

# 進捗報告資料

安達智哉

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

2019年12月19日

## 1 Idle タイマの制御方法

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とする。MME は現在収容している 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、収容可能な UE 台数が最大となる Idle タイマの値を最適と定義する。また、CPU より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、現在収容している UE 台数を  $N_{\text{UE}}$  とする。UE 台数が  $N_{\text{UE}}$ 、Idle タイマが  $T$  の時に観測される、CPU およびメモリの使用量をそれぞれ  $C_{N_{\text{UE}}}(T)$ 、 $M_{N_{\text{UE}}}(T)$  とする。

Idle タイマを  $T$  とした時に、収容可能な UE の総数を  $N_{\text{UE}}^{\text{capa}}(T)$  とすると  $N_{\text{UE}}^{\text{capa}}(T)$  は、 $C_{N_{\text{UE}}}(T)$ 、 $M_{N_{\text{UE}}}(T)$ 、 $C^{\max}$  および  $M^{\max}$  を用いて、以下の式 (1) 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。

$$N_{\text{UE}}^{\text{capa}}(T) = \lfloor N_{\text{UE}} \cdot \min\left\{\frac{C^{\max}}{C_{N_{\text{UE}}}(T)}, \frac{M^{\max}}{M_{N_{\text{UE}}}(T)}\right\} \rfloor \quad (1)$$

Idle タイマを制御するまでの目的関数を以下の式 (2) に示す。

$$\text{maximize : } N_{\text{UE}}^{\text{capa}}(T) \quad (2)$$

$N_{\text{UE}}^{\text{capa}}(T)$  を最大化する Idle タイマの値が明らかである場合は、その値を Idle タイマに設定すれば良い。しかし一般的に、UE の台数や通信周期は未知であり時間的に変動するため、 $N_{\text{UE}}^{\text{capa}}(T)$  を最大化する Idle タイマの値をすることは難しい。そのような場合は、 $N_{\text{UE}}^{\text{capa}}(T)$  を最大化するように、Idle タイマを適応的に制御する必要がある。具体的には、各リソースの使用量を観測して、 $N_{\text{UE}}^{\text{capa}}(T)$  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、最適な値に到達するまでに大きな時間がかかる場合がある。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。このような制御には様々な手法が考えられるが、本報告では動作がシンプルであり、汎用性が高い PID 制御を用いる。

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、CPU 使用量は 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、式 (1) を確認すると、 $\frac{C^{\max}}{C_{N_{\text{UE}}}(T)}$  は広義単調増加でありかつ、 $\frac{M^{\max}}{M_{N_{\text{UE}}}(T)}$  は広義単調減少であることがわかる。ここで、以下の式 (3) を満たすような  $T$  の集合を  $\mathbf{T}$  とし、 $N_{\text{UE}}^{\text{capa}}(T)$  を最大化するような  $T$  の集合を  $\mathbf{T}_{\text{optimal}}$  とすると、 $T \in \mathbf{T}$  であることは  $T \in \mathbf{T}_{\text{optimal}}$  であるための十分条件になる。

$$\frac{C^{\max}}{C_{N_{\text{UE}}}(T)} = \frac{M^{\max}}{M_{N_{\text{UE}}}(T)} \quad (3)$$

以上の議論のイメージを図 1、図 2a および図 2b 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 2a は図 1 と同じ UE を収容した時の、Idle タイマと  $N_{\text{UE}} \cdot \frac{C^{\max}}{C_{N_{\text{UE}}}(T)}$  と  $N_{\text{UE}} \cdot \frac{M^{\max}}{M_{N_{\text{UE}}}(T)}$  との関係を示している。また、図 2b は図 1 と同じ UE を収容した時の、Idle タイマと  $N_{\text{UE}}^{\text{capa}}(T)$  との関係を示している。図 2b を見ると、 $N_{\text{UE}}^{\text{capa}}(T)$  を最大化する Idle タイマの値と  $\frac{C^{\max}}{C_{N_{\text{UE}}}(T)} = \frac{M^{\max}}{M_{N_{\text{UE}}}(T)}$  となる時の Idle タイマの値が一致していることが確認できる。

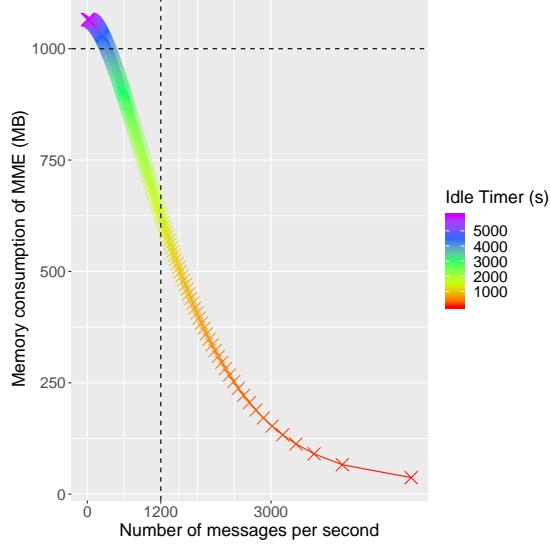


図 1: Idle タイマに対する、メッセージ処理頻度とメモリ使用量の関係

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 (4)、(5) のように定義する。 $t$  は時刻を表す変数である。

$$y(t) = \frac{C_{N_{\text{UE}}}(T)}{C^{\max}} - \frac{M_{N_{\text{UE}}}(T)}{M^{\max}} \quad (4)$$

$$r(t) = 0 \quad (5)$$

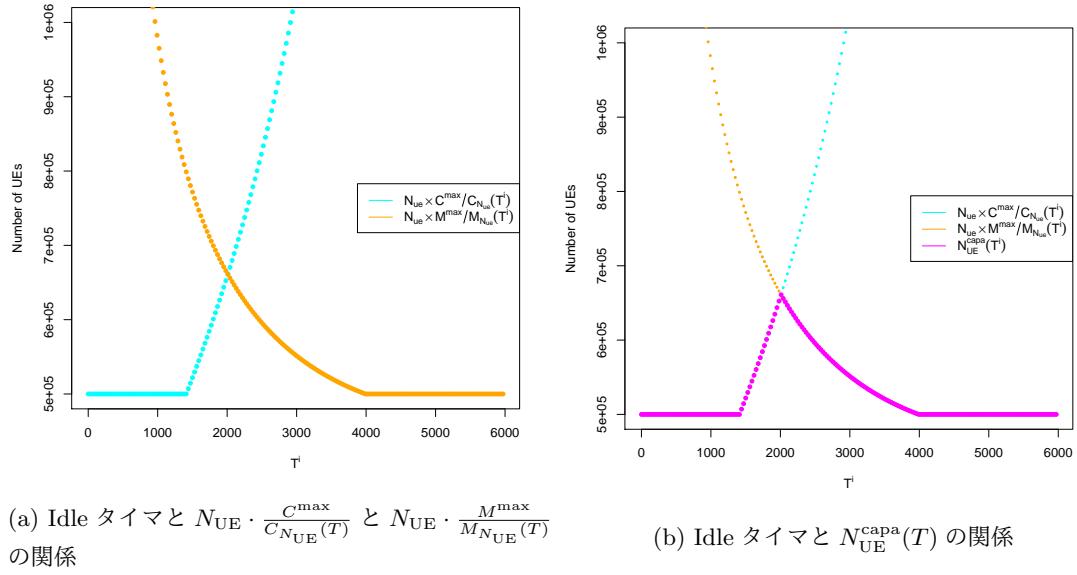


図 2

時刻  $t$  における  $y(t)$  と  $r(t)$  の差を  $e(t)$  として以下の式 (6) 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 (7) で表せる。

$$e(t) = r(t) - y(t) \quad (6)$$

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau) d\tau + K_d \cdot \frac{de(t)}{dt} \quad (7)$$

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。

## 2 ジーグラニコルスの限界感度法

PID 制御においては、比例ゲイン ( $K_p$ )、積分ゲイン ( $K_i$ )、微分ゲイン ( $K_d$ ) と呼ばれる 3 つの定数を設定する必要がある。これらの定数を“ジーグラ・ニコルスの限界感度法”と呼ばれる手順に基づき設定した。

ジーグラ・ニコルスの限界感度法に基づくゲインの求め方を以下に示す。

**ステップ 1** 積分ゲイン ( $K_i$ ) および微分ゲイン ( $K_d$ ) を 0 にして調節器が比例動作だけを行うようにする。

**ステップ 2** 比例ゲイン ( $K_p$ ) を 0 から徐々に大きくしていく、制御量が安定限界に達して一定振幅振動を持続するようになった時に  $K_p$  の増加を止める。

**ステップ 3** ステップ 2 の時の比例ゲインを限界感度 ( $K_u$ )、振動周期を限界周期 ( $P_u$ ) とし、これらの値から各ゲインを以下の表 1 のように求める。

表 1: ジーグラ・ニコルスの限界感度法

制御の種類	$K_p$	$K_i$	$K_d$	$T_i$	$T_d$
P	$0.5K_u$	0	0	-	-
PI	$0.45K_u$	$K_p/0.83P_u$	0	$0.83P_u$	-
PID	$0.6K_u$	$K_p/0.5P_u$	$K_p \cdot 0.125P_u$	$0.5P_u$	$0.125P_u$

ステップ 4 必要に応じてステップ 3 で求めた各ゲインの値を調整する。

以下のシナリオにおいて、ジーグラニコルスの限界感度法を用いて、PID 制御のゲインを求めた。UE 台数は 648,000 台であり、UE の持つ通信周期は 1 day, 2 hours, 1 hour, 30 minutes のいずれかである。それぞれの通信周期を持つ UE の割合は表 3 の通りである。また、各パラメータを表 2 に示す。

表 2: パラメータ設定

Parameter	Numerical setting
$T^{\text{ci}}$	10 s
$s_{\text{MME}}^{\text{c} \rightarrow \text{c}}$	0 messages
$s_{\text{MME}}^{\text{ci} \rightarrow \text{ci}}$	0 messages
$s_{\text{MME}}^{\text{c} \rightarrow \text{ci}}$	0 messages
$s_{\text{MME}}^{\text{ci} \rightarrow \text{c}}$	0 messages
$s_{\text{MME}}^{\text{ci} \rightarrow \text{i}}$	5 messages
$s_{\text{MME}}^{\text{i} \rightarrow \text{c}}$	5 messages
$m_{\text{MME}}^{\text{c}}$	17878 bits
$m_{\text{MME}}^{\text{ci}}$	17878 bits
$m_{\text{MME}}^{\text{i}}$	408 bits
$C^{\max}$	1200 messages/s
$M^{\max}$	1,000 MB
$d_h$	1

表 3: UE の通信周期の分布

	通信周期			
	1 day	2 hours	1 hour	30 minutes
UE 台数の割合	40%	40%	15%	5%

まず、限界感度および限界周期を求めるために、 $K_i$  および  $K_d$  を 0 にして、 $K_p$  を 0 から徐々に大きくしていった。その結果を図 3、図 4、図 5 および図 6 に示す。これらの図は  $K_p = 1$ 、 $K_p = 2$ 、 $K_p = 2.875$ 、 $K_p = 3$  の場合の結果を示している。これらの図を見ると、 $K_p = 2.875$  以下の場合は制御が収束するが、 $K_p = 3$  の場合は制御が収束せず、一定振幅振動していることがわかる。このことから、限界感度 ( $K_u$ ) を 3 とする。また、限界周期 ( $P_u$ ) は図 6 より、8000 s とする。

