進捗報告資料

安達智哉
to-adachi@ist.osaka-u.ac.jp

平成 30 年 12 月 26 日

1　デタッチ処理をサスペンドさせることで、CPU 負荷をメモリにオフロードするモデル

1.1　概要

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、ベ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) 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。また、M2M/IoT 端末は消費電力を抑えることを目的に、データの送信ごとにデタッチ処理を実行し、セッションを解放する特徴がある。この点においても、ネットワークから切り離されるまで、セッションを維持し続ける従来の端末とは異なる。

上述の M2M/IoT 端末の特性は、CPU とメモリのバランスの良い割り当てを難しくすると考えられる。なぜなら、M2M/IoT 端末はその通信特性から、多くのアタッチ処理およびデタッチ処理を引き起こし、CPU 負荷のみを増加させるため、従来の端末と比較すると、CPU とメモリに与える負荷の割合が異なるからである。また、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一因である。IoT 端末は、スマートフォンのような従来の端末とは異なり、家電や自動車、電気メーター、センサーなど様々な場所、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。このように、通信特性の異なりかつ、接続台数の予測が難しい IoT 端末の増加により、今後のネットワーク事業者は、CPU とメモリのバランスの良い割り当てがより難しいくなると予想される。

上述のようなネットワーク (CPU とメモリの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、リソースの過剰分配が発生する問題がある。なぜなら、サーバやインスタンス一台あたりのリソース量は固定であるため、細かい粒度でリソースを制御できないためである。また、CPU とメモリのリソース比は変

化しないことも、リソースの過剰供給の原因である。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過剰分配されたことになる。

このような背景から、CPU とメモリのリソース消費の予測が難しよ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、データセンターなどの分野では行われている。しかし、モバイルネットワークに特化した研究は行われていない。そこで、私はモバイルネットワークに特化した、CPU とメモリのリソースのオフロードを可能にする仕組みを考えた。この方法により、CPU が過負荷である場合は、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、CPU の負荷を増加させることによりメモリの負荷を削減できる。この仕組みにより、CPU とメモリのリソース消費の予測が難しい場合や、変動が激しいネットワークであっても、CPU およびメモリ双方のリソース利用率の向上が期待でき、収容可能な端末の増加が期待できる。

私の案は、本来ならデタッチ処理が行われるタイミングであってもあえてデタッチ処理を行わないようにすることである。つまり、M2M/IoT 端末がデータの送信を終え、電源を OFF にした後も、一部のセッションはそのまま維持しつつける。そして、M2M/IoT 端末が再び ON になりデータを送信する際には、以前と同じベアラを用いてデータを送信する。この方法によって、最初に一度アタッチ処理を行えば、その後のデータ送信においては、アタッチ処理は発生しないで済むため、アタッチおよびデタッチ処理を行うために必要な CPU 負荷を削減しすることが可能である。一方、本来は解放されるはずのセッション情報を維持し続けるため、その分、メモリの負荷が大きくなる。

1.2 負荷について

1.2.1 CPU 負荷

CPU 負荷は、主にアタッチやデタッチなどのシグナリング処理を各ノードで実行する際に発生する。時間当たりのシグナリング処理の実行回数の増加に伴い、負荷が増加する。CPU 負荷の増加は、各ノードにおける処理時間の増大を引き起こす。また、CPU が過負荷状態になると、それ以上のシグナリング処理の実行が困難になる。

1.2.2 メモリ負荷

メモリ負荷は、主にベアラなどのセッションを維持するために必要な情報 (ベアラ識別子、UE 情報、ステート情報など) を保持する際に発生する。維持するベアラの増加に伴い、負荷が増加する。メモリが過負荷状態になると、新しくベアラを確立するために、古いベアラを解放する必要がある。

1.2.3 負荷のオフロードについて

CPU とメモリのリソースを監視し、CPU リソースが枯渇し、メモリリソースが余っているような場合においては、CPU 負荷をメモリにオフロードすることができる (図**??**)。具体的には、ベ

