社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

スマートシティにおけるクラウドレットの優先順位付けジョブオフローディングシステムの提案

横田　侑紀†　　宮田　純子†

† 芝浦工業大学　〒135–8548 東京都江東区豊洲 3–7–5
E-mail: †{ma23203,sumiko}@shibaura-it.ac.jp

あらまし　近年注目されているスマートシティにおいて，街のあらゆるセンサ情報や自動運転と街中でのドローン飛行などに必要な処理を低遅延で行う方法として，小規模で可用性の高いクラウドレットネットワークを街中に設置し，ユーザから限りなく近い位置で処理を行うことが提案されている．スマートシティにおける処理は多岐にわたり，適切な順番で処理が行われるべきであると同時に，クラウドレットの容量を考慮すると，一つのクラウドレットで処理できるタスクには限りがあり，ユーザからのジョブを適切なクラウドレットへとオフロードする必要がある．本稿ではこれらを待ち行列理論を用いてモデル化し，優先順位を考慮していない場合よりもシステム全体の平均遅延時間が小さくなるオフロード割合を導出する．

キーワード　スマートシティ，クラウドレット，オフローディング，待ち行列理論

Proposal of Prioritized Job Offloading Cloudlet System in Smart City

Yuki YOKOTA† and Sumiko MIYATA†

† Shibaura Institute of Technology First University　3–7–5 Toyosu, Koto-ku, Tokyo, 135–8548 Japan
E-mail: †{ma23203,sumiko}@shibaura-it.ac.jp

Abstract　Smart cities have recently gained attention for proposing the installation of a network of small, high-availability cloudlets within the city. This allows for the processing of sensor information, automatic driving, and drone flights with low-latency, as close to the user as possible. Processing in a smart city should be done in a specific order, while considering the capacity of cloudlets. There is a limit to the number of tasks that a single cloudlet can process, and jobs from users must be offloaded to the appropriate cloudlet. This paper models these processes using queuing theory and derives an offloading ratio that reduces the average delay of the entire system compared to the case where priority is not taken into account.

Note: This report is not a reviewed paper.

Key words　Smart City, Cloudlet, Offloading, Queuing Theory

1. はじめに

スマートシティとは，社会のあらゆる分野にまたがる数多くの技術が集積されたものであり，生活の質を向上させ，業務効率を高め，持続可能な発展を促進することが目的である [1]．公共事業や都市の交通だけでなく，環境や治安，教育においても数多くの都市サービスに対して様々なスマートシステムが提案されており，情報通信技術（ICT: Information and Communication Technology）は，こうしたシステムの基盤となるインフラとして機能し，リアルタイムのデータ収集，通信，分析を可能にすることで，スマートシティの広範なビジョンを支えることが求められている [2]．逆に言えば，スマートシティの実現には効率的で円滑なネットワークシステムが必要不可欠である．

スマートシティにおける円滑な通信を妨げる要因として，異なる種類の処理が必要となる情報が混在し，それぞれが要求する遅延時間も大きく異なるということが挙げられる [3]．例えば，自動運転やドローン制御に必要な処理は安全性や信頼性の観点から短い時間で処理が終わることが求められる反面 [4]，街のセンサ情報や監視カメラデータの処理など緊急性の低いジョブも存在する [5]．そのため，これらのジョブは一括で処理されるべきではなく，ネットワーク内で分散させて処理する必要があると考えられる [6]．

Mahadev ら [7] によって提唱されたクラウドレットは，ユーザと同一 LAN (Local Area Network) 内に存在し 1 ホップで接続可能なコンピュータクラスタ群である．サーバを基地局に設置するようなエッジコンピューティングと異なり，LAN(Local

Area Network 内のルータなどにクラウドレットを設置することでより小さい遅延でユーザと通信を行うことが可能となる．このようなクラウドレットをより広域な街単位の MWAN (Wireless Metropolitan Area network) で利用することが検討されている [8].