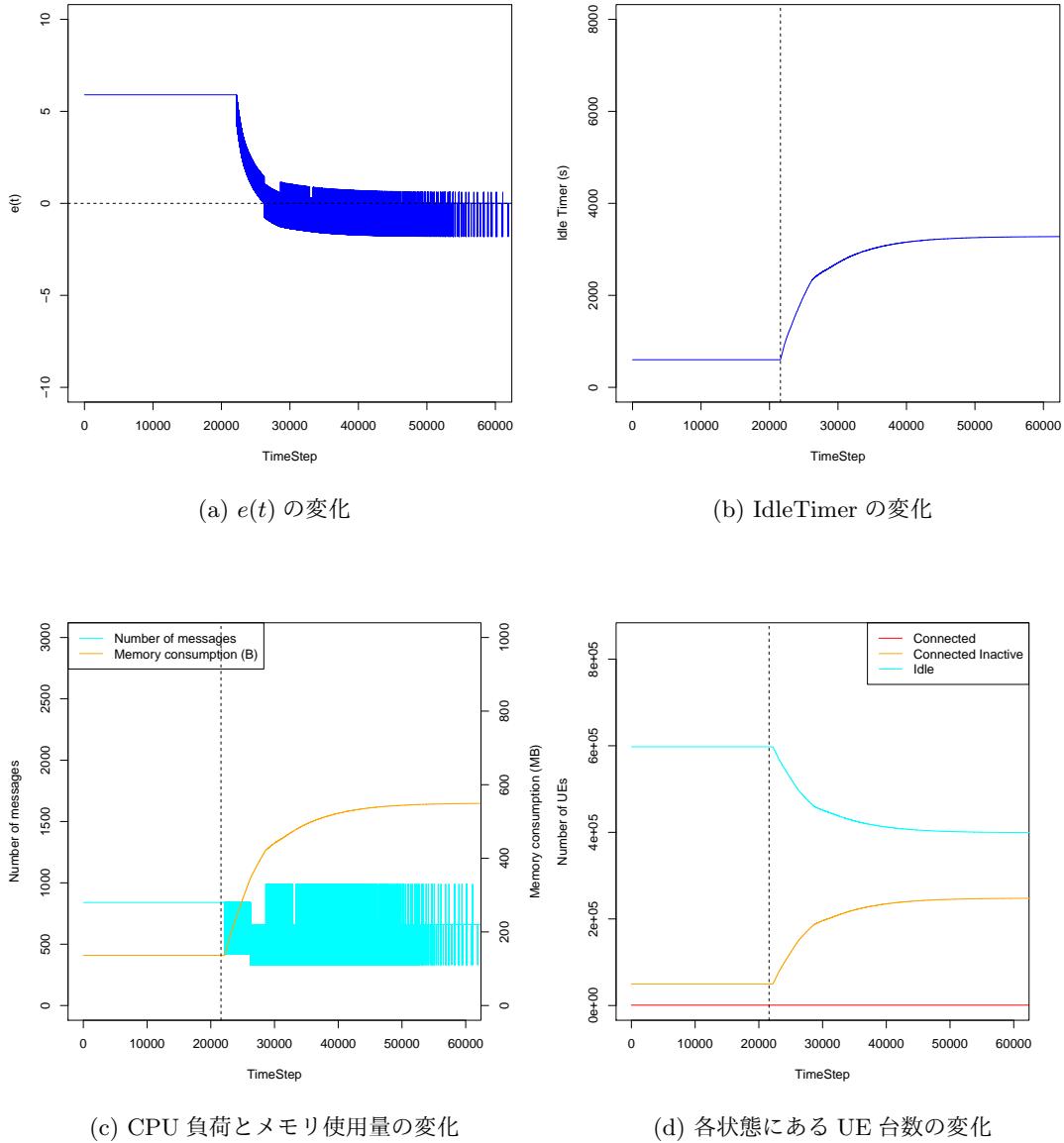


図 3: 理想 PID( $K_p = 1, K_i = 0, K_d = 0$ )

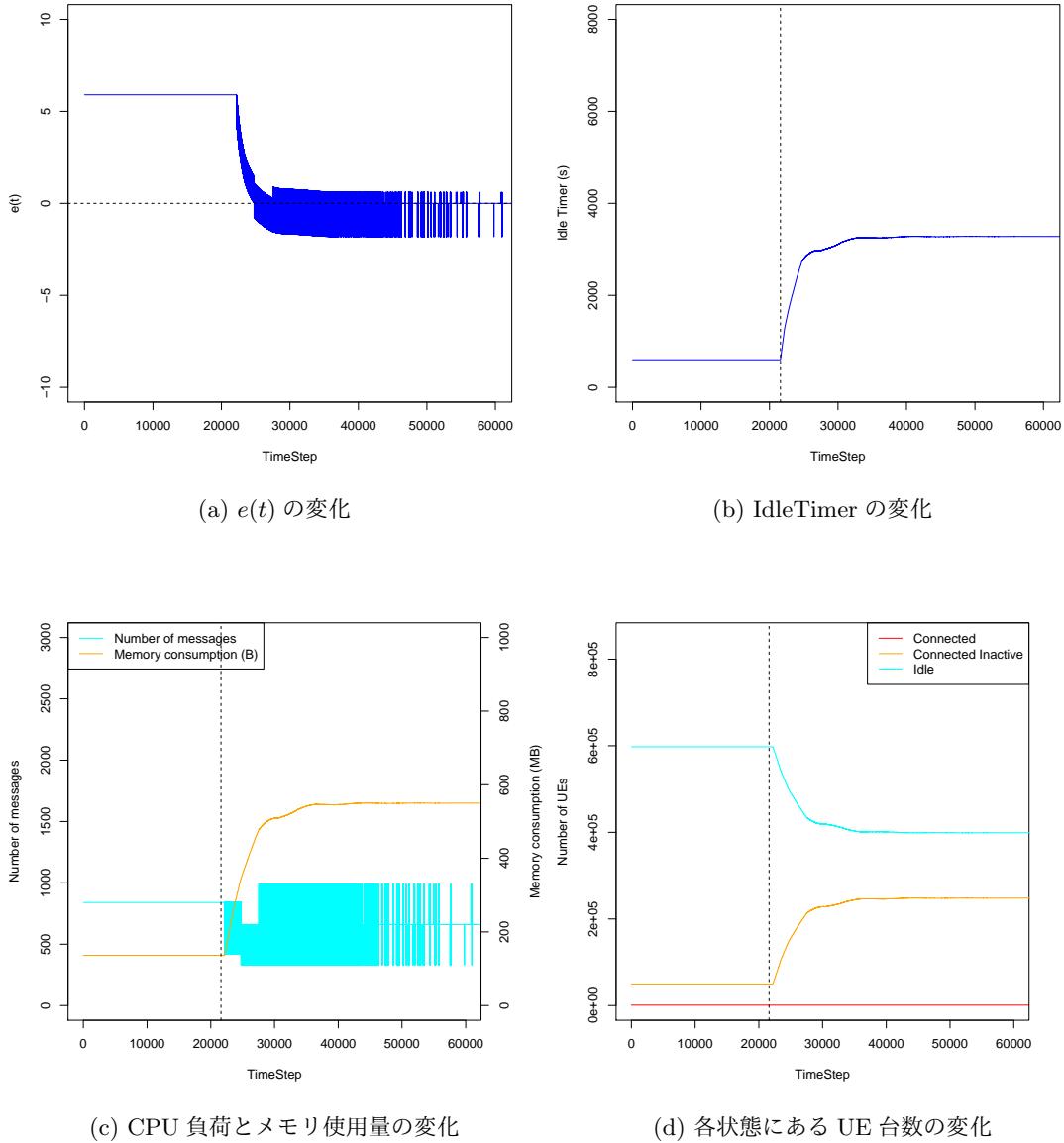


図 4: 理想 PID( $K_p = 2, K_i = 0, K_d = 0$ )

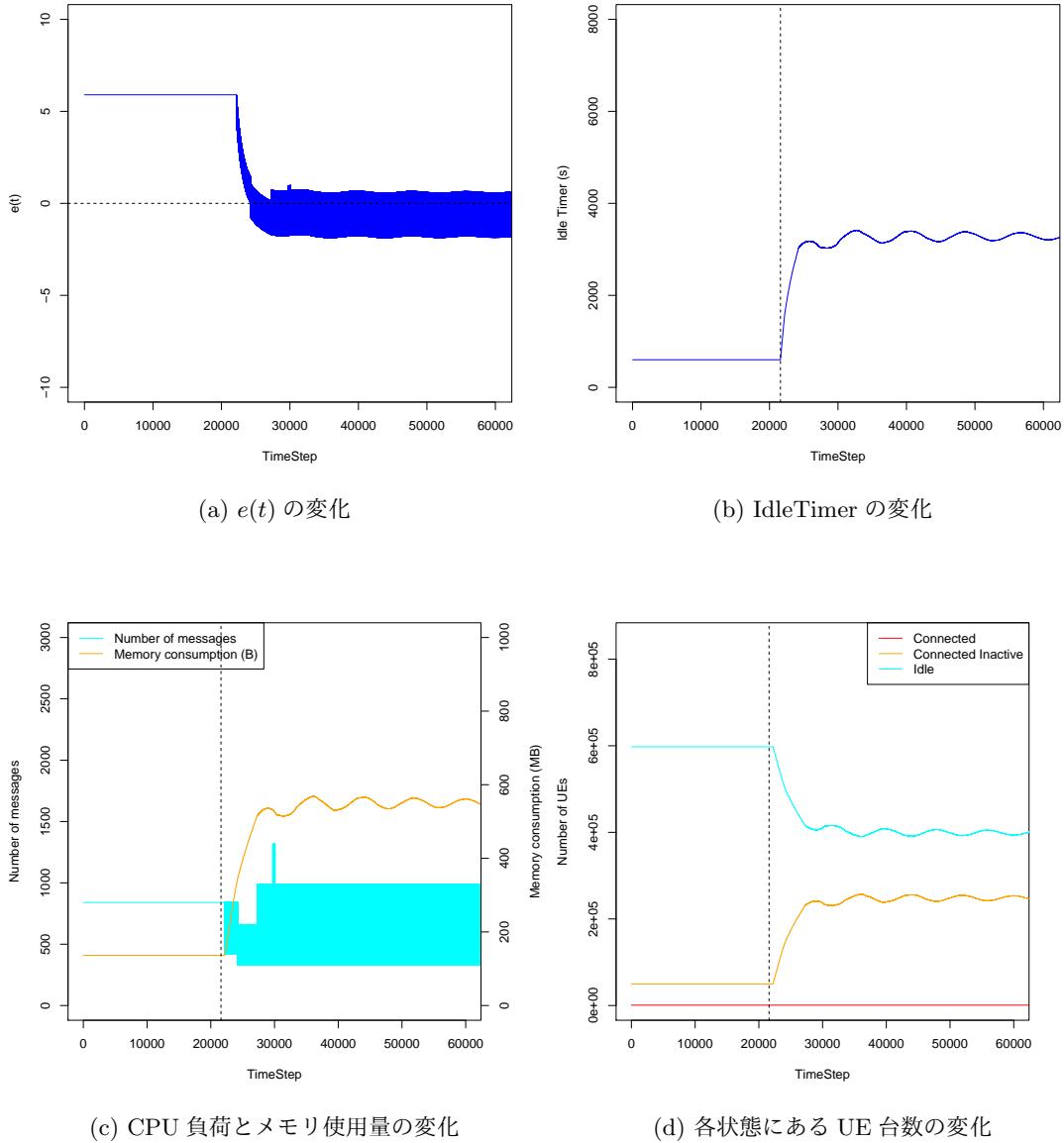


図 5: 理想 PID( $K_p = 2.875, K_i = 0, K_d = 0$ )

