

進捗報告資料

安達智哉

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

2019 年 8 月 20 日

1 Idle Timer の最適化

以前までの評価において、Idle Timer 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 Timer 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 Timer 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 Timer の制御方法に関して述べる。まず、第 1.1 節で Idle Timer を制御する上での目的関数を定義する。そして第 1.2 節で目的関数を最小化するための Idle Timer の制御方法を述べる。

1.1 目的関数の定義

CPU リソースの使用率およびメモリリソースの使用率をそれぞれ U_{cpu} および U_{memory} と定義する。 U_{cpu} および U_{memory} がどちらも小さい値を取るような Idle Timer の設定が MME にとって良いことは明らかであるが、 U_{cpu} と U_{memory} の間にはトレードオフの関係があるため、 U_{cpu} と U_{memory} の双方を最小化させることは一般的に困難である。

一方、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。そのため、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する必要がある。

上述のような点を考慮しつつ目的関数 f を以下の式 (1) のように定義する。 f は CPU とメモリのうち、使用率が高い方の値を取る。この目的関数 (f) を最小化するような Idle Timer の値が最適であると考えられる。

$$f = \max\{U_{\text{cpu}}, U_{\text{memory}}\} \quad (1)$$

1.2 Idle Timer の制御方法

今回の評価では、CPU 負荷は Idle Timer 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 Timer 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 Timer 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 Timer を制御すれば、Idle Timer 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 Timer 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 Timer を制御する。

この時、1 ステップごとの Idle Timer 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 Timer の変化量を大きく設定すると、Idle Timer 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 Timer 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 Timer 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 Timer 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 Timer の変化量を調整しつつ、最適値に近づけることができる。