クラウドレットの問題点として，従来のクラウドコンピューティングやエッジコンピューティングと比べると規模が小さく，処理能力が著しく低いことが挙げられる．そのため，クラウドレットの限られた資源を効率よく利用することが求められる．無線アクセスネットワークにおける既存研究としてクラウドレットの配置問題に注目した手法 [9] があるが，分散型コンピューティングシステムでは到着した各ジョブがどのコンピュータで処理されるかを決めることで，システム全体の負荷分散を行うことが重要である [10]．携帯端末ユーザの複雑な移動パターン及び到着率のランダム性を考慮すると，クラウドレットの負荷の分散が大きくなることは免れられない．そこで，クラウドレット間でジョブのオフローディングを行って効率的な資源の利用を可能とし，負荷の分散を小さくできる手法の検討が必要である [11].

クラウドレット間でジョブのオフローディングを行う手法として，各ジョブの遅延時間の値を許容遅延と直接評価しつつ，クラウドレット間での負荷の偏りを限りなく小さくする手法が提案されている [12]．これにより，ユーザ満足度を満たしたうえで限られたクラウドレットの資源を最大限利用することが可能となっている．しかし，この研究では処理時間の異なるジョブの到着に関する言及はなく，スマートシティという環境で適用できる手法ではない.

街の分散型コンピューティング手法のひとつとして，ジョブをコンテンツごとに管理し，ユーザ間とユーザ-サーバ間で処理をオフロードする手法が Fang ら [13] によって提案されているが，この研究ではシステム全体の効用をもとにオフロードの有無を決定しており，ジョブごとの要求には注目していない.

そこで本稿では，異一般分布に従う処理時間を持つジョブがランダムに到着するようなクラウドレット環境を待ち行列理論を用いてモデル化し，ジョブの優先度を考慮しつつクラウドレットの平均遅延時間が最小化されるようなジョブのオフロード割合を決定する手法を提案する.

本稿の構成は以下の通りである．第 2 章でスマートシティにおける様々な機能とクラウドレットの負荷分散に関する研究について述べ，第 3 章でシステムのモデル化と，オフロード後のジョブの優先度を考慮した平均遅延時間を導出する．また，その平均遅延時間が最も小さくなるような最適オフロード割合を求めるためのアルゴリズムを示す．第 4 章で数値解析の結果をまとめ，第 5 章で本研究における結論と課題を述べる.

2. 関 連 研 究

スマートシティにおいて求められる機能として，大量のデータを管理する仕組み [6]，IoT デバイスやセンサの制御 [5]，その他アプリケーションを開発し，制御するためのプラットフォームなど，需要に合わせて多く検討されている．これらの機能はどれも持続可能なスマートシティを実現させるために必要な技術であるが，到着する処理の種類が混在するネットワーク環境についての理論的な解析はされていない.

スマートシステムの 1 つである自動運転技術における分散型ネットワークの例として，基地局に置かれたエッジサーバと、道路上に配置された RSU (Road Side Unit) に演算処理をオフロードするシステムが提案されている [4]．こちらの手法では待ち行列理論を用いた理論的な解析がされているが，処理するタスクの種類に関する言及はない．また，既存のクラウドレットの負荷を分散させる手法として，エリア内のユーザ密度に応じてクラウドレットを設置するアクセスポイントを決め，各ユーザの平均遅延時間が最も小さくなるような配置決定を行っている手法 [8], [9], [14] や，携帯端末からクラウドレットへのジョブ要求の割り当てを平均遅延時間などに基づいて決める手法 [15], [16]，そしてクラウドレット間でジョブをオフローディングを行う手法 [17], [18], [12] などが提案されている．その中でも Yokota ら [12] はクラウドレットを $M/M/1$ 待ち行列モデルを用いてモデル化し，ゲーム理論を用いたオフローディングによってクラウドレット間の負荷の差を限りなく小さくする手法を提案している．しかし，モデル化の妥当性に対する検討が不十分である.