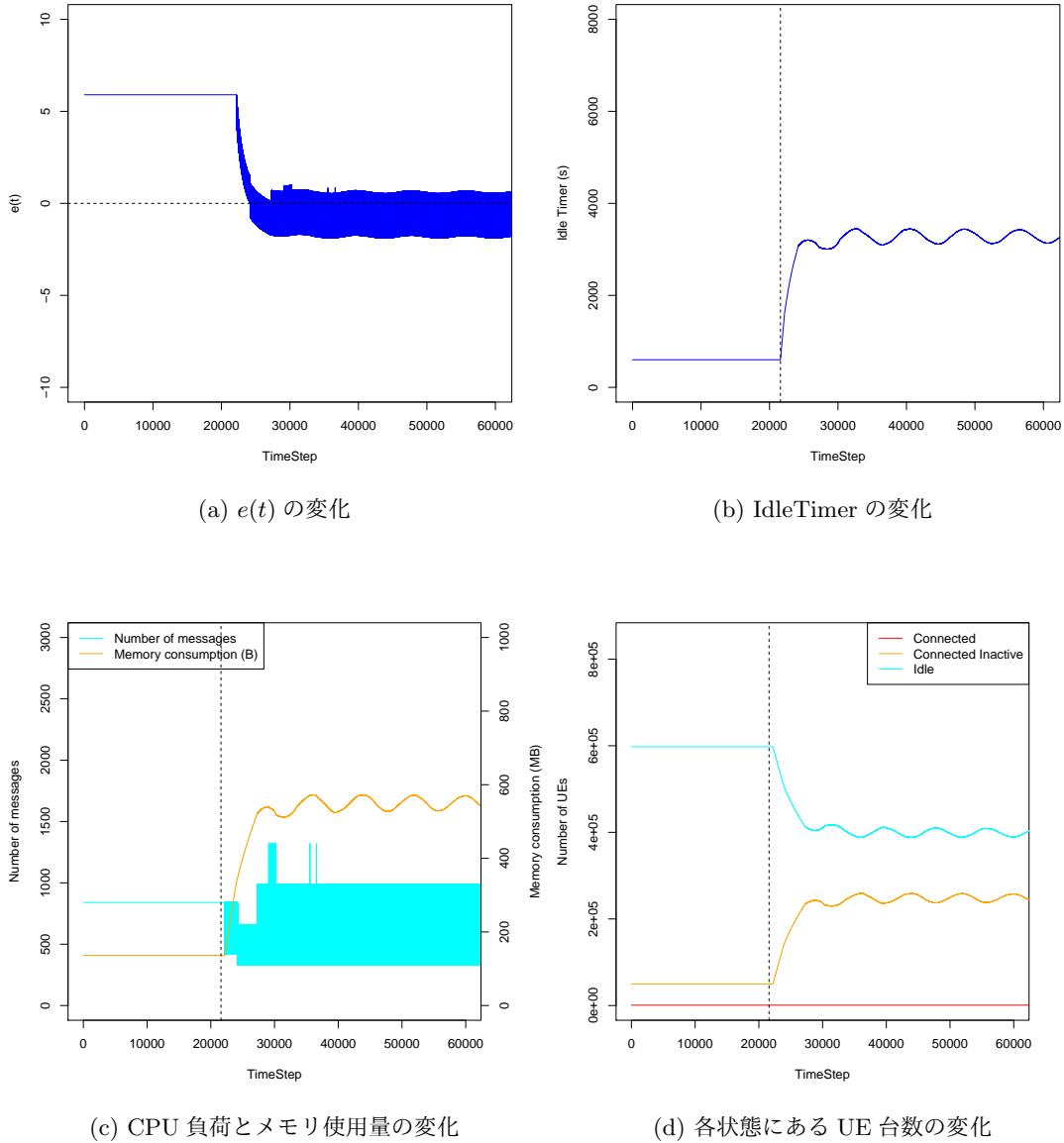


図 6: 理想 PID( $K_p = 3, K_i = 0, K_d = 0$ )

表 4: ジーグラ・ニコルスの限界感度法に基づく設定

制御の種類	$K_p$	$K_i$	$K_d$	$T_i$	$T_d$
P	1.5	0	0	-	-
PI	1.35	0.000203	0	6640	-
PID	1.8	0.00045	1800	4000	1000

以上の結果より、各ゲインの値は以下の表 4 のように求まる。

表4に示したP制御の値を $K_p$ 、 $K_i$ および $K_d$ にそれぞれ設定した場合の評価結果を図7に示す。

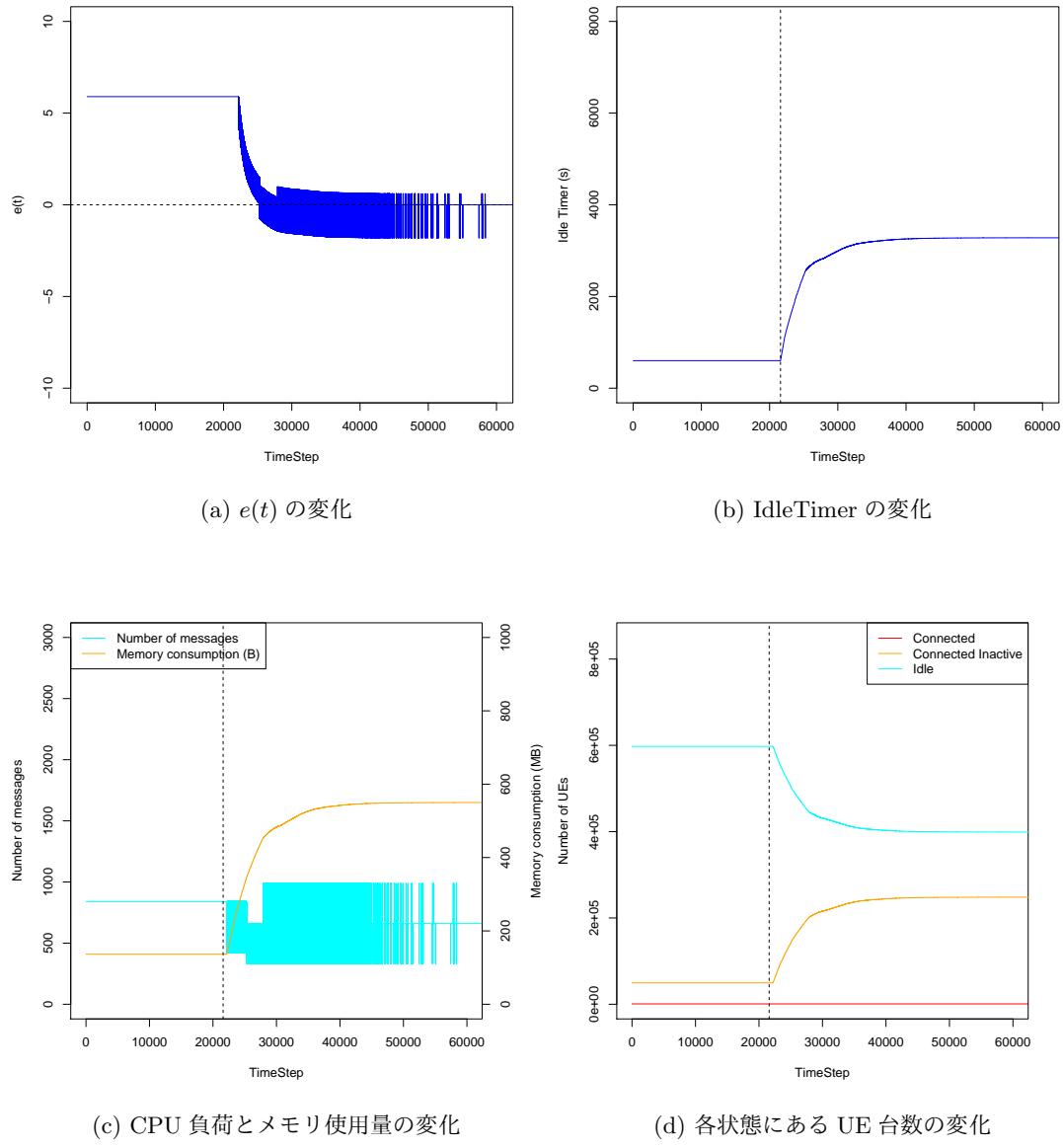


図 7: 理想 PID( $K_p = 1.5$ 、 $K_i = 0$ 、 $K_d = 0$ )

