>	17470
Idle	<code>ue_description_s</code>	408

4.3 MME が許容できるメモリ負荷の推定

メモリ負荷は MME に搭載されているメモリのサイズや一般的なサーバに搭載されているメモリサイズなどと比較することにより、どの程度までの負荷が許容されるのかを議論することが可能である。今回は 1000MB とした。

4.4 MME が許容できる CPU 負荷の推定

シグナリングを処理することによって CPU に負荷がかかることが予想できる。しかし、一般的にシグナリング処理数と CPU 負荷との関係は自明でない。そこで今回は、上野さんの実験結果を参考にしつつ、MME が許容できる CPU の範囲において、どの程度の数のシグナリングが処理できるのかを推定した。

上野さんの論文 [10] に示されている結果を図 7 および図 8 に示す。図 7 では、128 台の UE が、特定の同期精度でアタッチ処理を開始した時に発生するレイテンシを示している。同期精度は T_{expect} というパラメータで表現されている。 $T_{expect} = t$ である時は、特定の時刻から t 秒の範囲内でランダムに全 UE がアタッチ処理を開始することを意味する。この図を見ると T_{expect} の値が 1.6 以下になると、急激にレイテンシが増加していることがわかる。

また、図 8 では、レイテンシの内訳を示している。これを見ると、 T_{expect} の値が 1.6 以下の時は、MME のレイテンシが大幅に増加していることがわかる。また、MME のレイテンシが全体のレイテンシに対して大部分を占めていることもわかる。

以上の理由から、 T_{expect} が 1.6 以下の時に、128 台の UE がアタッチ処理を開始した場合、MME の処理にかかる遅延が大幅に増加することが分かる。つまり、 T_{expect} 1.6 以下の状況で、128 台の UE がアタッチ処理を開始した時に発生する負荷は、MME をオーバーロード状態にする。

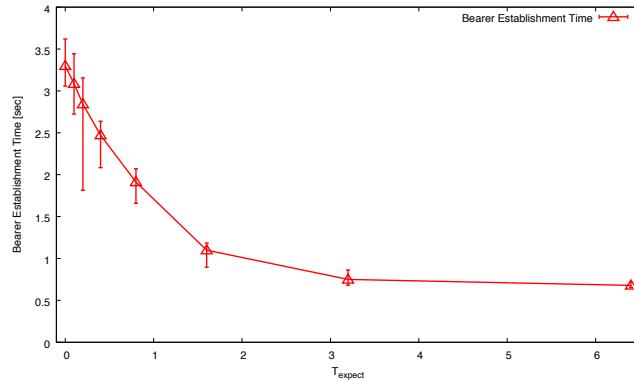


図 7: Relationship between T_{expect} and bearer establishment time

以上の調査から、毎秒約 80 台の UE のアタッチ処理を実行する際に発生する負荷以下ならば、CPU はオーバーロード状態にならないと言える。図 9 に示すように、MME は 1 つのアタッチ処理完了するために、15 回のシグナリング処理を行っている。このことから、毎秒約 1200 回のシグナリング処理数以下であれば、CPU はオーバーロード状態にならないと推定できる。

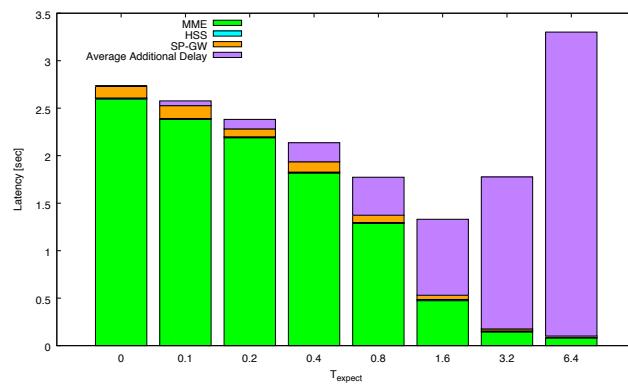


図 8: Relationship between T_{expect} and signaling processing time on each EPC node