3. 提 案 手 法

3.1 システムモデル

本稿では，スマートシティにおけるあらゆるセンサやドローン，自動運転，さらに，住人などのユーザから送られる様々な計算処理をジョブと定義し，これらを処理するため WMAN 内に設置されたクラウドレット $i \in \{1, 2, ..., N\} = C$, $N \geq 2$ について考える．スマートシティ内の大量のクラスタデータから，ユーザから送信されるジョブの到着率はポアソン分布に従うと仮定でき [19]，各ジョブの処理時間は一般分布に従うと仮定する．このようなシステムは，複数の独立した到着流を持つ $M/G/1$ でモデル化できる.

クラウドレット i には M 種類の異なる処理時間を持つジョブ $m \in \{1, 2, ..., M\}, M \geq 2$ が到着し，ポアソン分布に従うクラウドレット i への平均到着率を $\boldsymbol{\Lambda_i} = (\lambda_{i,1}, \lambda_{i,2}, ..., \lambda_{i,M})$ [jobs/s] のように定義する．独立なポアソン流の重畳は同様にポアソン流であると考えられるため，異なるジョブの到着流を合わせたクラウドレット i への全体的なジョブの到着率は次のように表される.

$$\lambda_i = \sum_{m=1}^{M} \lambda_{i,m} \tag{1}$$

また，各ジョブの処理時間は一般分布に従い，それぞれの平均処理時間を $\boldsymbol{b} = (b_1, b_2, ..., b_M)$ [s/job] と表したとき，重畳した到着流 λ_i の処理時間は

$$b_i^{\text{sum}} = \sum_{m=1}^{M} \frac{\lambda_{i,m}}{\lambda_i} b_m \tag{2}$$

であり，ジョブの処理時間の 2 次積率を $b_m^{(2)}$ とすると，

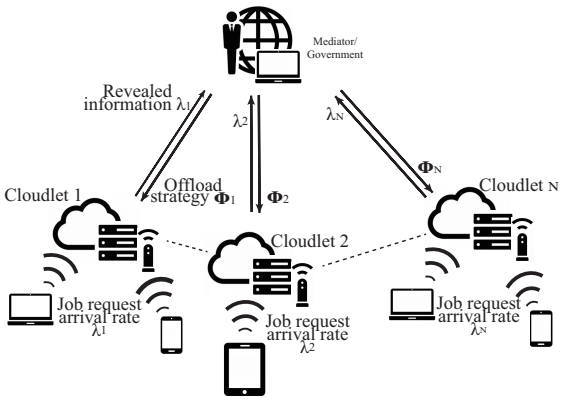

図 1　システムモデル

$$b_i^{\mathrm{sum}(2)} = \sum_{m=1}^{M} \frac{\lambda_{i,m}}{\lambda_i} b_m^{(2)} \tag{3}$$

と計算できる．

ジョブが FCFS で処理されるとき，上記のパラメタを用いて計算された単一の到着流をもつ $M/G/1$ 待ち行列モデルの結果が，独立な複数の到着流をもつ $M/G/1$ に適用できる．したがって，クラウドレット i に到着する重畳した到着流の利用率 $\rho_i = \lambda_i b_i^{\mathrm{sum}} < 1$ であると仮定したとき，定常状態におけるクラウドレット i の待ち行列遅延と処理遅延を合わせた平均遅延時間 T_i は次のように表せる．

$$T_i = \frac{\lambda_i b_i^{\mathrm{sum}(2)}}{2b_i^{\mathrm{sum}}(1 - \lambda_i b_i^{\mathrm{sum}})} \tag{4}$$

次に，クラウドレット間でジョブのオフロードを行った場合について考える．

図 1 に，本手法におけるシステムのオフロードモデルを示す．各クラウドレットへの到着率の情報は管理者へと集められ，それらの情報をもとにオフロード方法が決められる．ジョブがクラウドレット i に到着したとき，処理が行われる前にまずオフロードの有無が決められる．オフロードされないジョブは，到着したクラウドレット i でそのまま処理され，結果がユーザへと返される．オフロードされる場合，ジョブは他のクラウドレット $j, j \neq i$ へと送られ、そこで処理された結果がもとのクラウドレット i を経由してユーザへと返される．