表 4 に示した PI 制御の値を  $K_p$ 、 $K_i$  および  $K_d$  にそれぞれ設定した場合の評価結果を図 8 に示す。

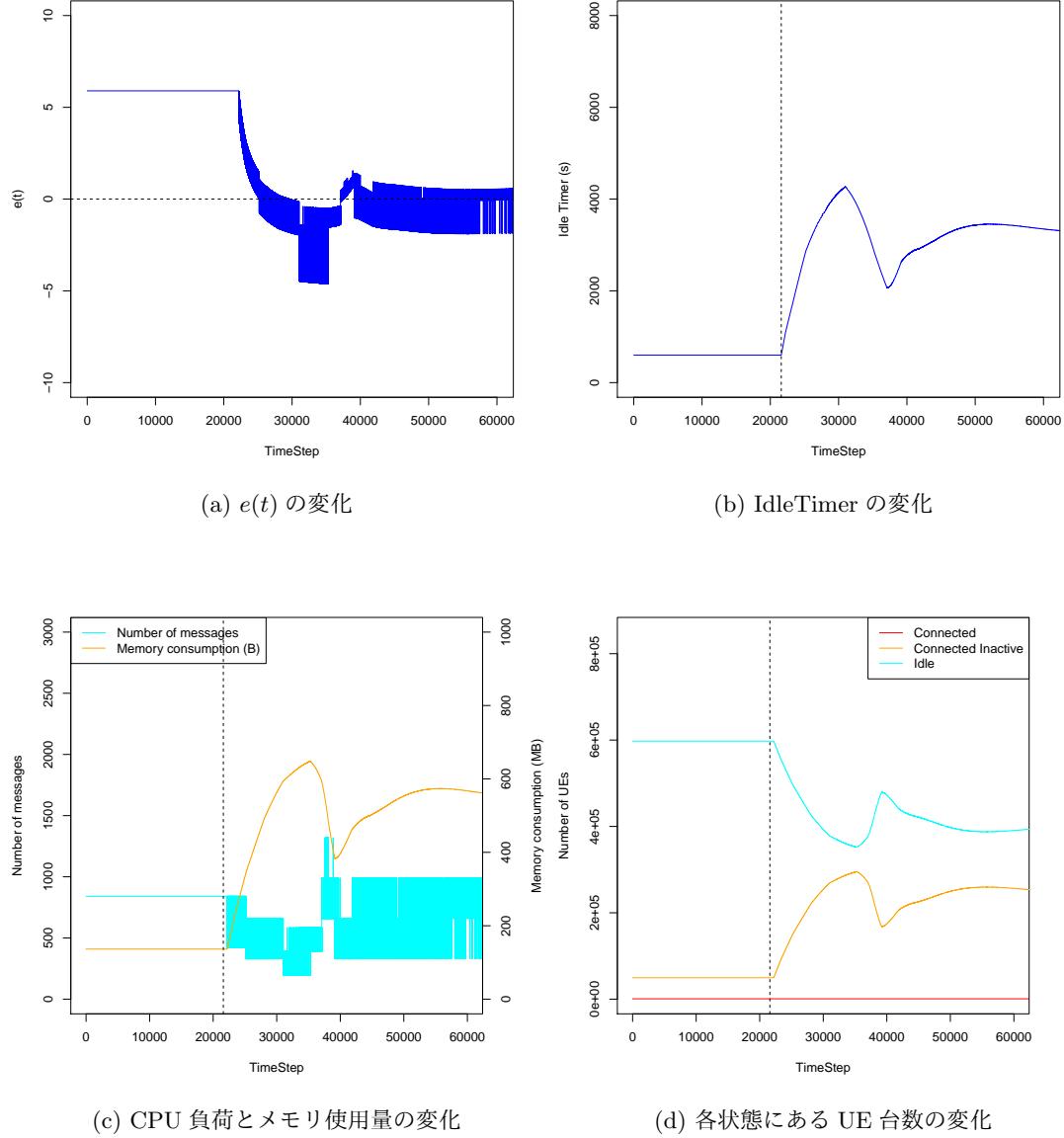


図 8: 理想 PID( $K_p = 1.35$ 、 $K_i = 0.000203$ 、 $K_d = 0$ )

表4に示したPID制御の値を $K_p$ 、 $K_i$ および $K_d$ にそれぞれ設定した場合の評価結果を図9に示す。

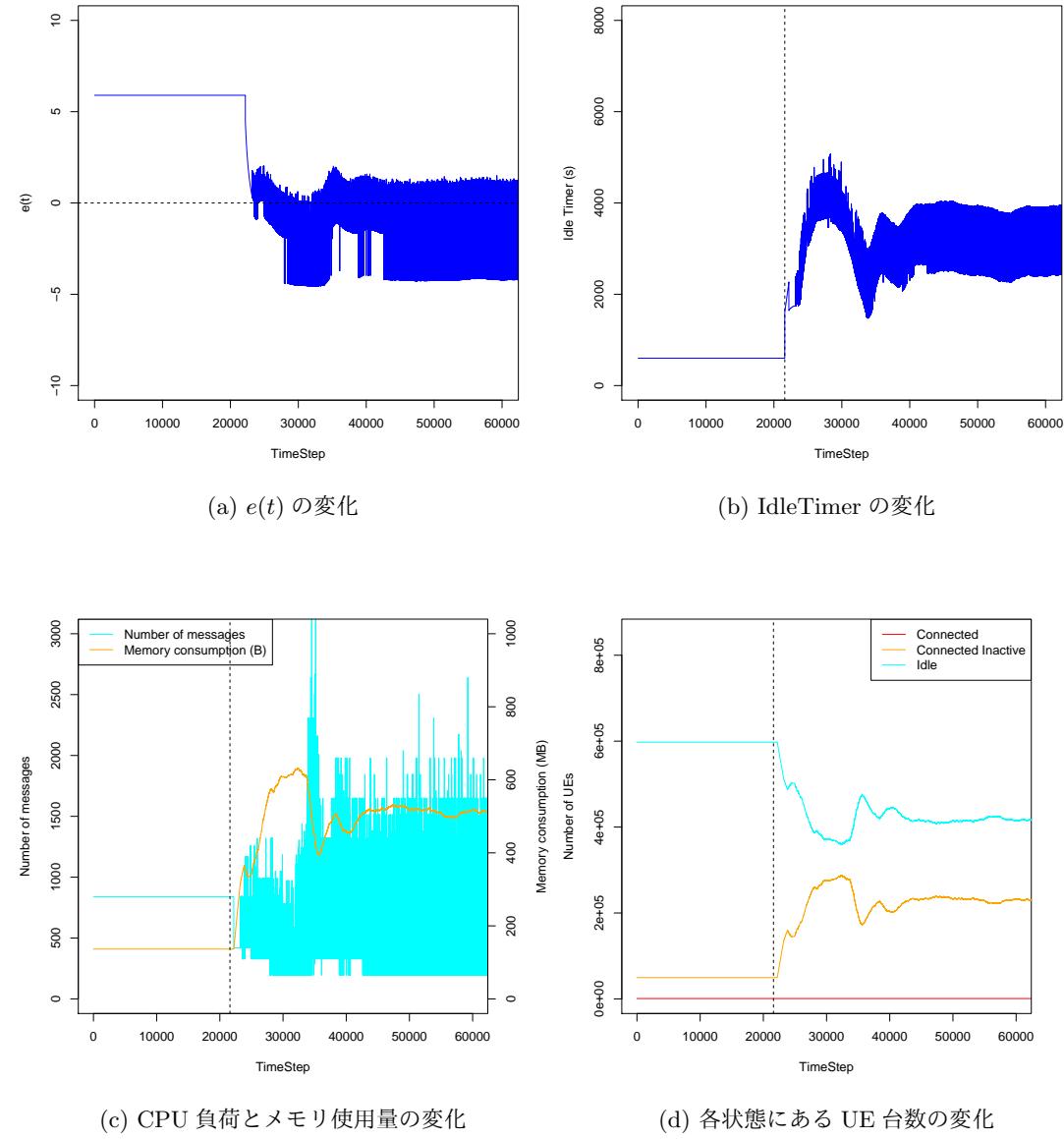


図 9: 理想 PID( $K_p = 1.8$ 、 $K_i = 0.00045$ 、 $K_d = 1800$ )

以上の図 7、8 図 9 を比較すると、図 7 に示した P 制御が最も制御が安定していることがわかる。また、 $E(t)$  が 0 に収束するまでの時間も最も短くなっている。以上の結果より、比例ゲインに 1.5 を設定した P 制御が Idle タイマの制御に最も適していると言える。

一方で、PI 制御は、P 制御と比較して良い結果が得られなかった。積分ゲインを利用しても制御が改善しなかった理由は、比例ゲインのみで制御した場合に発生するオフセット（定常偏差）が小さいことがあげられる。オフセットとは、定常状態の時に出力値と目標値の差である。一般的にオフセットを 0 に収束させるために積分制御が用いられるが、今回評価している Idle タイマの制御では、元々オフセットが小さい。よって、積分制御を導入するのメリットが小さくなっている。

また、PID 制御では、微分ゲインを利用することにより、偏差発生から定常状態に至るまでの過渡応答特性を改善することができた。これは一般的に微分ゲインを導入するメリットの一つである。しかし、一般的に、微分制御は、「対象の変化」ではなく、ノイズにより過敏に反応する危険性がある。本評価では、離散的な負荷の変動が発生しているため、Idle タイマの制御が不安的になっている。

### 3 実用 PID 制御についての調査

第 2 章の評価において、PID 制御の微分動作が離散的な負荷の変動の影響を受けやすいため、今回の評価においては制御が不安定になることがわかった。通常の微分（完全微分）を用いて動作する PID 制御のことを“理想 PID 制御”とよび、以下の式 (8) で表せる。以下の図 10 に、理想 PID 制御のブロック図を示す。

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

ここで、積分ゲイン  $K_i = K_p/T_i$ 、微分ゲイン  $K_d = K_p \cdot T_d$  と表し、式 (8) に代入すると、式 (9) になる。

$$u(t) = K_p \cdot (e(t) + \frac{1}{T_i} \cdot \int_0^t e(\tau) d\tau + T_d \cdot \frac{de(t)}{dt}) \quad (9)$$

式 (9) をラプラス変換すると式 (10) になり、伝達関数  $C(s)$  は式 (11) となる。

$$\begin{aligned} \mathcal{L}[u(t)] &= \mathcal{L}[K_p \cdot (e(t) + \frac{1}{T_i} \cdot \int_0^t e(\tau) d\tau + T_d \cdot \frac{de(t)}{dt})] \\ U(s) &= K_p \cdot (1 + \frac{1}{T_i \cdot s} + T_d \cdot s) \cdot E(s) \end{aligned} \quad (10)$$

$$C(s) = \frac{U(s)}{E(s)} = K_p \cdot (1 + \frac{1}{T_i \cdot s} + T_d \cdot s) \quad (11)$$

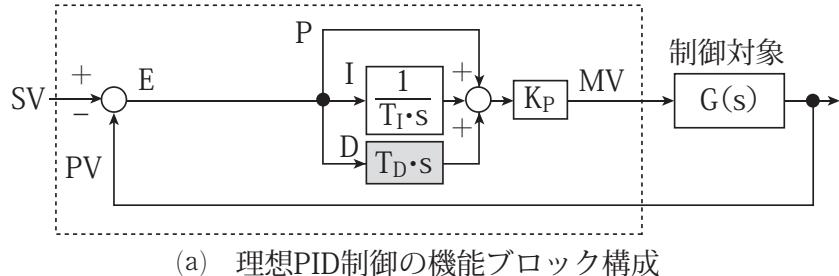


図 10: 理想 PID 制御のブロック図

理想 PID 制御の持つ、ノイズに弱いという欠点を解決するために、ノイズを抑制するローパスフィルタ（一次遅れフィルタ）を組み込んだ制御を実用 PID 制御という。実用 PID 制御のブロック図を以下の図 11 に示す。実用 PID 制御を伝達関数で表現すると、以下の式 (12) になり、プロッ

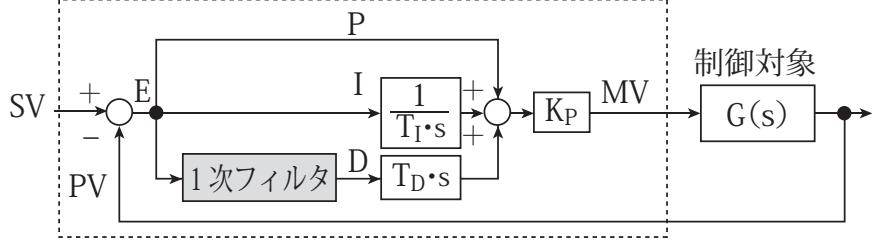


図 11: 実用 PID 制御のブロック図

ク図は図 12 のようになる。このように、フィルタを導入した微分制御のことを不完全微分という。ここで  $\eta$  は微分係数という定数であり、通常 0.1 から 0.125 の値を設定する [1]

$$C(t) = \frac{U(s)}{E(s)} = K_p \cdot \left\{ 1 + \frac{1}{T_i \cdot s} + \frac{T_d \cdot s}{1 + \eta \cdot T_d \cdot s} \right\} \quad (12)$$

