ミーティング資料

安達智哉
to-adachi@ist.osaka-u.ac.jp

2019 年 9 月 12 日

1　CQ 研究会

論文を執筆中である。

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 およびメモリのリソース使用率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、第 2.1 節で Idle タイマを制御する上での目的関数を定義する。そして第 2.2 節で目的関数を最小化するための Idle タイマの制御方法を述べる。

2.1　目的関数の定義

本節では、UE の強制的な状態変化を引き起こさないという前提で、Idle タイマを適切な値を定義する。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ないとする。

シグナリング処理及び 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をそれぞれ $C^{\max}$，$M^{\max}$ と定義する。Idle タイマを T^{i} とした時の各リソースの使用量を $C^{T^{\mathrm{i}}}$ および $M^{T^{\mathrm{i}}}$ と定義する。各リソースの使用率 $U_{\mathrm{cpu}}(T^{\mathrm{i}})$ および $U_{\mathrm{memory}}(T^{\mathrm{i}})$ をそれぞれ以下の式 1、2 のように定義する。

$$U_{\mathrm{cpu}}(T^{\mathrm{i}}) = \quad C(T^{\mathrm{i}})/C^{\max} \tag{1}$$

$$U_{\mathrm{memory}}(T^{\mathrm{i}}) = \quad M(T^{\mathrm{i}})/M^{\max} \tag{2}$$

$U_{\mathrm{cpu}}(T^{\mathrm{i}})$ および $U_{\mathrm{memory}}(T^{\mathrm{i}})$ がどちらも小さい値を取るような T^{i} の設定が MME にとって良いことは明らかであるが、$U_{\mathrm{cpu}}(T^{\mathrm{i}})$ と $U_{\mathrm{memory}}(T^{\mathrm{i}})$ の間にはトレードオフの関係があるため、$U_{\mathrm{cpu}}(T^{\mathrm{i}})$ と $U_{\mathrm{memory}}(T^{\mathrm{i}})$ の双方を最小化させることは一般的に困難である。

一方で、突発的な負荷の増加に対応するという観点から、最も余力のある状態が T^{i} の最適値とする考え方がある。そのような最適化を目標とする場合、$U_{\mathrm{cpu}}(T^{\mathrm{i}})$ および $U_{\mathrm{memory}}(T^{\mathrm{i}})$ がどのような時に、最も余力のある状態と言えるのかを考える必要がある。まず、突発的な負荷は、現在のネットワークに存在する UE と同じような通信周期を持つ UE が大量にネットワークに参加することにより発生すると仮定する。この仮定に基づくと以前の評価結果より、$U_{\mathrm{cpu}}(T^{\mathrm{i}})$ および $U_{\mathrm{memory}}(T^{\mathrm{i}})$ は共に UE 台数に対して比例して増加することになる。つまり、UE 台数が α 倍になった時、CPU とメモリの負荷はそれぞれ $\alpha \cdot U_{\mathrm{cpu}}(T^{\mathrm{i}})$ および$\alpha \cdot U_{\mathrm{memory}}(T^{\mathrm{i}})$ になる。今回の条件では、CPU およびメモリのどちらか一方でも過負荷状態になると、UE の収容が難しくなるため、使用率の高い方のリソースがボトルネックになることに注意し、以下の式 (4) ように目的関数を定義する。

$$\begin{aligned} &\text{maximize :} && \alpha && \\ &\text{subject to :} && \max\{\alpha \cdot U_{\mathrm{cpu}}(T^{\mathrm{i}}), \alpha \cdot U_{\mathrm{memory}}(T^{\mathrm{i}})\} \leq 1 && (3) \\ & && && (4) \end{aligned}$$

2.2 Idle タイマの制御方法

今回の評価では、CPU 負荷は Idle タイマの値に対して広義単調減少 (単調非増加) でありかつ、メモリ負荷は Idle タイマの値に対して広義単調増加 (単調非減少) であるため、CPU 負荷を削減する場合には Idle タイマを増加させ、メモリ負荷を削減する場合はその逆の操作を行えば良い。このことを考慮しつつ、目的関数 (f) を最小化するように、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、リソースの使用率を観測して、f を小さくする向きに 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 の分布によって、Idle タイマを変化がリソース負荷へ与える影響が変化す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。

3 今後の予定

- 論文の執筆
  - 5,6 章: 〜 9/18
  - 完成: 〜 9/25
- 発表スライドの作成: 〜10/16