

ミーティング資料

安達智哉

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

2019 年 10 月 2 日

1 CQ 研究会

論文を執筆しました。

2 Idle タイマの最適化

以前までの評価において、Idle タイマを適切に設定することにより、CPU 負荷およびメモリ使用量の削減が期待できることを示した。それと同時に、Idle タイマの最適な値は、UE の通信周期やその分布に依存して大きく変化することも示した。

一方で、現実的には UE の通信周期やその分布は明らかではない。また、それらは時間とともに変化するものである。そのため、UE の通信周期やその分布が不明であり、動的に変化するような環境においても、Idle タイマを適切な値に設定するような制御方法が必要となる。そこで本章では、Idle タイマの制御方法に関して述べる。

まず、Idle タイマを各 UE に適用するタイミングは、大きく分けて、UE の強制的な状態遷移を引き起こさない方法と引き起こす方法の 2 種類がある。まず、UE の強制的な状態遷移を引き起こさない方法として以下の 2 つが考えられる。

- UE がアタッチしたタイミング
- UE がデータ送信を行うタイミング

次に、UE の強制的な状態遷移を引き起こす方法として以下の 2 つが考えられる。

- UE の動作や状態に依存しない、定期的なタイミング
- 任意のタイミング

それぞれには、メリットデメリットが考えられる。

UE の強制的な状態遷移を引き起こさない方法の場合、更新タイミングが UE の動作に依存するため、新しい Idle タイマを UE に設定するまでにかかる時間が UE ごとに異なるという問題がある。これにより、異なる Idle タイマを持つ UE が同時に存在するような状況が発生するため、Idle タイマの制御が複雑になると考えられる。また、Idle タイマを変更した後、CPU 負荷とメモリ使用量に変化が現れるまでに遅延が発生するため、MME のリソースの制御が難しくなると考えられる。しかし、アタッチやデータ送信など、UE と MME が通信するタイミングで Idle タイマの更新を行うため、追加のシグナリングや状態遷移が少なく、オーバーヘッドが小さい。

一方、更新タイミングがUEの動作や状態に依存しない場合、Idle タイマを更新するためにUEの状態を変化させる必要がある場合があり、オーバーヘッドが大きくなるという問題がある。具体的には、MMEと通信できない状態にあるUEのIdle タイマを変化させるためには、UEを一度接続状態へと遷移させる必要がある。この際に、状態遷移に伴うシグナリング処理が発生するため、MMEのCPU負荷が増加する。しかし、Idle タイマをUEに反映させるまでにかかる時間はUEに依存しないため、全UEのIdle タイマの値を一定期間内で更新できる。さらに、設定するIdle タイマの値を0にすることで、MMEは任意のタイミングで任意のUEを強制的にアイドル状態へ遷移させることができる。これにより、UEの強制的な状態遷移を引き起こさない方法の場合と比較してMMEのリソース制御が容易になると考えられる。

2.1 Idle タイマの制御方法 (UEの強制的な状態遷移を引き起こさない方法)

本節では、UEの強制的な状態変化を引き起こさないことを前提にする。つまり、Idle タイマが切れていないUEを強制的にIdle状態へ遷移させることはしないとする。また、Idle タイマの更新は、UEがデータ送信を行うタイミングで実行するものとする。MMEはUEを収容するために使用されているCPUおよびメモリリソース量を観測できるものとする。つまり、UEの収容とは無関係な処理によって発生する負荷を取り除いたCPU負荷およびメモリ使用量を知ることができる。MMEは現在収容されているUE台数を観測できるものとする。

突発的な負荷の増加に対応するという観点から、現在収容しているUEに加え、最も多くのUEを収容できるようなIdle タイマの値が最適と考える。具体的には、現在収容しているUEと同じ通信周期を持つUEがネットワークに参加すると仮定し、最も多くのUEを追加で収容できるIdle タイマの値を最適と定義する。また、CPUおよびメモリのどちらも過負荷状態でないことは、UEを収容可能であることの必要十分条件であるとする。