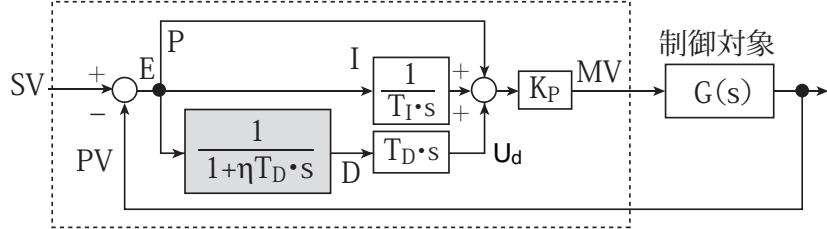


図 12: 実用 PID 制御のブロック図 (伝達関数での表現)

式 (12) に示した伝達関数のうち、微分制御に関する部分を抽出した式を以下の式 (13) に示す。

$$U_d(s) = \frac{T_d \cdot s}{1 + \eta \cdot T_d \cdot s} \cdot E(s) \quad (13)$$

式 (13) を式 (14) のように式変形する。

$$\begin{aligned} U_d(s) \cdot (1 + \eta \cdot T_d \cdot s) &= T_d \cdot s \cdot E(s) \\ U_d(s) + \eta \cdot T_d \cdot s \cdot U_d(s) &= T_d \cdot s \cdot E(s) \end{aligned} \quad (14)$$

そして、逆ラプラス変換を行うことにより、通常の微分方程式になる (15)。

$$\begin{aligned} \mathcal{L}^{-1}[U_d(s) + \eta \cdot T_d \cdot s \cdot U_d(s)] &= \mathcal{L}^{-1}[T_d \cdot s \cdot E(s)] \\ u_d(t) + \eta \cdot T_d \cdot \frac{d}{dt} u_d(t) &= T_d \cdot \frac{d}{dt} e(t) \\ u_d(t) &= T_d \left( \frac{d}{dt} e(t) - \eta \cdot \frac{d}{dt} u_d(t) \right) \end{aligned} \quad (15)$$

以上の議論より、実用 PID 制御は以下の式で表すことができる。

$$u(t) = K_p \cdot \{e(t) + \frac{1}{T_i} \cdot \int_0^t e(\tau) d\tau + T_d \left( \frac{d}{dt} e(t) - \eta \cdot \frac{d}{dt} u_d(t) \right)\} \quad (16)$$

## 4 移動平均、実用 PID 制御

表 4 に示した PID 制御の値を  $K_p$ 、 $K_i$  および  $K_d$  にそれぞれ設定し、シグナリング頻度 ( $s(t)$ ) を指数移動平均で平滑化した値 ( $\overline{s(t)}$ ) を PID 制御への入力とした場合の評価結果を図 13、図 14 および図 15 に示す。

$$\overline{s(t)} = \alpha \cdot s(t) + (1 - \alpha) \cdot \overline{s(t-1)} \quad (17)$$

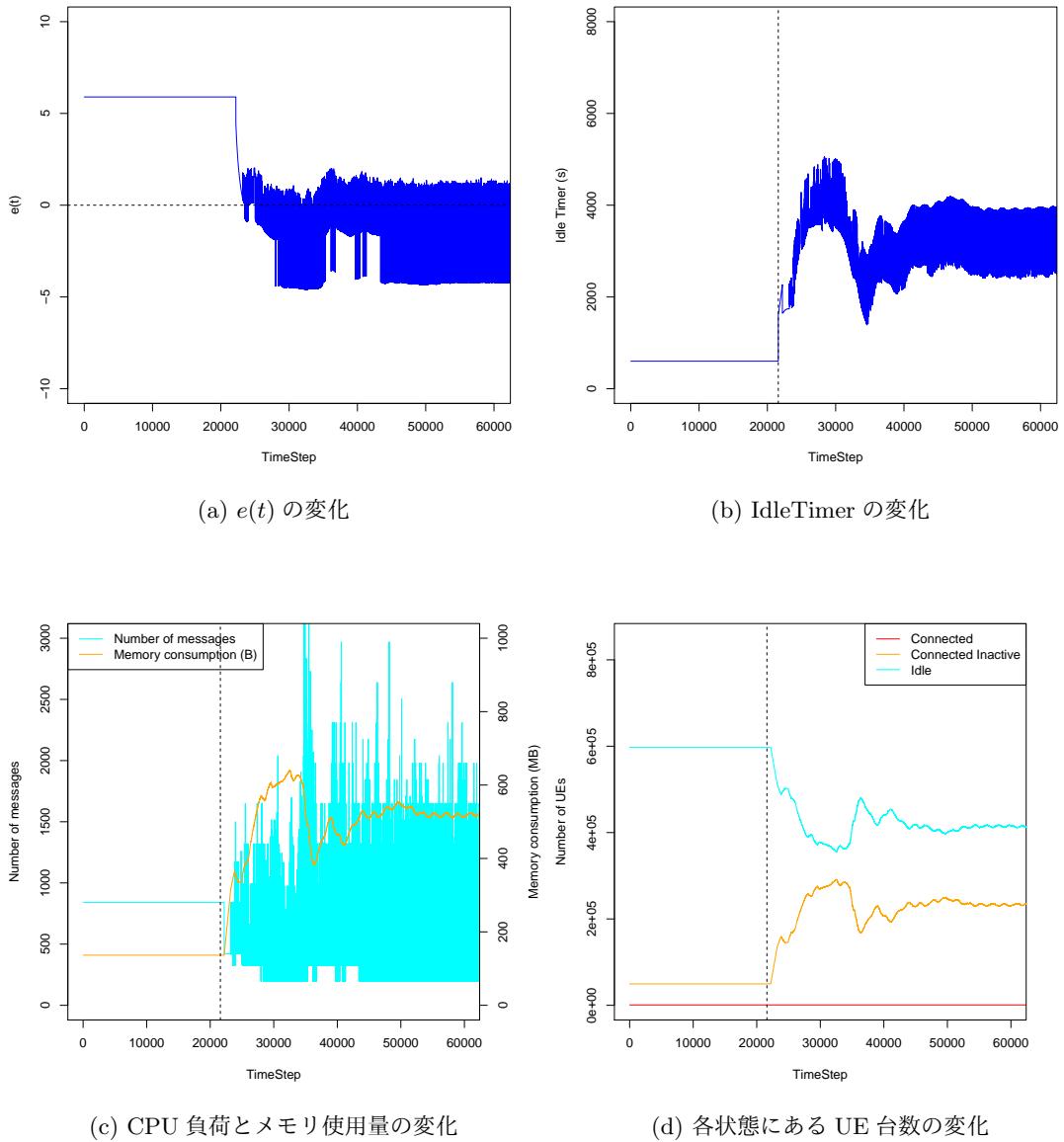


図 13: 理想 PID( $K_p = 1.8$ 、 $K_i = 0.00045$ 、 $K_d = 1800$ 、 $\alpha = 0.8$ )

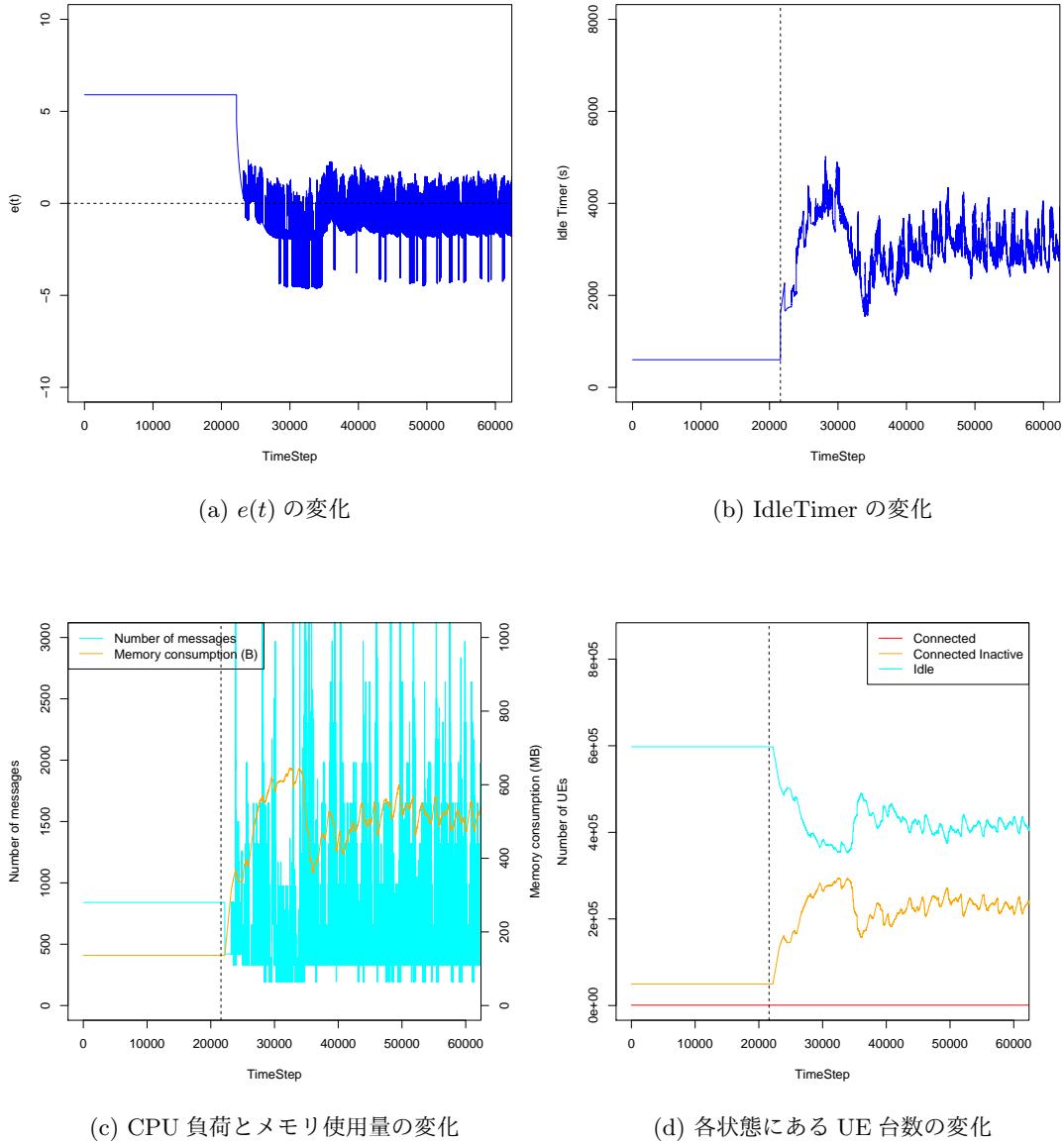


図 14: 理想 PID( $K_p = 1.8, K_i = 0.00045, K_d = 1800, \alpha = 0.1$ )