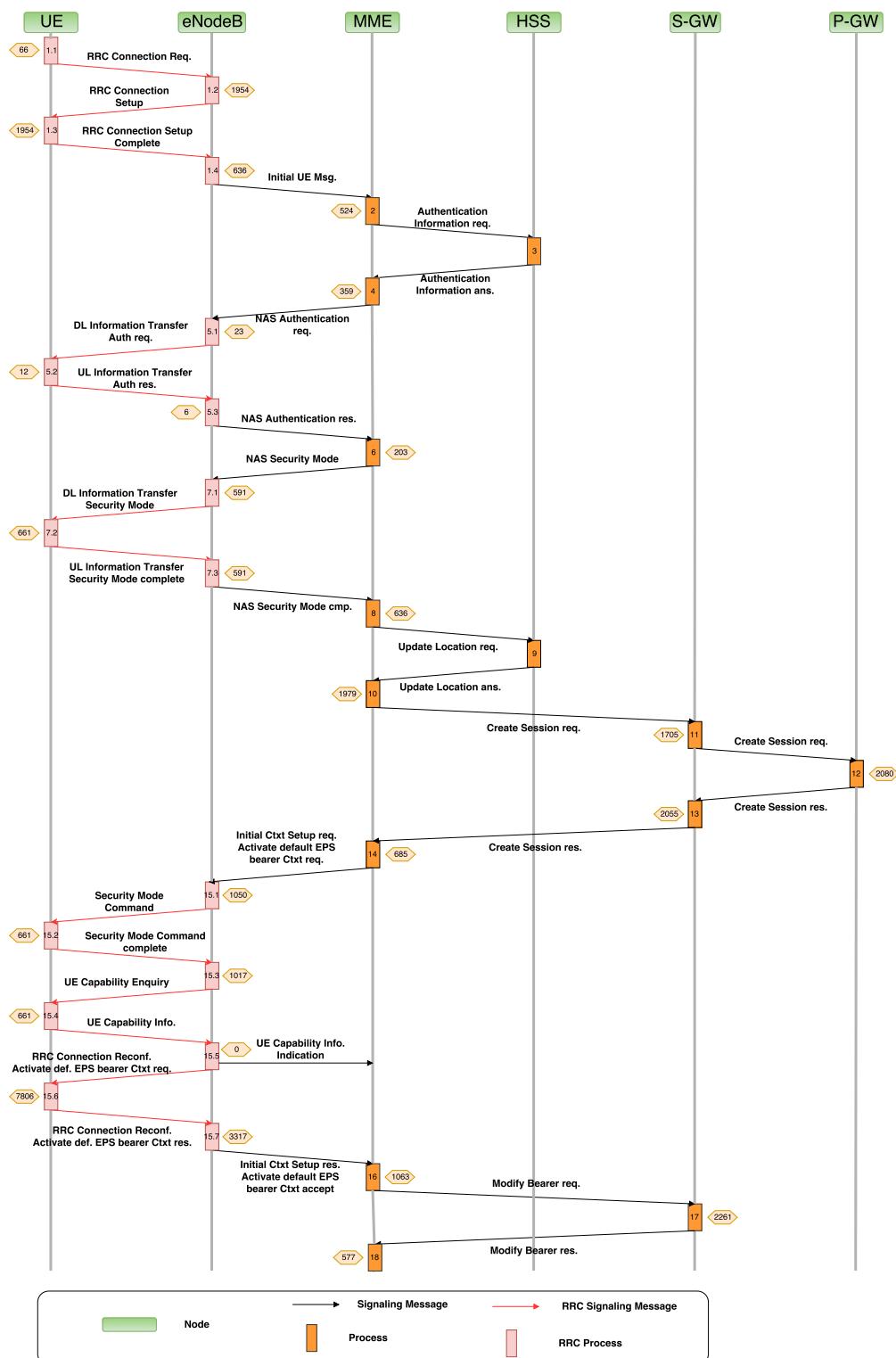


図 9: アタッチのシグナリングフロー

5 MME 負荷の試算

第 3.2 章で調査したメモリ負荷および、第 4.1 章で調査した CPU 負荷を参考にし、第 3 章で述べた数式を用いることで、MME に発生する負荷を概算することができる。

5.1 UE の試算 1

以下のように条件やパラメータを仮定した。

- UE ごとに通信周期は固定であり、途中で変化することはない。
- 最後の送信が終了したあと、Connected 状態の UE が Connected Inactive 状態へ遷移するまでの時間 (T^{ci}) は全 UE で共通かつ不変の値として $T^{ci} = 10s$ とした。
- UE の送信するデータサイズは十分大きいものとする。つまり、データ送信を行うタイミングで必ず Connected 状態に遷移するものとする ($d_h = 1$)。
- UE の通信周期に対する UE 台数の分布は以下の図 10、図 11、図 12 に示す、3 つのデータセットを用意した。それぞれ、ネットワークに存在する UE の台数は 17,400 台、469,200 台、474,600 台である。また、通信周期に対して UE の台数は一様分布である。
- データ送信にかかる時間は UE の通信周期と比べて十分小さいものとし、送信が失敗することはないと仮定する。