クラウドレット i に到着したジョブのオフロード先を決めるため，割合ベクトル $\boldsymbol{\varphi}_i = (\varphi_{i1}, \varphi_{i2}, ..., \varphi_{iN}) \subset \mathbb{R}^N$, $\varphi_{ij} = [0, 1] \subset \mathbb{R}$, $\sum_{j=1}^{N} \varphi_{ij} = 1, i, j \in C$ を定義する．この時，φ_{ij} は i 番目のクラウドレットが j 番目のクラウドレットにオフロードするジョブの割合を示す．したがって，$\varphi_{ii}\lambda_i$ のジョブがオフロードされずにクラウドレット i で処理され，$\varphi_{ij}\lambda_i$ のジョブがクラウドレット j へとオフロードされる．

他クラウドレットから到着するオフロードジョブを新しい到着流だと考えると，オフロード後のクラウドレット i の到着率 $\lambda_i^{\mathrm{offload}}$ は重畳により

$$\lambda_i^{\mathrm{offload}} = \sum_{j=1}^{N} \sum_{m=1}^{M} \varphi_{ji}\lambda_{j,m} \tag{5}$$

と求められる．したがって，この到着流の処理時間とその 2 次積率は以下のように求められる．

$$b_i^{\mathrm{offload}} = \sum_{j=1}^{N} \sum_{m=1}^{M} \frac{\varphi_{ji}\lambda_{j,m}}{\lambda_i^{\mathrm{offload}}} b_m \tag{6}$$

$$b_i^{\mathrm{offload}(2)} = \sum_{j=1}^{N} \sum_{m=1}^{M} \frac{\varphi_{ji}\lambda_{j,m}}{\lambda_i^{\mathrm{offload}}} b_m^{(2)} \tag{7}$$

よって，オフロード後のクラウドレット i の平均遅延時間 T_i^{offload} は

$$\begin{aligned} T_i^{\mathrm{offload}} &= \frac{\lambda_i^{\mathrm{offload}} b_i^{\mathrm{offload}(2)}}{2b_i^{\mathrm{offload}}(1 - \lambda_i^{\mathrm{offload}} b_i^{\mathrm{offload}})} \\ &= \frac{b_m^{(2)}}{2b_m(1 - \sum_{j=1}^{N}\sum_{m=1}^{M} \varphi_{ji}\lambda_{j,m} b_m^{(2)})} \end{aligned} \tag{8}$$

となる．

3.2　優先順位を考慮した平均遅延時間

つづいて，クラウドレット i に到着するオフロード後のジョブの到着流のうち，特定のジョブ α が優先的に処理されることを考える．

クラウドレット i においてオフロード後のあるジョブ m の到着率の和は $\sum_{j=1}^{N} \varphi_{ji}\lambda_{j,m}$ であり，このときの利用率 $\rho_{m,i}$ は

$$\rho_{m,i} = b_m \sum_{j=1}^{N} \varphi_{ji}\lambda_{j,m} \tag{9}$$

となる．ジョブ α の平均遅延時間 $T_{i,\alpha}$ は，処理されるのを待っている他のジョブ α の平均処理時間の総和の平均と，ジョブ α が到着したときに処理されているジョブの平均残余寿命の和の 2 つの合計で求められる．よって，ジョブの処理時間の 2 次積率，式 (9) とリトルの公式を用いると，$T_{i,\alpha}$ は次のように計算される．

$$T_{i,\alpha} = \frac{\sum_{m=1}^{M} b_m^{(2)}\rho_{m,i}}{2b_m(1 - \rho_{\alpha,i})} \tag{10}$$