アラをサスペンドする (UE がネットワークから切断されても、ベアラを維持する) 割合を増やすことによって、実現できる。逆に、CPU リソースが余っている状態で、メモリが枯渇した場合は、ベアラをサスペンドする割合を小さくすることで、メモリから CPU へ負荷のオフロードが可能である。

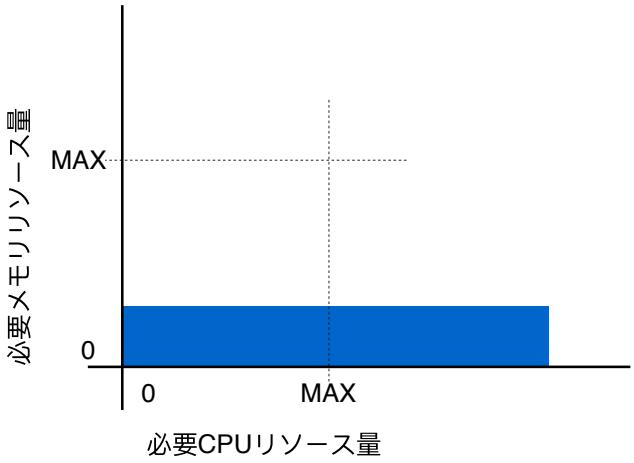

図 1: CPU 過負荷状態

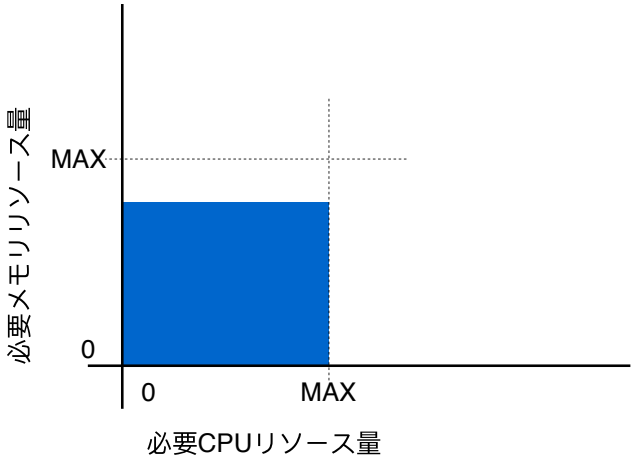

図 2: CPU 負荷をメモリへオフロード

1.3 説明

図 6 は、M2M/IoT 端末が最初にネットワークに接続した際の様子を示す。通常通り、アタッチの処理を行い、ベアラを確立する。図 7 は、M2M/IoT 端末がネットワークから切断された際の様子を示す。通常と異なり、デタッチ処理が行われないため、ベアラはそのまま維持される (UE-eNodeB 間の無線帯域は解放される)。図 8 は M2M/IoT 端末が再びネットワークに接続した時の様子を示す。この場合、以前使用したベアラが残っているため、M2M/IoT 端末はアタッチ処理をスキップしてデータの送受信を開始できる。