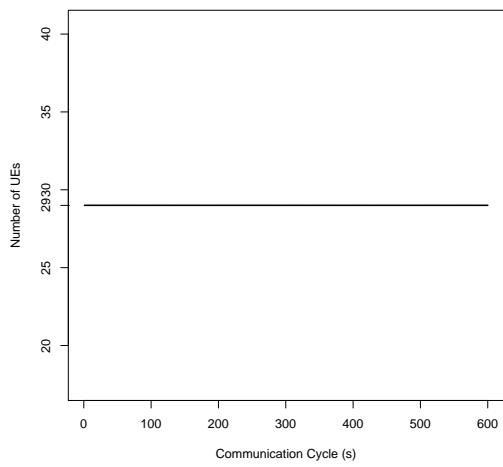


図 10: UE 台数と通信周期 (データセット 1)

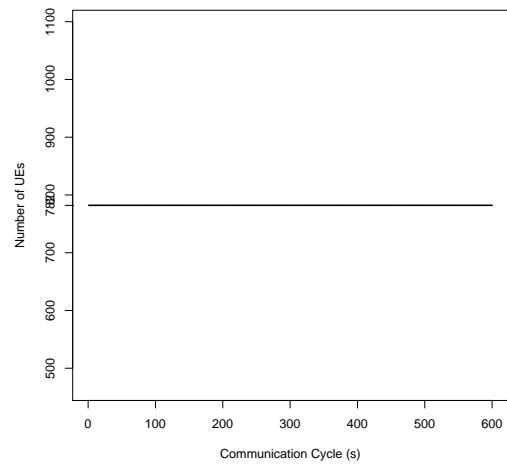


図 11: UE 台数と通信周期 (データセット 2)

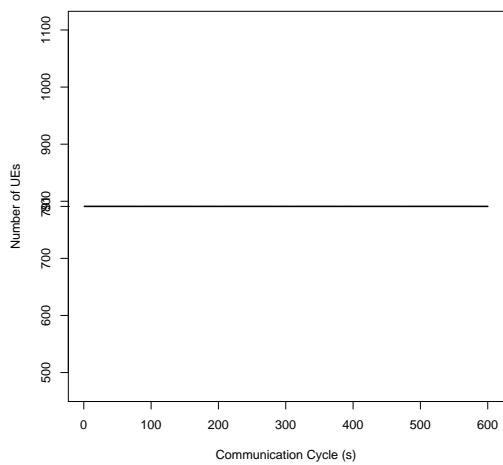


図 12: UE 台数と通信周期 (データセット 3)

MME の CPU 負荷と MME のメモリ負荷の関係を示した図を図 13 に示す。データセット 1 は、 T^i を最小値 (10s) に設定したとしても、シグナリング発生数が MME が処理可能な限界値 (1200 回/s) を超えないような UE の一様分布のうち、最も UE 台数が多い分布である。データセット 2 は、 T^i を最大値 (600s) に設定したとしても、メモリ消費量が MME の限界値 (1GB) を超えないような UE の一様分布のうち、最も UE 台数が多い分布である。データセット 3 は、 T^i を適切に設定した時には、シグナリング発生数およびメモリ消費量が MME の限界値を超えないような UE の一様分布のうち、最も UE 台数が多い分布である。

これらの結果を見ると、今回の条件においては、メモリ負荷の限界値よりもシグナリング発生数の限界値の方が、制約として厳しいことが分かる。また、Idle Timer を適切に設定して、CPU 負荷とメモリ負荷を相互にオフロードさせる効果はほとんどないと言える。実際、Idle Timer を最大に設定した場合 (データセット 2) と Idle Timer を適切に設定した場合 (データセット 3) では収容可能な UE 台数は 1.2%程度しか増加していない。Idle Timer に大きな値を設定し、CPU 負荷をメモリ負荷にオフロードする時に、ネットワーク全体で収容できる UE の台数が向上することが分かる。

また、データセット 1、2、3 のグラフを比較することによって、UE の台数が変化した時のグラフの変化を見ることができる。まず、y 軸切片に着目する。y 軸切片は、Idle Timer を最大に設定して CPU 負荷を全てメモリ負荷にオフロードした場合において、メモリに対して発生する負荷を示す値である。データセット 1、2、3 の y 軸切片の値はそれぞれ、37MB、1000MB、1011MB である。UE 台数がそれぞれ、17,400 台、469,200 台、474,600 台であったことを踏まえると、UE 台数と y 軸切片との間には比例の関係があることが分かる。つまり、UE の分布の傾向 (今回の場合、通信周期が 1s から 600s の範囲内で UE は一様分布しているという条件) が同じであるならば、Idle Timer を最大に設定して CPU 負荷を全てメモリ負荷にオフロードした場合に発生するメモリ負荷は、ネットワークに存在する UE の台数に比例する。