まず、UE一台あたりが各リソースに与える負荷の平均を推定する。現在収容しているUE台数を N_{UE} とする。UE台数が N_{UE} 、Idle タイマが T^i の時に観測される、CPU負荷およびメモリ使用量をそれぞれ $C_{N_{UE}}(T^i)$ 、 $M_{N_{UE}}(T^i)$ とする。この時、UE一台あたりが与えるCPU負荷およびメモリ使用量の平均($C_1(T^i)$ 、 $M_1(T^i)$)は以下の式(1)、(2)で表せる。

$$C_1(T^i) = \frac{C_{N_{UE}}(T^i)}{N_{UE}} \quad (1)$$

$$M_1(T^i) = \frac{M_{N_{UE}}(T^i)}{N_{UE}} \quad (2)$$

Idle タイマを T^i とした時に、 N_{UE} 台のUEを収容している状態から追加で収容可能なUE台数を $N_{UE}^{add}(T^i)$ とする。 $N_{UE}^{add}(T^i)$ は、 $C_1(T^i)$ 、 $M_1(T^i)$ 、 $C_{N_{UE}}(T^i)$ 、 $M_{N_{UE}}(T^i)$ 、 C^{max} および M^{max} を用いて、以下の式(3)で表せる。ここで、 C^{max} 、 M^{max} はそれぞれシグナリング処理およびUEのセッション情報を保持するために使用可能なCPUリソース量およびメモリリソース量である。

$$N_{UE}^{add}(T^i) = \min\left\{\left\lfloor \frac{C^{max} - C_{N_{UE}}(T^i)}{C_1(T^i)} \right\rfloor, \left\lfloor \frac{M^{max} - M_{N_{UE}}(T^i)}{M_1(T^i)} \right\rfloor\right\} \quad (3)$$

Idle タイマを制御する上での目的関数を以下の式(4)に示す。

$$\text{maximize : } N_{UE}^{add}(T^i) \quad (4)$$

$N_{UE}^{add}(T^i)$ を最大化するように、Idle タイマを制御すれば、Idle タイマを最適な値に近づけることができる。具体的には、各リソースの使用量を観測して、 $N_{UE}^{add}(T^i)$ を大きくする向きにIdle タイマを変化させる。このステップを複数回繰り返すことにより、Idle タイマを制御する。

この時、1ステップごとの Idle タイマの変化量を考える必要がある。この値を小さく設定すると、最適値に到達するまでに大きな時間がかかってしまう場合がある。逆に Idle タイマの変化量を大きく設定すると、Idle タイマが発振する可能性もあり、制御が不安定になる。また、UE の通信周期によって、Idle タイマが変化した時に各リソースの負荷の変化量が異なる点も考慮する必要がある。

つまり、ネットワークの変化に短い時間スケールで対応しつつ、安定した制御を実現するためには、ネットワークの環境に応じて Idle タイマの変化量を制御する仕組みが必要である。この仕組みには PID 制御が利用できると考えている。Idle タイマの設定を入力、各リソースの使用率を出力をして捉えることで、PID 制御により、Idle タイマの変化量を調整しつつ、最適値に近づけることができる。

まず、PID 制御における出力値 $y(t)$ および目標値 $r(t)$ を設定する。以前の評価より、UE 台数を固定した時、CPU 負荷は Idle タイマの値に対して広義単調減少でありかつ、メモリ使用量は Idle タイマの値に対して広義単調増加であることがわかっている。このことから、 $C_1(T^i)$ および $C_{N_{UE}}(T^i)$ は T^i に対して広義単調減少であることがわかる。同様に $M_1(T^i)$ および $M_{N_{UE}}(T^i)$ は T^i に対して広義単調増加であることがわかる。以上を踏まえて式 (3) を確認すると、 $\lfloor \frac{C^{\max} - C_{N_{UE}}(T^i)}{C_1(T^i)} \rfloor$ は広義単調増加でありかつ、 $\lfloor \frac{M^{\max} - M_{N_{UE}}(T^i)}{M_1(T^i)} \rfloor$ は広義単調減少であることがわかる。ここで、 $\lfloor \frac{C^{\max} - C_{N_{UE}}(T^i)}{C_1(T^i)} \rfloor$ と $\lfloor \frac{M^{\max} - M_{N_{UE}}(T^i)}{M_1(T^i)} \rfloor$ の差分を最小化するような T^i の集合を \mathbf{T}^i とする。また、 $N_{UE}^{\text{add}}(T^i)$ を最大化するような T^i の集合を $\mathbf{T}_{\text{optimal}}^i$ とする。すると、 $\lfloor \frac{C^{\max} - C_{N_{UE}}(T^i)}{C_1(T^i)} \rfloor$ は広義単調増加でありかつ、 $\lfloor \frac{M^{\max} - M_{N_{UE}}(T^i)}{M_1(T^i)} \rfloor$ は広義単調減少であることを考慮すると、 $T^i \in \mathbf{T}^i$ であることは $T^i \in \mathbf{T}_{\text{optimal}}^i$ であるための十分条件になる。