次に，クラウドレット i におけるジョブ α の次に優先して処理されるジョブ β の平均遅延時間 $T_{i,\beta}$ について考える．

ジョブ β がクラウドレット i に到着した後，処理が開始されるまで他に到着するジョブがなかったとすれば，平均遅延時間はジョブ α と同様に考えられる。実際には，自身の処理が行われるのを待っている間に新たに到着するジョブ α の処理を追加で待つことになるため，計算すると平均的に $\frac{1}{1-\rho_\alpha}$ 時間分多く待つ．よって，$T_{i,\beta}$ は

$$T_{i,\beta} = \frac{\sum_{m=1}^{M} b_m^{(2)}\rho_{m,i}}{2b_m(1 - \rho_{\alpha,i})(1 - \rho_{\alpha,i} - \rho_{\beta,i})} \tag{11}$$

と求められる．

最後に，ジョブ β の次に優先度の低いジョブ $m(m = 3, ..., M)$ の平均遅延時間について考える．

ジョブ α からジョブ $(m-1)$ までのジョブ m よりも優先度の高いジョブの到着流を重畳させ，新たなジョブ H としたとき，ジョブ m の遅延時間はジョブ β の遅延時間と同様に考えるこ

とができる．到着流の重畳によって得られるジョブ H のパラメータを以下に示す．

$$\lambda_H = \sum_{j=1}^{m-1} \lambda_{i,j} \tag{12}$$

$$b_H = \sum_{j=1}^{m-1} \frac{\lambda_{i,j}}{\lambda_H} b_j \tag{13}$$

$$b_H^{(2)} = \sum_{j=1}^{m-1} \frac{\lambda_{i,j}}{\lambda_H} b_j^{(2)} \tag{14}$$

上記パラメータと，利用率 $\rho_{H,i} = \rho_{1,i} + \rho_{2,i} + \ldots + \rho_{m-1,i}$ を用いてジョブ H とジョブ m として式 (11) に代入すると，ジョブ m の平均遅延時間 $T_{i,m}$ は

$$\begin{aligned} T_{i,m} &= \frac{\sum_{m=1}^{M} b_m \rho_{m,i}}{2(1-\rho_H)(1-\rho_H-\rho_m)} \\ &= \frac{\sum_{m=1}^{M} b_m \rho_{m,i}}{2(1-\sum_{j=1}^{m-1}\lambda_{i,j})(1-\sum_{j=1}^{m}\lambda_{i,j})} \end{aligned} \tag{15}$$

と導出できる．

3.3 最適オフロード割合の導出

本稿では，クラウドレット全体の平均遅延時間が最も小さくなるようなオフロードを行うことが最適であるとしている．そこで，オフロードによって発生しうるジョブの到着率の偏りを軽減するため，単純平均ではなくオフロード後に処理されるジョブの到着率の重みつき平均によって全体の平均遅延時間を計算する．

よって，式 (4) を用いればオフロードしない場合のクラウドレット全体の遅延時間は

$$T_{\text{nooffload}} = \sum_{i=1}^{N} \sum_{m=1}^{M} \lambda_{i,m} T_i \tag{16}$$

となる．また，式 (8) を用いればオフロードした場合のクラウドレット全体の遅延時間は

$$T_{\text{offload}} = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{m=1}^{M} \varphi_{ji} \lambda_{i,m} T_i^{\text{offload}} \tag{17}$$

となる．最後に，式 (10)，(11)，(15) を用いれば優先順位を考慮した場合のクラウドレット全体の遅延時間は

$$T_{\text{priority}} = \sum_{i=1}^{N} \sum_{j=1}^{N} \varphi_{ji} (\lambda_{j,\alpha} T_{i,\alpha} + \lambda_{j,\beta} T_{i,\beta} + \sum_{m=3}^{M} \lambda_{j,m} T_{i,m}) \tag{18}$$

となる．

したがって，クラウドレット全体の平均遅延時間が最も小さくなるような最適オフロード割合は，T_{priority} と T_{offload} のそれぞれが最も小さくなるような $\varphi_{ij}^*, i, j \in C$ によって求められる．

4. 数値解析

4.1 評価方法

本解析で用いる優先順位はそれぞれのジョブの平均処理時間に基づくものとする．M 個の異なるジョブの平均処理時間の大小関係が $b_1 < b_2 < \ldots < b_{M-1} < b_M$ で与えられたとき，平均処理時間の小さいジョブ 1 から優先的に処理が行われ，最後にジョブ M の処理が行われる．