同様に、x 軸切片付近 (※メモリ負荷は 0 にはならないため、厳密には x 軸切片ではない) に着目しつつ、データセット 1、2、3 のグラフを比較する。x 軸切片付近の CPU 負荷は、Idle Timer を最小に設定してメモリ負荷を出来るだけ CPU 負荷にオフロードした場合に発生する、CPU 負荷を示す値である。データセット 1、2、3 のこの値はそれぞれ、1,173.3/s、31,639.8/s、32,003.9/s である。UE 台数がそれぞれ、17,400 台、469,200 台、474,600 台であったことを踏まえると、UE 台数とこの値との間には比例の関係があることが分かる。つまり、UE の分布の傾向 (今回の場合、通信周期が 1s から 600s の範囲内で UE は一様分布しているという条件) が同じであるならば、Idle Timer を最小に設定してメモリ負荷を出来るだけ CPU 負荷にオフロードした時の CPU 負荷は、ネットワークに存在する UE の台数に比例する。

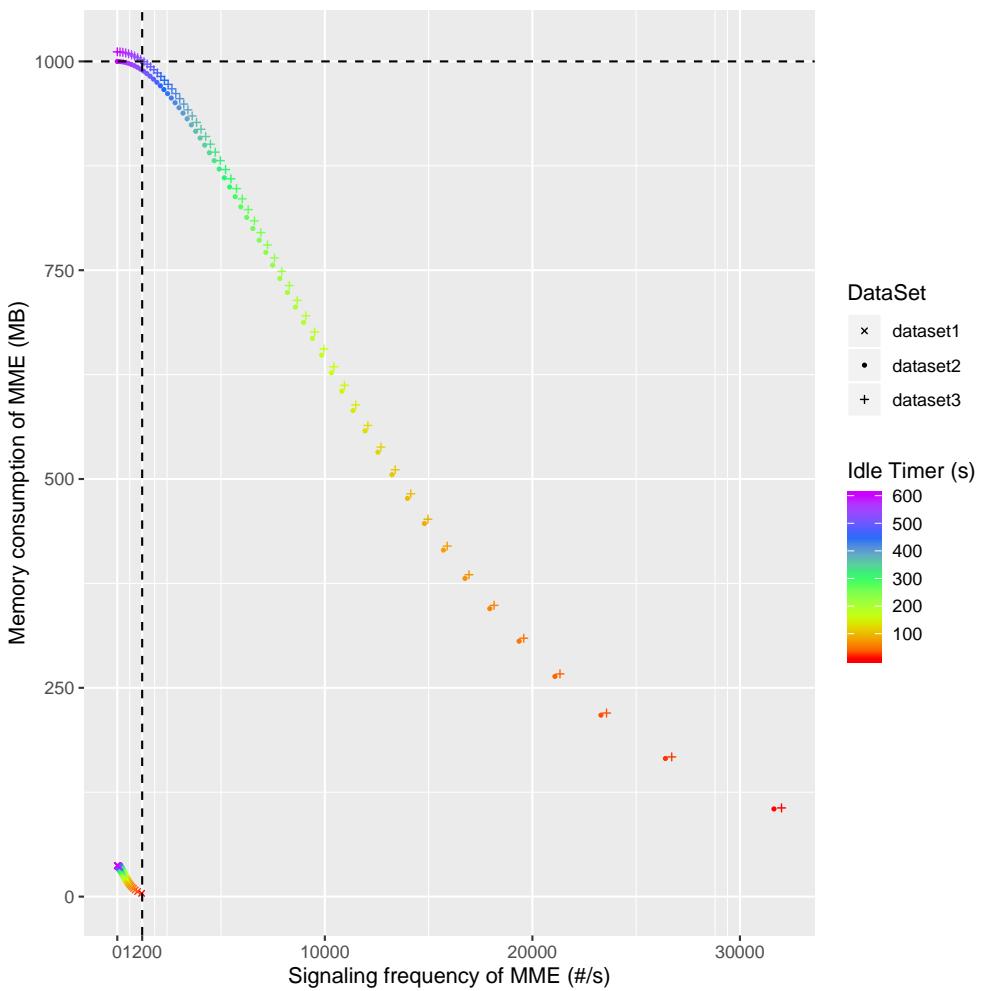


図 13: MME に対して発生する、1sあたりのシグナリング数に対する MME のメモリ負荷の全体像 (データセット 1,2,3)