以上の議論のイメージを図 1、図 2 および図 3 に示す。図 1 は UE 台数が 500,000 台、UE ごとの通信周期は 10 s から 6,000 s の範囲で一様分布とした時の、Idle タイマと各リソース負荷の関係を示したものである。図 2 は図 1 と同じ UE を収容した時の、Idle タイマと $\lfloor \frac{C^{\max} - C_{N_{UE}}(T^i)}{C_1(T^i)} \rfloor$ と $\lfloor \frac{M^{\max} - M_{N_{UE}}(T^i)}{M_1(T^i)} \rfloor$ との関係を示している。また、図 3 は図 1 と同じ UE を収容した時の、Idle タイマと $N_{UE}^{\text{add}}(T^i)$ との関係を示している。図 3 を見ると、 $N_{UE}^{\text{add}}(T^i)$ を最大化する Idle タイマの値と $\lfloor \frac{C^{\max} - C_{N_{UE}}(T^i)}{C_1(T^i)} \rfloor$ と $\lfloor \frac{M^{\max} - M_{N_{UE}}(T^i)}{M_1(T^i)} \rfloor$ の差分を最小化する Idle タイマの値が一致していることが確認できる。

このことを踏まえ、PID 制御における出力値 $y(t)$ および目標値 $r(t)$ を以下の式 (5)、(6) のように定義する。 t は時刻を表す変数である。

$$y(t) = \lfloor \frac{C^{\max} - C_{N_{UE}}(T^i)}{C_1(T^i)} \rfloor - \lfloor \frac{M^{\max} - M_{N_{UE}}(T^i)}{M_1(T^i)} \rfloor \quad (5)$$

$$r(t) = 0 \quad (6)$$

時刻 t における $y(t)$ と $r(t)$ の差を $e(t)$ として以下の式 (7) ように定義すると、PID 制御における操作量 ($u(t)$) は以下の式 (8) で表せる。

$$e(t) = r(t) - y(t) \quad (7)$$

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau) d\tau + K_d \cdot \frac{de(t)}{dt} \quad (8)$$

ここで、 K_p 、 K_i および K_d はそれぞれ、比例ゲイン、積分ゲインおよび微分ゲインと呼ばれる定数である。これらの定数は、 $e(t)$ およびその積分値、微分値が $u(t)$ にどの程度寄与するのかを決定する。