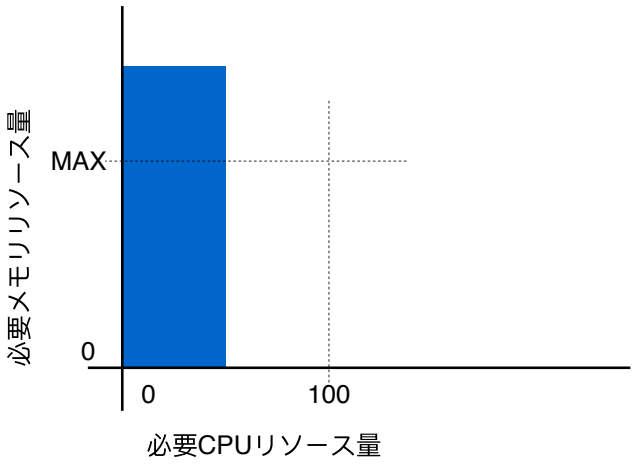

図 3: メモリ過負荷状態

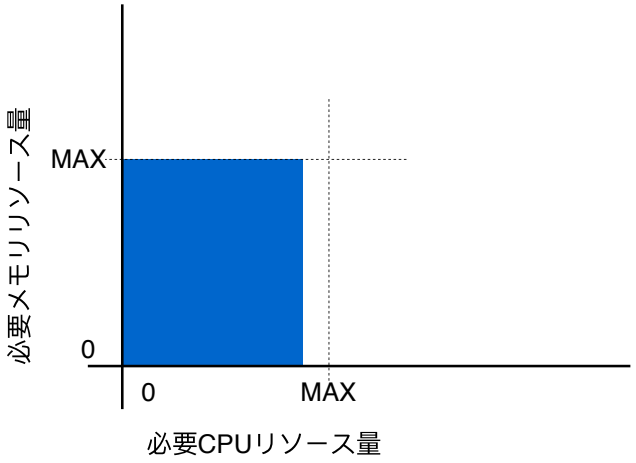

図 4: メモリ負荷を CPU へオフロード

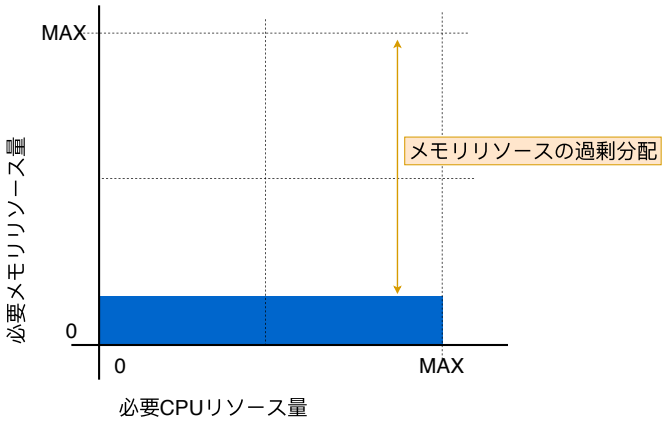

図 5: スケールアウトによるリソース増強

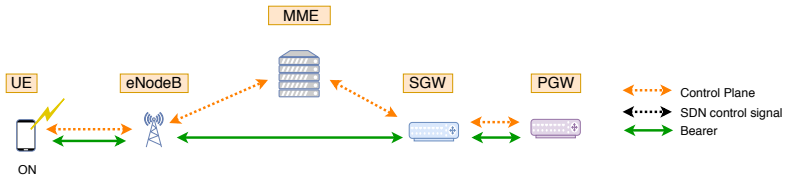

図 6: 端末が最初にネットワークに接続した時と処理

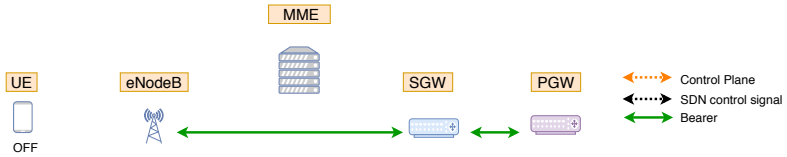

図 7: 端末がネットワークから切り離された時の処理

参考文献

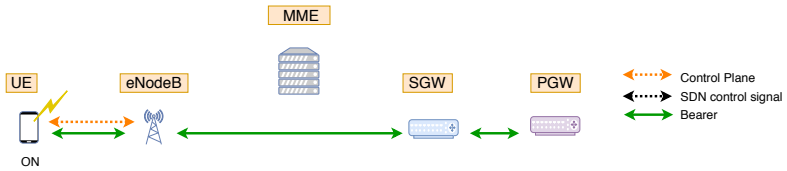

図 8: 端末が 2 回目以降にネットワークに接続した時の処理