5.2 UE の試算 2

以下のように条件やパラメータを仮定した。 $d_h = 0$ である点以外は、第 5.1 節の仮定と同じである。

- UE ごとに通信周期は固定であり、途中で変化することはない。
- 最後の送信が終了したあと、Connected 状態の UE が Connected Inactive 状態へ遷移するまでの時間 (T^{ci}) は全 UE で共通かつ不变の値として $T^{ci} = 10s$ とした。
- UE の送信するデータサイズは十分小さいものとする。つまり、Connected Inactive 状態で発生したデータ送信に関しては、Connected 状態への状態遷移を引き起こさないものとする ($d_h = 0$)。
- UE の通信周期に対する UE 台数の分布は図 10、図 11、図 12 に示す、3 つのデータセットを用意した。それぞれ、ネットワークに存在する UE の台数は 17,400 台、469,200 台、474,600 台である。また、通信周期に対して UE の台数は一様分布である。
- データ送信にかかる時間は UE の通信周期と比べて十分小さいものとし、送信が失敗することはないと仮定する。

この時の結果は、第 5.1 節に示した結果と全く同じである。なぜなら今回の試算では、 d_h の値が変化しても、メモリおよび CPU の負荷には全く影響が出ないためである。これは、今回の試算では UE の状態が Connected 状態および Connected Inactive 状態である時に MME のメモリに与える負荷は同じでありかつ、Connected 状態と Connected Inactive 状態を遷移する際に MME に発生するシグナリングの数は 0 であることから説明できる。

5.3 UE の試算 3

以下のように条件やパラメータを仮定した。

- UE ごとに通信周期は固定であり、途中で変化することはない。
- 最後の送信が終了したあと、Connected 状態の UE が Connected Inactive 状態へ遷移するまでの時間 (T^{ci}) は全 UE で共通かつ不变の値として $T^{ci} = 10s$ とした。
- UE の送信するデータサイズは十分大きいものとする。つまり、データ送信を行うタイミングで必ず Connected 状態に遷移するものとする ($d_h = 1$)。
- UE の通信周期に対する UE 台数の分布は以下の図 14、図 15、図 16 に示す、3 つのデータセットを用意した。それぞれ、ネットワークに存在する UE の台数は 108,000 台、468,000 台、660,000 台である。また、通信周期に対して UE の台数は一様分布である。
- データ送信にかかる時間は UE の通信周期と比べて十分小さいものとし、送信が失敗することはないと仮定する。

MME の CPU 負荷と MME のメモリ負荷の関係を示した図を図 17 に示す。データセット 4 は、 T^i を最小値に設定した場合でも、シグナリング発生数が MME が処理可能な限界値 (1200 回/s) を超えないような UE 台数の分布である。データセット 5 は、 T^i を最大値に設定した場合でも、メ

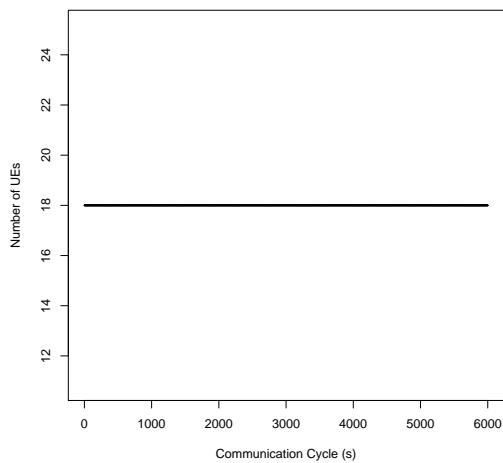


図 14: UE 台数と通信周期 (データセット 4)

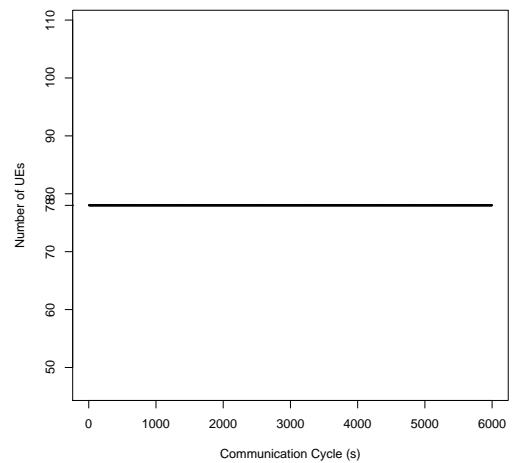


図 15: UE 台数と通信周期 (データセット 5)