本解析では，$N = 2$ のクラウドレット間で $M = 2$ 種類のジョブの間でのオフローディングについて評価する．各ジョブの平均処理時間を $b_1 = 0.6 * 10^{-4}$ [s/job]，$b_2 = 1.2 * 10^{-4}$ [s/job] とし，各ジョブの到着率を $\lambda_{11} = 6000, \lambda_{12} = \lambda_{21} = 2000$ [jobs/s] で固定させ，優先度の高いジョブの到着率を λ_{12} を 0 − 9000 [jobs/s] 増やした場合において，優先度を考慮した場合のクラウドレットの最小平均遅延時間 T_{priority} と，優先度を考慮せずオフロードのみ行う従来の手法による平均遅延時間 T_{offload}，オフロードも行わず優先度を考慮しない場合における平均遅延時間 $T_{\text{nooffload}}$ の 3 つと比較することで，提案手法の有効性を評価する．

4.2 解析結果

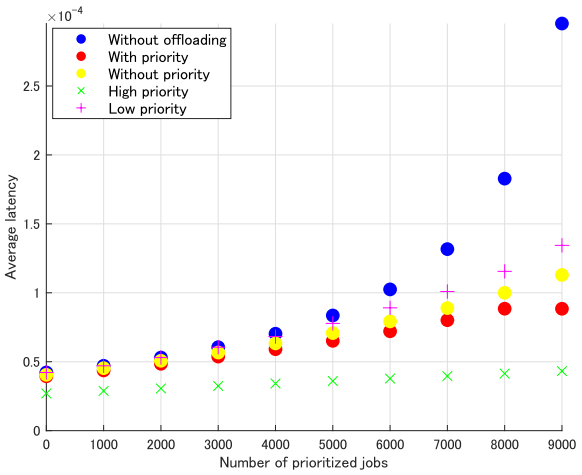

図 2　クラウドレット全体の平均遅延時間

図 2 に，優先度の高いジョブの量を変化させたときの各パターンにおけるクラウドレット全体の平均遅延時間の変化を示す．

横軸に示された優先度の高いジョブの増加の伴い，各パターンにおけるクラウドレットの平均遅延時間はおおよそ右肩上がりで増加している．特に，青い点で示された優先度も考慮せずオフロードも行わない場合だと，クラウドレット全体の平均遅延時間は著しく増加していくことが確認できる．これは，クラウドレット 1 の利用率が 1 に近づいているためである．このとき，緑色で示された優先度の高いジョブの増加度合いは他の遅延時間と比べると微小であり，優先的な処理によってジョブの増加にかかわらず短い遅延時間で処理が行われていることが分かる．一方で，紫色で示す優先度の低いジョブの遅延時間については，黄色で示された優先度を考慮しない場合と比較しても同程度まで遅延時間が増加してしまっている．これは，優先されたジョブの増加によって平均待ち時間が増加したためである．しかし，赤色で示す各優先度のジョブの割合を考慮した場合における重みつき平均の値は，優先度を考慮しない場合やオ

フロードを考慮しない場合と比較しても遅延時間が下回っており，提案手法の有効性が確認できた．

5. おわりに

本稿では，スマートシティの実現のために一般分布に従う処理時間を持つジョブがランダムに到着するようなクラウドレット環境を待ち行列理論を用いてモデル化し，ジョブの優先度を考慮しつつクラウドレットの平均遅延時間が最小化されるようなジョブのオフロード割合を決定する手法を提案した．結果として，全くオフロードしない場合と優先度を考慮しない場合の両方と比較して平均遅延時間を削減することができた．

今後は，クラウドレットの数やジョブの数を増やした場合においても同様に平均遅延時間が小さくなることを示し，本手法の拡張性について確認したい．また，優先順位の最適な決定方法や，他のクラウドレットの性能に大きく影響を与えうるオフロード割合の合理的な決め方についても，さらなる議論が必要だと考えられる．

謝　辞

本研究成果は，国立研究開発法人情報通信研究機構の委託研究（採択番号 JPJ012368C05601）により得られたものである．また，本研究は科研費（19K11947，22K12015，22H00247）の助成を受けたものである．

文　献

[1] C. Goumopoulos, “Smart city middleware: A survey and a conceptual framework,” *IEEE Access*, vol. 12, pp. 4015–4047, 2024.