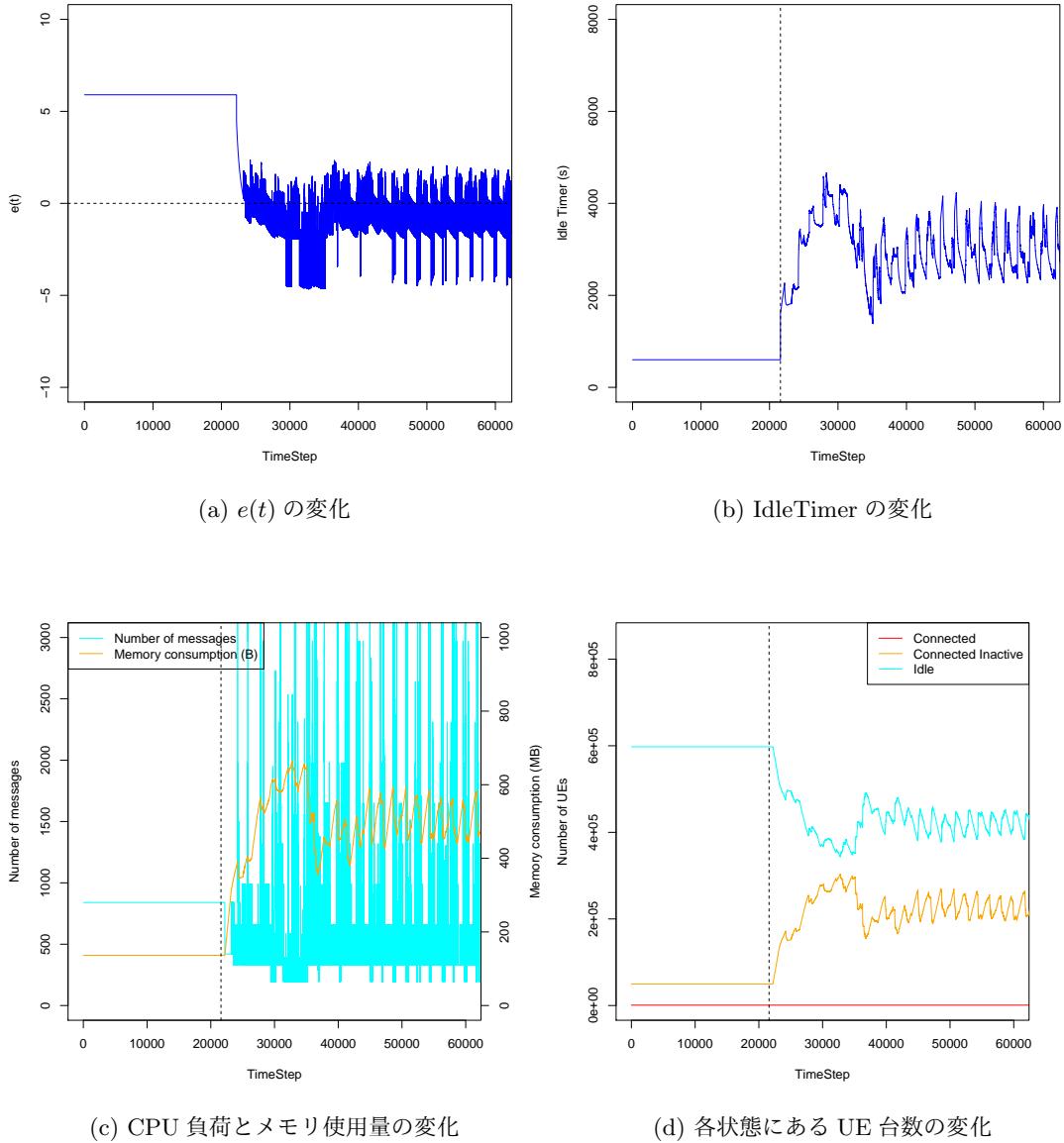


図 15: 理想 PID( $K_p = 1.8, K_i = 0.00045, K_d = 1800, \alpha = 0.01$ )

表4に示したPID制御の値を $K_p$ 、 $K_i$ および $K_d$ にそれぞれ設定し、不完全微分を用いた実用PID制御を行った場合の評価結果を図16、図17および図18に示す。

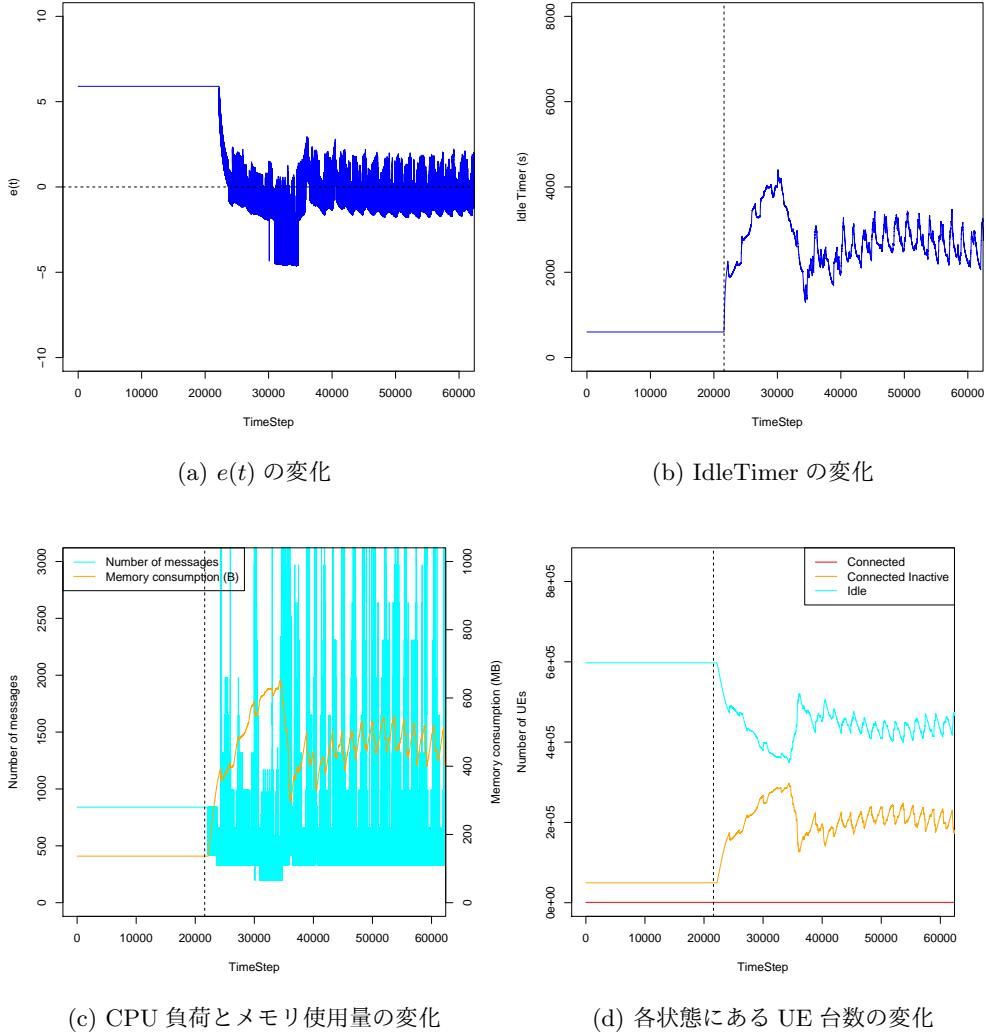


図 16: 理想 PID( $K_p = 1.8$ 、 $K_i = 0.00045$ 、 $K_d = 1800$ 、 $\eta = 0.125$ )

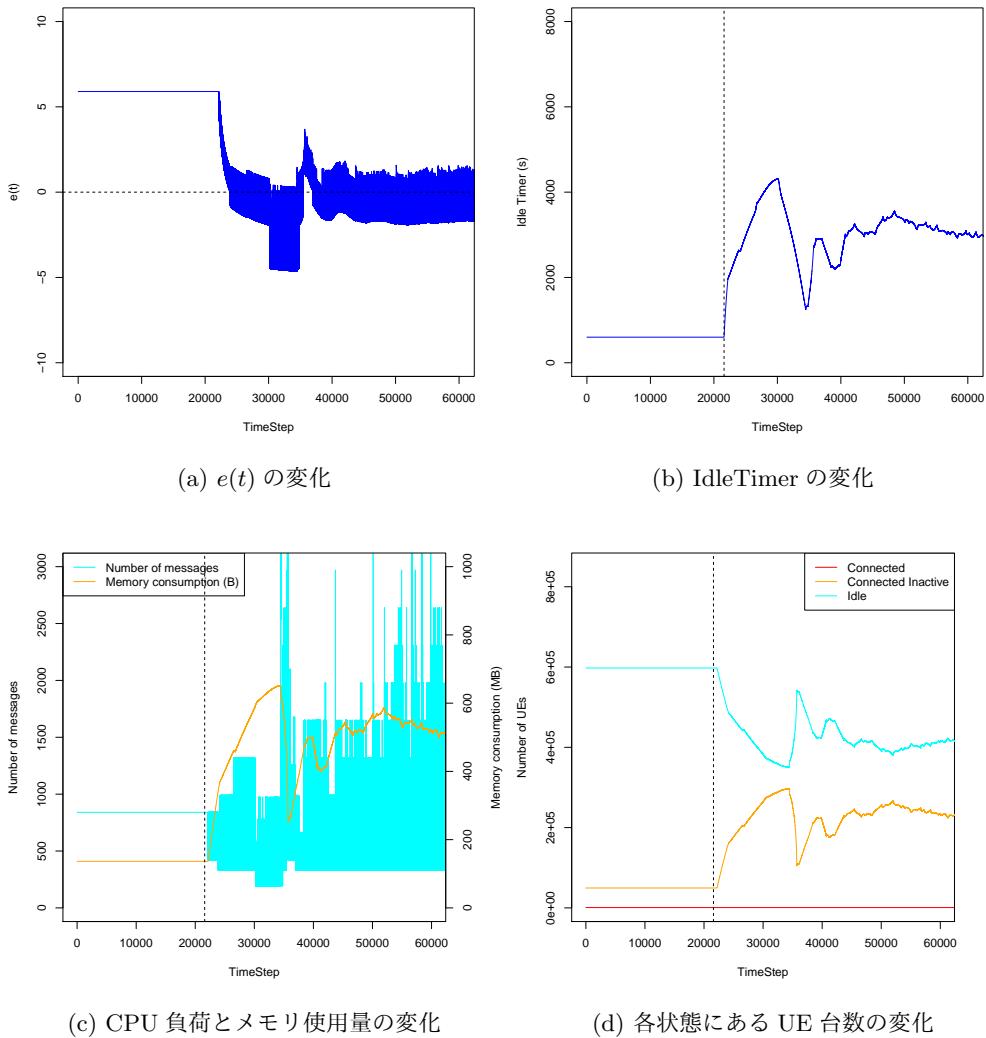


図 17: 理想 PID( $K_p = 1.8, K_i = 0.00045, K_d = 1800, \eta = 0.5$ )