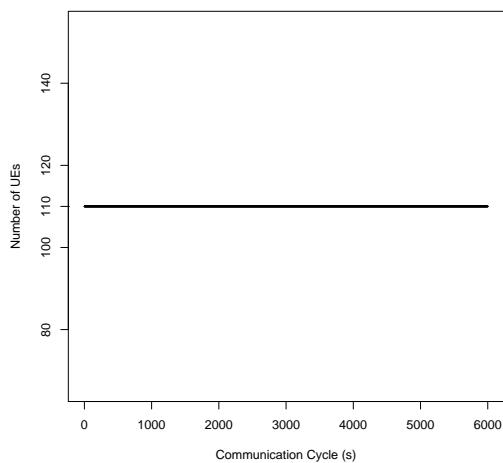


図 16: UE 台数と通信周期 (データセット 6)

モリ消費量が MME の限界値 (1000MB) を超えないような UE 台数の分布である。データセット 6 は、 T^i を適切に設定した場合において、シグナリング発生数およびメモリ消費量が MME の限界値を超えないような UE 台数の分布である。

これらの結果を見ると、第 5.1 節で示した結果と比較して、メモリ負荷に対する CPU 負荷の制約が相対的に緩くなっている。これは、通信周期が長い UE の方が、単位時間あたりに発生するシグナリングが少ないという性質が関係している。本節の評価では、第 5.1 節と比較して通信周期の長い UE が相対的に多く、通信周期の短い UE が相対的に少ないデータセットを用意したため、ネットワーク全体で見た時の CPU 負荷が小さくなっている。そのため、Idle Timer を変化させ CPU 負荷とメモリ負荷を相互にオフロードすることにより収容可能な UE 台数を向上させる効果が、第 5.1 節で示した結果と比較して大きい。実際、Idle Timer を最小に設定した場合 (データセット 4) と Idle Timer を最大に設定した場合 (データセット 5) と比較して、収容可能な UE 台数はそれぞれ 611%、141% に向上している。また、第 5.1 節で述べた結果と同様に、x 軸方向、y 軸方向共に、UE の台数と比例の関係があり、グラフの形状は相似である。

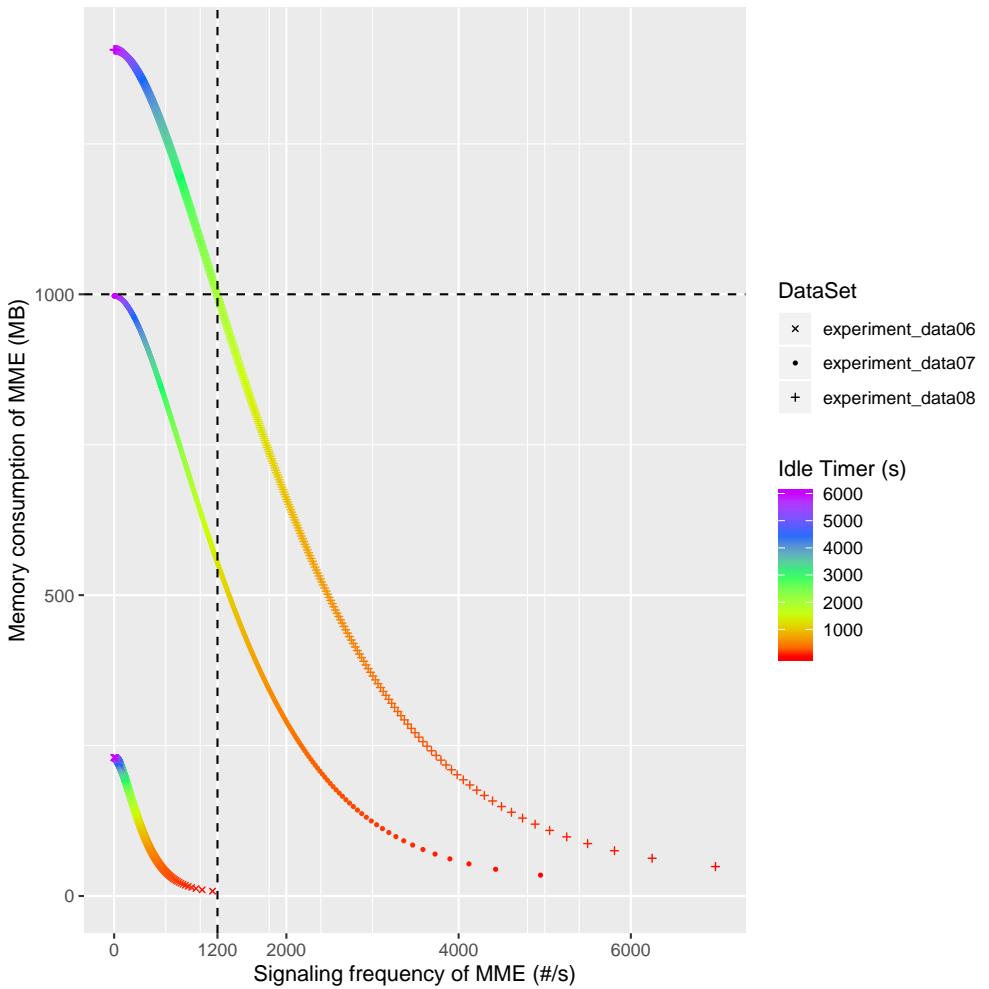


図 17: MME に対して発生する、1s あたりのシグナリング数に対する MME のメモリ負荷の全体像 (データセット 4,5,6)