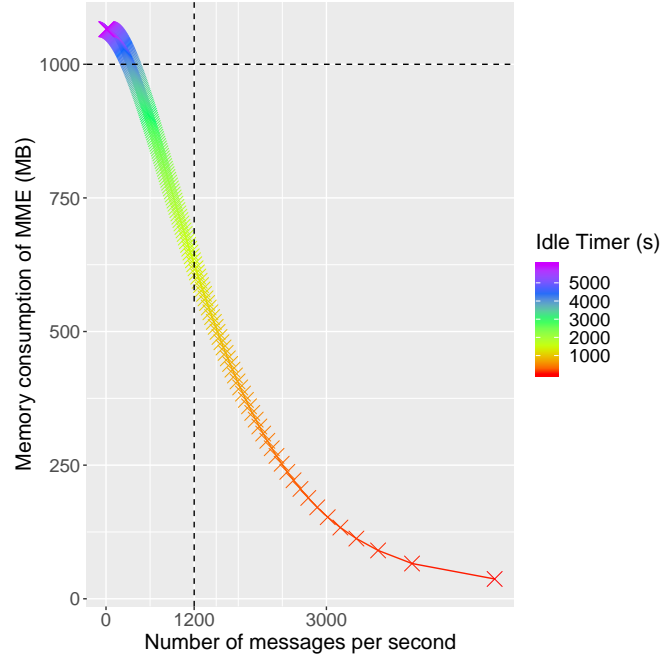


図 1: Idle タイマに対する，メッセージ処理頻度とメモリ使用量の関係

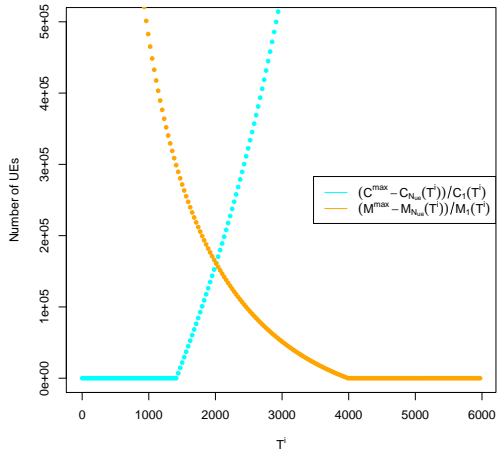


図 2: Idle タイマと $\lfloor \frac{C^{\max} - C_{NUE}(T^i)}{C_1(T^i)} \rfloor$ と $\lfloor \frac{M^{\max} - M_{NUE}(T^i)}{M_1(T^i)} \rfloor$ の関係

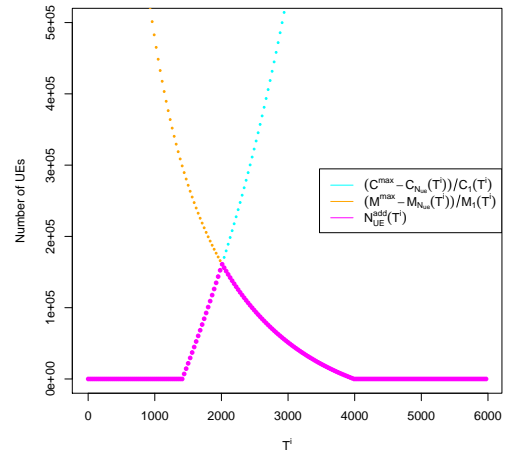


図 3: Idle タイマと $N^{\text{add}}_{UE}(T^i)$ の関係

PID 制御を機能させるためには、これら 3 つの定数を適切に設定する必要がある。しかし、一般的に、これらの定数の最適値を数学的に導出することは困難である。そのため、試行錯誤を繰り返しながら経験的にパラメータ調整を行う必要があると言われている。しかし一方で、パラメータの設定方法に関しては、いくつか有名な手順が存在するため、それらに従って設定することもできる。以下に代表的なパラメータの設定手順を示す。

- ジーグラ・ニコルス法
- CHR 法

2.2 Idle タイマの制御方法 (UE の強制的な状態遷移を引き起こす方法)

本節では、UE の強制的な状態変化を引き起こすことを前提にする。つまり、Idle タイマが切れていない UE を強制的に Idle 状態へ遷移させることが可能である。また、MME は CPU およびメモリリソースの使用量および、現在収容している UE 台数を観測できるものとする。

第 2.1 節と同様に、現在収容している UE に加え、最も多くの UE を収容できるような Idle タイマの値が最適と考える。

本節は第 2.1 節と異なり、UE を強制的に Idle 状態へ遷移させることが可能であるため、短期間にメモリ使用量を削減することが可能である。そこで、常にメモリに使用量が大きい状態にしておき、UE の増加に応じて一部の UE を Idle 状態へ遷移させるような制御が良いと考える。例えば、Idle タイマとシグナリング頻度およびメモリ使用量の関係が図 1 のようになる場合、メモリ使用量がメモリ容量を超えないギリギリの値 (約 4,000 s) を Idle タイマに設定する。そして、UE 台数が増加し、メモリ使用量が増加した場合には、Idle タイマを小さくする。具体的には、メモリの過負荷状態が解消されるまで、最後のデータ送信時刻が古い UE から順に Idle 状態へ落としていく。そして、メモリ負荷が解消された時に、Connected Inactive 状態を維持している UE のなかで、最後のデータ送信の時刻が最も古い UE を参考にし、新しい Idle タイマを設定する。

3 今後の予定

- PID 制御に関する学習
- 発表スライドの作成: ~10/16