[2] N. Tcholtchev and I. Schieferdecker, “Sustainable and reliable information and communication technology for resilient smart cities,” *Smart Cities*, vol. 4, no. 1, pp. 156–176, 2021. [Online]. Available: https://www.mdpi.com/2624-6511/4/1/9

[3] D. Puiu, P. Barnaghi, R. Tönjes, D. Kümper, M. I. Ali, A. Mileo, J. Xavier Parreira, M. Fischer, S. Kolozali, N. Farajidavar, F. Gao, T. Iggena, T.-L. Pham, C.-S. Nechifor, D. Puschmann, and J. Fernandes, “Citypulse: Large scale data analytics framework for smart cities,” *IEEE Access*, vol. 4, pp. 1086–1108, 2016.

[4] M. Khabbaz, “Deadline-constrained rsu-to-vehicle task offloading scheme for vehicular fog networks,” *IEEE Transactions on Vehicular Technology*, vol. 72, no. 11, pp. 14 955–14 961, 2023.

[5] D. Lymperis and C. Goumopoulos, “Sedia: A platform for semantically enriched iot data integration and development of smart city applications,” *Future Internet*, vol. 15, no. 8, 2023. [Online]. Available: https://www.mdpi.com/1999-5903/15/8/276

[6] K.-C. Chang, K.-C. Chu, H.-C. Wang, Y.-C. Lin, and J.-S. Pan, “Agent-based middleware framework using distributed cps for improving resource utilization in smart city,” *Future Generation Computer Systems*, vol. 108, pp. 445–453, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19334648

[7] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, “The case for vm-based cloudlets in mobile computing,” *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[8] K. Peng, X. Qian, B. Zhao, K. Zhang, and Y. Liu, “A new cloudlet placement method based on affinity propagation for cyber-physical-social systems in wireless metropolitan area networks,” *IEEE Access*, vol. 8, pp. 34 313–34 325, 2020.

[9] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, “Efficient algorithms for capacitated cloudlet placements,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 2866–2880, 2016.

[10] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, “All one needs to know about fog computing and related edge computing paradigms: A complete survey,” *Journal of Systems Architecture*, vol. 98, pp. 289–330, 2019.

[11] Y. Jiang, “A survey of task allocation and load balancing in distributed systems,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 585–599, 2016.

[12] Y. Yoktoa and S. Miyata, “Latency focused load balancing method between cloudlets using game theory,” vol. E-15N, no. 2, 2024.

[13] T. Fang, J. Chen, and Y. Zhang, “Content-aware multi-subtask offloading: A coalition formation game-theoretic approach,” *IEEE Communications Letters*, vol. 25, no. 8, pp. 2664–2668, 2021.

[14] Q. Fan and N. Ansari, “Cost aware cloudlet placement for big data processing at the edge,” pp. 1–6, 2017.

[15] H. Cao and J. Cai, “Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: A game-theoretic machine learning approach,” *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 752–764, 2018.

[16] Q. Fan and N. Ansari, “Application aware workload allocation for edge computing-based iot,” *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2146–2153, 2018.

[17] M. Jia, W. Liang, Z. Xu, and M. Huang, “Cloudlet load balancing in wireless metropolitan area networks,” pp. 1–9, 2016.

[18] D. Zhang, Y. Ma, C. Zheng, Y. Zhang, X. S. Hu, and D. Wang, “Cooperative-competitive task allocation in edge computing for delay-sensitive social sensing,” pp. 243–259, 2018.

[19] F. Varshosaz, M. Moazzami, B. Fani, and P. Siano, “Day-ahead capacity estimation and power management of a charging station based on queuing theory,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 10, pp. 5561–5574, 2019.