5.4 UE の試算 4

3GPP の仕様書 [11] を参考にし、以下のように条件やパラメータを仮定した。

- UE ごとに通信周期は固定であり、途中で変化することはない。
- 最後の送信が終了したあと、Connected 状態の UE が Connected Inactive 状態へ遷移するまでの時間 (T^{ci}) は全 UE で共通かつ不変の値として $T^{ci} = 10s$ とした。
- UE の送信するデータサイズは十分大きいものとする。つまり、データ送信を行うタイミングで必ず Connected 状態に遷移するものとする ($d_h = 1$)。
- UE の通信周期に対する UE 台数の分布を以下の図 18、図 19、図 20 に示す。ネットワークに存在する UE の台数はそれぞれ、925,700、469,200、1,178,100 台である。
- データ送信にかかる時間は UE の通信周期と比べて十分小さいものとし、送信が失敗することないと仮定する。

結果を図 21 に示す。データセット 10 は、 T^i を最小値に設定した場合でも、シグナリング発生数が MME が処理可能な限界値 (1200 回/s) を超えないような UE 台数の分布である。データセット 11 は、 T^i を最大値に設定した場合でも、メモリ消費量が MME の限界値 (1000MB) を超えないような UE 台数の分布である。データセット 12 は、 T^i を適切に設定した場合において、シグナリング発生数およびメモリ消費量が MME の限界値を超えないような UE 台数の分布である。この結果より、Idle Timer を適切に設定した場合は、Idle Timer を最小に設定した場合と最大に設定した場合それぞれと比較して、収容可能な UE 台数が 127% および 251% に増加したことが分かる。

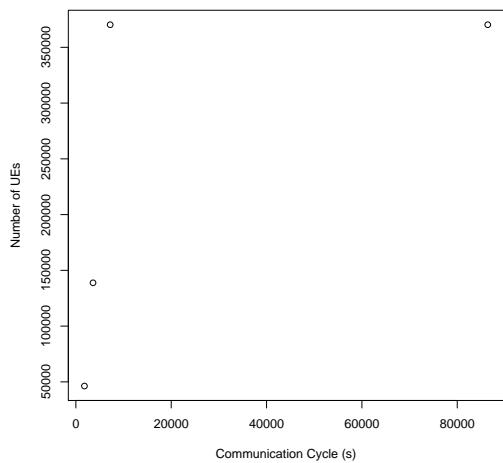


図 18: UE 台数と通信周期 (データセット 10)

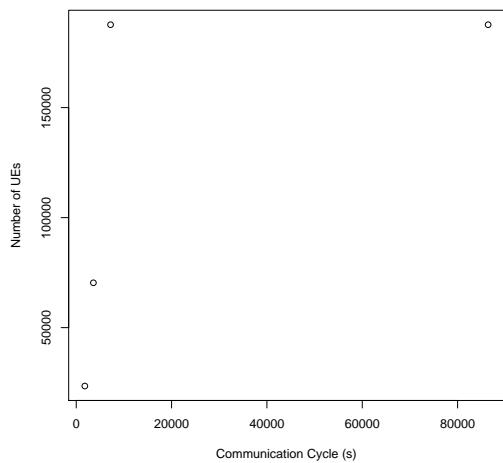


図 19: UE 台数と通信周期 (データセット 11)

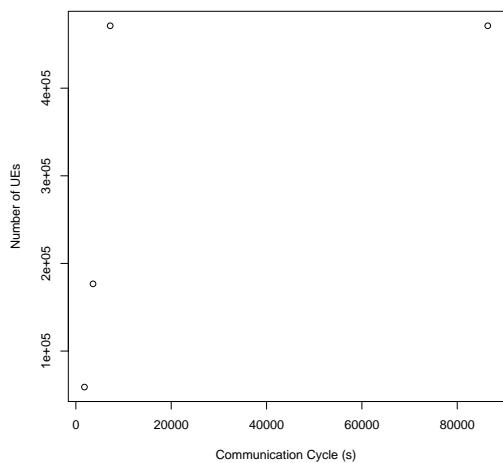


図 20: UE 台数と通信周期 (データセット 12)