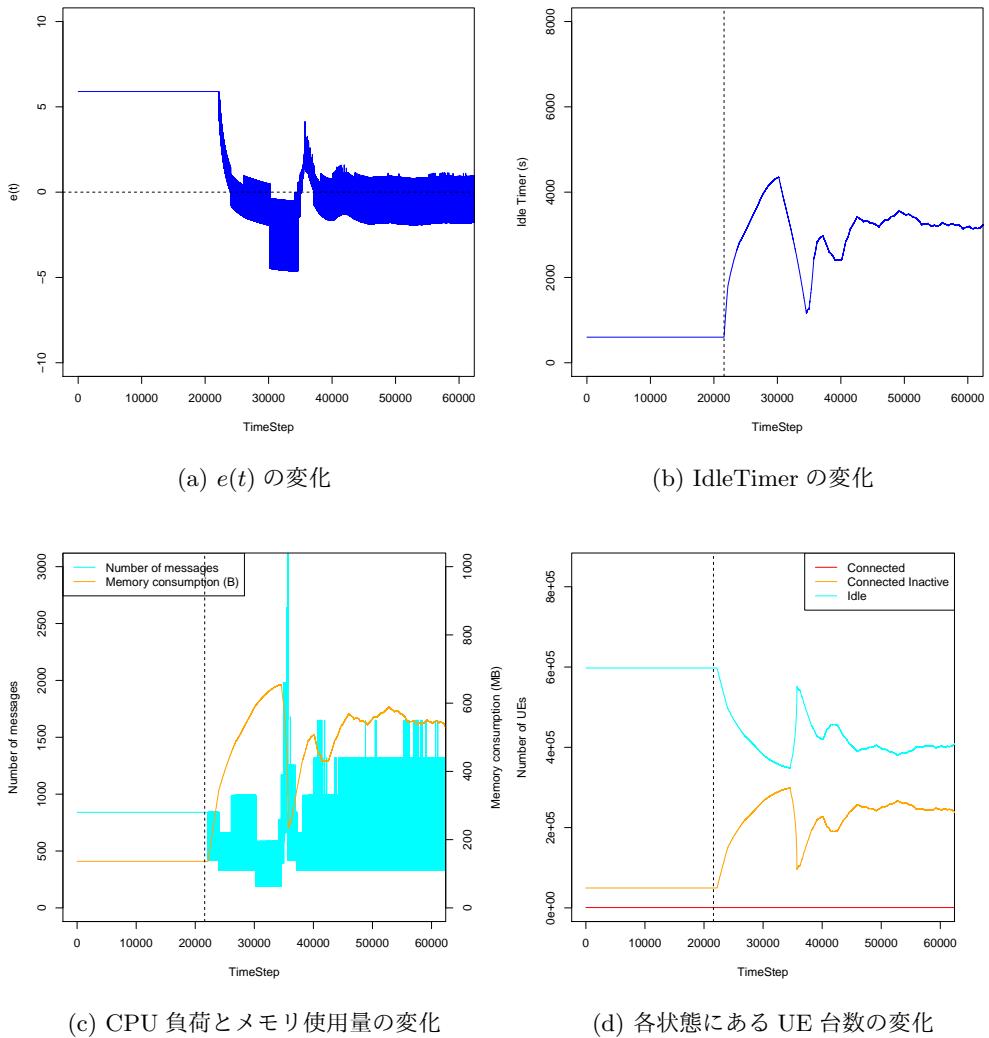


図 18: 理想 PID( $K_p = 1.8, K_i = 0.00045, K_d = 1800, \eta = 0.8$ )

以上の結果より、指数移動平均や実用 PID 制御を用いて入力の離散的な変動を平滑化することによって、そのような制御を行わない場合(図9)と比較して制御が安定することがわかる。しかし、依然として Idle タイマの制御が十分安定しているとは言えない。

指数移動平均を用いた結果を見ると、 $\alpha$  の値が小さい場合、短い周期での Idle タイマが変動が安定することがわかる。一方で、オーバーシュートの発生により、周期の大きな変動が発生していることがわかる。

同様に、実用 PID 制御を用いた結果を見ると、微分係数  $\eta$  の値が大きい場合、短い周期での Idle タイマの変動を抑制できることがわかる。一方で、オーバーシュートの発生により、周期の大きな変動が発生していることがわかる。

## 5 制御の性質

**比例制御** 比例ゲインが小さい場合は制御が収束するまでにかかる時間が増加し、比例ゲインが大きい場合はオーバーシュートやハンチング等が発生して制御が不安定になる(一般的な性質)。

**積分制御** 一般的に積分ゲインは定常偏差を抑制するために有効であるが、本評価では定常偏差が発生しないため、積分制御の効果は限定的であると言える。

**微分制御** 微分制御により、過渡応答特性を改善することが可能である(一般的な性質)。一方で、本評価では、離散的な負荷の変動が発生しているため、制御が不安定になる。

**指数移動平均を用いた制御** 指数移動平均を用いることで離散的な負荷の変動を平滑化して、制御を安定させることができ。制御の安定性と、変動に対する応答性はトレードオフの関係がある。平滑化係数の値を小さくすると離散的な負荷の変動を平滑化できるが、オーバーシュートやハンチングが発生しやすくなる。

**実用 PID 制御** 不完全微分を用いることで、一時的な負荷の変動を平滑化して制御を安定させることができます。制御の安定性と、変動に対する応答性はトレードオフの関係がある。微分係数の値を大きくすると、一時的な負荷の変動を平滑化できるが、過渡応答特性が低下する。

## 6 異なるシナリオでの評価

UE 台数は 480,000 台であり、UE の持つ通信周期とそれぞれの通信周期を持つ UE の割合は表 5 の通りである。また、各パラメータは表 2 と同じである。このシナリオにおいて、Idle タイマを

表 5: UE の通信周期の分布

	通信周期					合計
	100 s	200 s	300 s	…	6000 s	
UE 台数の割合	$\frac{1}{60}\%$	$\frac{1}{60}\%$	$\frac{1}{60}\%$	…	$\frac{1}{60}\%$	100%
UE 台数	8000	8000	8000	…	8000	480,000

10 s から 6,000 s まで変化させた場合における、メッセージ処理頻度とメモリ使用量の関係を図 19 に示す。

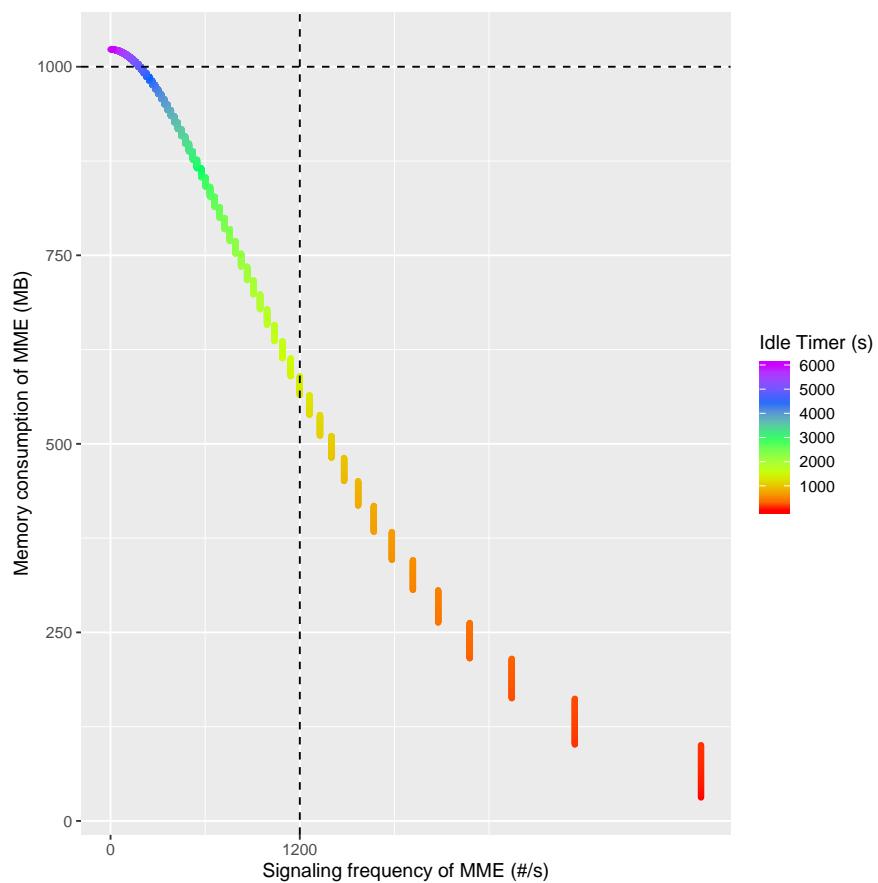


図 19: Idle タイマに対する、メッセージ処理頻度とメモリ使用量の関係

Idle タイマを 1,500 s に固定した時の評価結果を図 21 に示す。シミュレーションの初期状態では、ネットワークに接続している UE は 0 台である。その後、1 タイムステップ毎に通信周期の異なる UE がそれぞれ 1 台ずつ（合計 60 台ずつ）ネットワークに接続しデータ送信を行う。

図 20c を見ると各リソースの負荷が変動していることがわかる。Idle タイマの制御を行っていない状態において負荷が変動する理由は、UE の通信周期が異なるため、データ送信のタイミングの同期があるためである。また、最初の 8000 s 間は過渡的な状況であるが、その後は安定している。