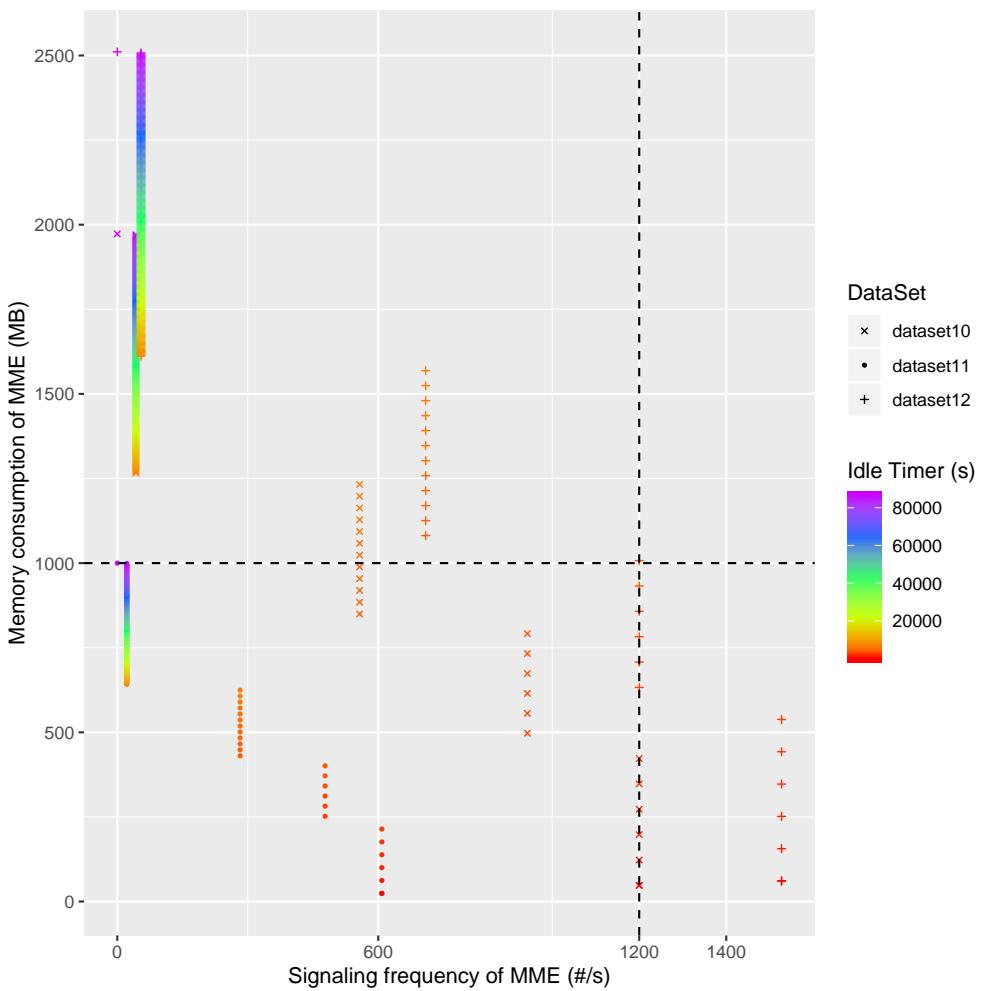


図 21: MME に対して発生する、1sあたりのシグナリング数に対する MME のメモリ負荷 (データセット 10,11,12)

6 今後の予定

- ベイズ最適化の考えを用いた Idle Timer の制御方式を考える。
- UE の通信周期の分布やデータサイズなど、様々なパラメータを変化させた場合の試算を行う。
- IdleTimer の設定方法を決める良いアルゴリズムがないか調査する。
- Connected Inactive 状態において “状態遷移を伴わないデータ送信” が可能なデータ量を調査する。

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. Säily, 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] 3GPP, “Study on architecture enhancements for Cellular Internet of Things (CIoT),” 3rd Generation Partnership Project (3GPP), Technical Report (TR) 23.720, Mar. 2016, version 13.0.0. [Online]. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2894>
- [10] 上野真生, 長谷川 剛, 村田正幸, “多数の M2M/IoT 端末からの集中アクセスを考慮したモバイルコアネットワークの実験評価,” in *Proceedings of 電子情報通信学会技術研究報告 (NS2018-226)*, Mar. 2019.
- [11] 3GPP, “Cellular System Support for Ultra-low Complexity and Low Throughput Internet of Things (CIoT),” 3rd Generation Partnership Project (3GPP), Technical Report (TR) 45.820, Dec. 2015, version 13.1.0. [Online]. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2719>