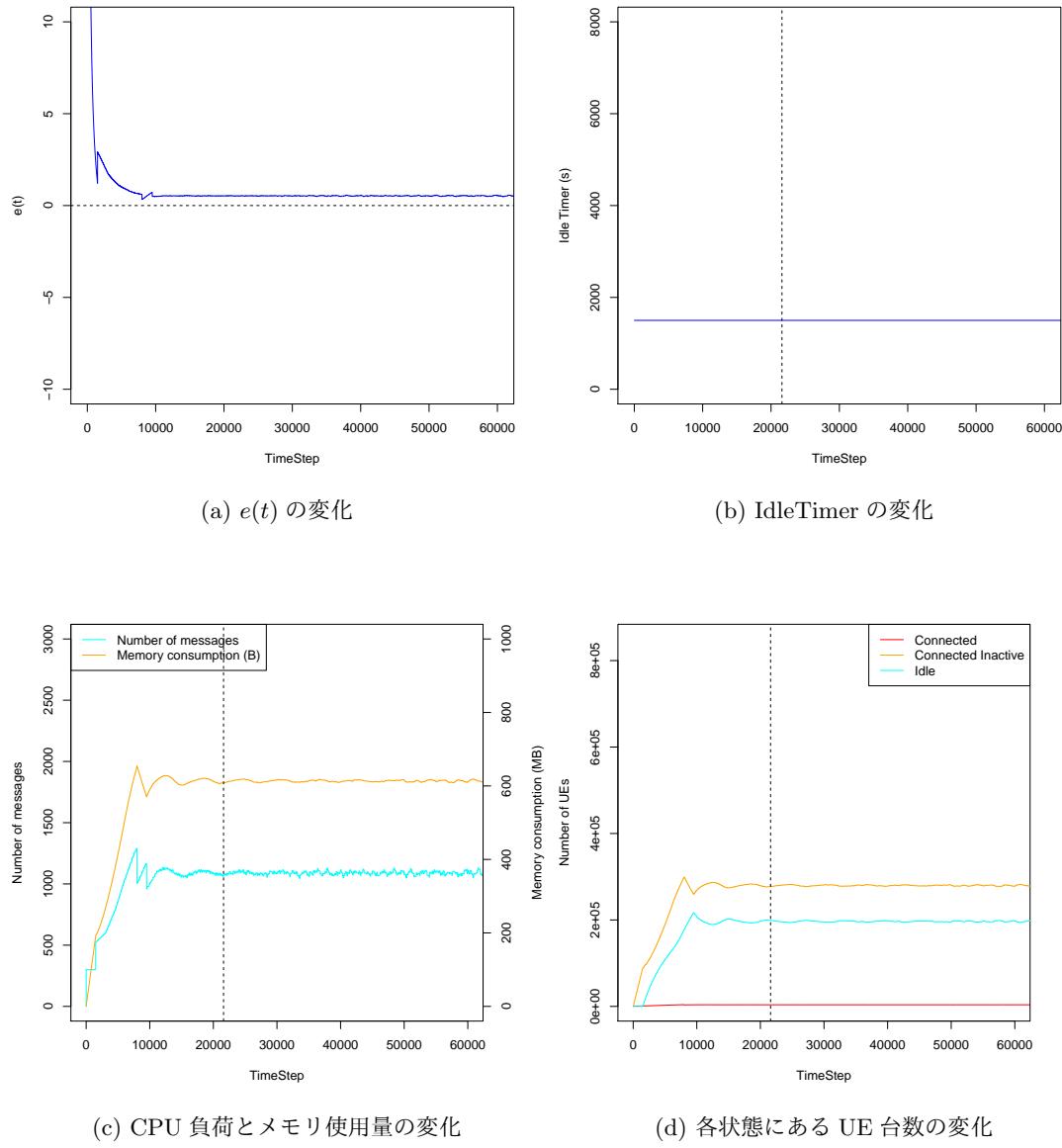


図 20: 理想 PID( $K_p = 0, K_i = 0, K_d = 0$ )

表 4 に示した P 制御の値を  $K_p$ 、 $K_i$  および  $K_d$  にそれぞれ設定した場合の評価結果を図 21 に示す。



図 21: 理想 PID( $K_p = 1.5$ 、 $K_i = 0$ 、 $K_d = 0$ )

表 4 に示した PI 制御の値を  $K_p$ 、 $K_i$  および  $K_d$  にそれぞれ設定した場合の評価結果を図 22 に示す。

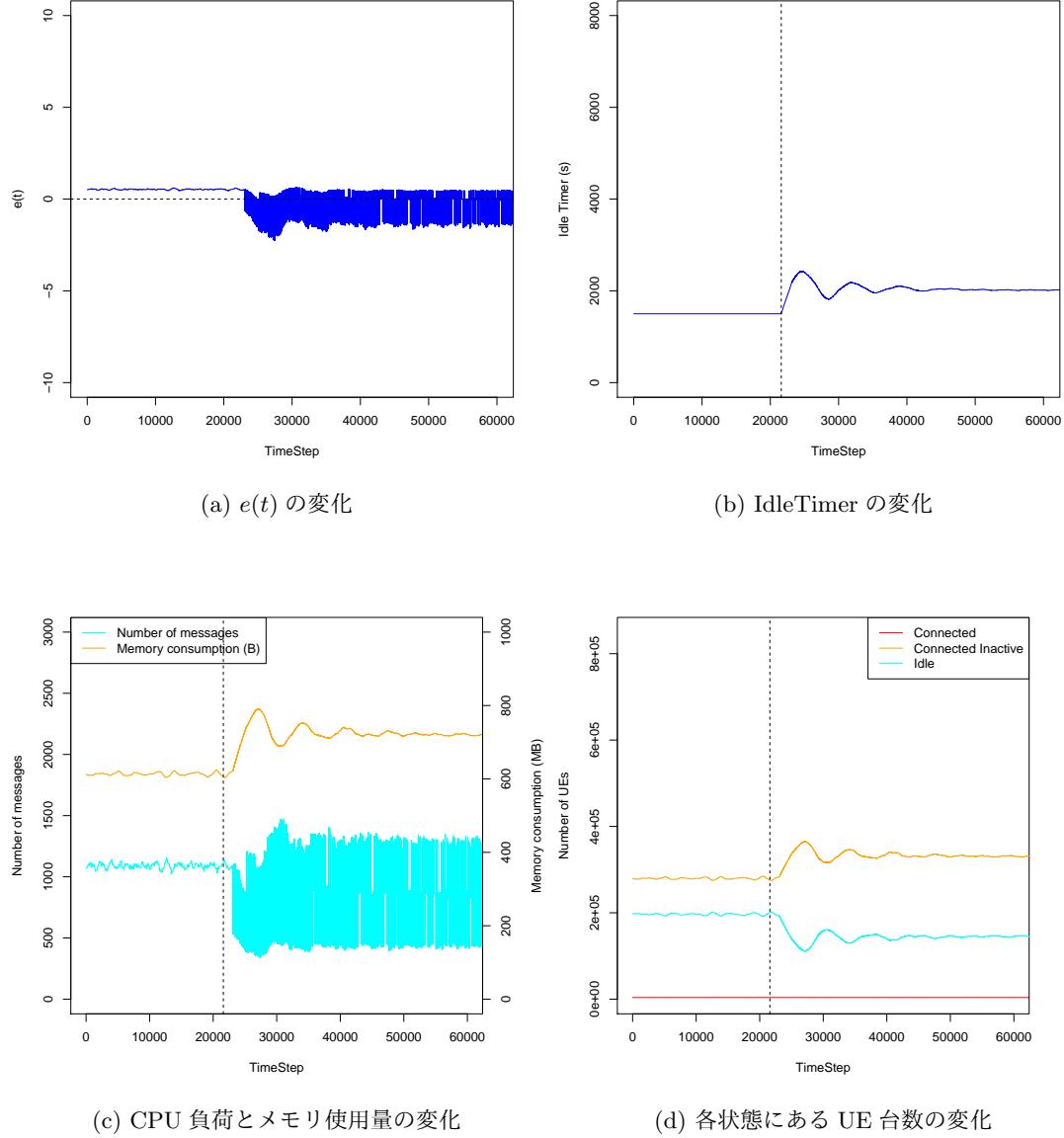


図 22: 理想 PID( $K_p = 1.35$ ,  $K_i = 0.000203$ ,  $K_d = 0$ )

表4に示したPID制御の値を $K_p$ 、 $K_i$ および $K_d$ にそれぞれ設定した場合の評価結果を図23に示す。

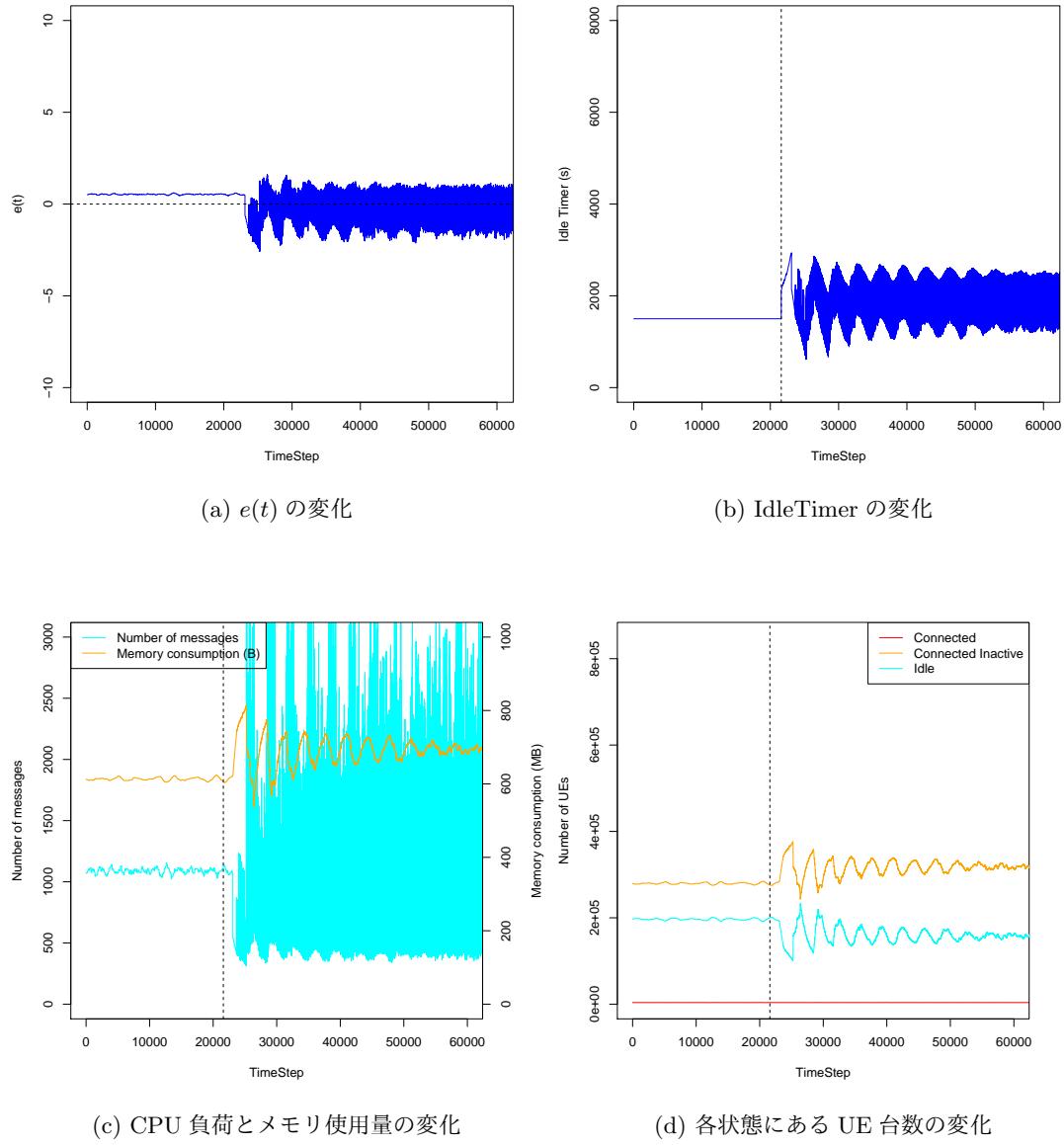


図 23: 理想 PID( $K_p = 1.8$ 、 $K_i = 0.00045$ 、 $K_d = 1800$ )

## 7 今後の予定

- PID制御の出力値( $y(t)$ )に関する式を変更した上で、これまで行っていた一連の評価結果を差し替える(次回の報告まで)
- 制御の安定性と制御方式の関係について整理する

- リソースが不足した際の制御を考える

1/10 CQ 研究会のタイトル及び著者、概要を決定し、発表申し込み

1/20 両論文の目次案策定

2/10 修士論文をメインに両論文の執筆

2/18 修士論文の提出と発表

2/29 CQ 研究会に提出する論文の修正及び発表準備

3/5、6 CQ 研究会での発表

## 参考文献

[1] “実用 PID に向けての工夫（その 1）.” [Online]. Available: [https://www.nikko-pb.co.jp/products/k\\_data/28P\\_31P\\_13.pdf](https://www.nikko-pb.co.jp/products/k_data/28P_31P_13.pdf)