STRATEGI OFFLOADING TUGAS ADAPTIF BERBASIS HEURISTIK PADA EDGE COMPUTING IOT

Ata Amrullah, Mohammad Rifqi Dwi Ardiansyah, Fakhrul Rijal
Prodi Informatika, Universitas Islam Darul Ulum
Jln. Airlangga 03 Sukodadi Lamongan, Jawa Timur, Indonesia
ata@unisda.ac.id

ABSTRAK

Pertumbuhan pesat Internet of Things (IoT) menyebabkan peningkatan jumlah perangkat dan volume data yang harus diproses secara real-time dengan latensi rendah. Ketergantungan penuh pada cloud computing sering kali menimbulkan keterlambatan akibat keterbatasan bandwidth dan jarak geografis. Oleh karena itu, edge computing diperkenalkan untuk mendekatkan proses komputasi ke sumber data. Namun, keterbatasan sumber daya pada node edge menjadikan penentuan strategi offloading tugas sebagai permasalahan penting dalam menjaga kualitas layanan. Penelitian ini mengusulkan strategi offloading tugas adaptif berbasis heuristik dengan mempertimbangkan beberapa parameter sistem, yaitu latensi, utilisasi CPU, dan panjang antrean. Metode yang diusulkan dievaluasi melalui simulasi pada skenario layanan IoT real-time dan dibandingkan dengan pendekatan baseline static latency-only. Hasil simulasi menunjukkan bahwa strategi yang diusulkan mampu menurunkan latensi rata-rata sebesar 15–30% serta menghasilkan distribusi beban yang lebih seimbang pada node edge, terutama pada kondisi beban tinggi. Hasil ini menunjukkan bahwa pendekatan heuristik adaptif efektif dalam meningkatkan kinerja sistem edge computing untuk layanan IoT real-time.

Kata kunci : *Internet of Things, edge computing, task offloading, heuristik adaptif, latensi*

1. PENDAHULUAN

Internet of Things (IoT) menunjukkan perkembangan yang semakin pesat seiring meningkatnya pemanfaatan perangkat cerdas pada berbagai sektor, seperti layanan kesehatan, sistem perkotaan cerdas, dan industri berbasis otomasi. Penerapan IoT dalam skala besar berdampak pada bertambahnya jumlah perangkat yang secara kontinu menghasilkan data dan menuntut pemrosesan dengan waktu respons yang cepat. Pada aplikasi IoT yang bersifat real-time, misalnya pemantauan kondisi vital atau sistem kontrol berbasis sensor, keterlambatan pemrosesan data dapat berimplikasi langsung pada penurunan kualitas layanan serta keandalan sistem secara keseluruhan [1], [2]. Oleh sebab itu, ketersediaan mekanisme komputasi yang mampu mendukung latensi rendah dan pemrosesan yang responsif menjadi kebutuhan utama dalam perancangan sistem IoT modern [3].

Pada tahap awal pengembangan sistem IoT berskala besar, cloud computing banyak dimanfaatkan sebagai pusat pemrosesan data karena kemampuannya menyediakan sumber daya komputasi dan penyimpanan yang bersifat elastis. Meskipun demikian, pendekatan yang berorientasi penuh pada cloud menunjukkan sejumlah keterbatasan ketika diterapkan pada aplikasi IoT yang membutuhkan respons cepat. Faktor jarak geografis antara perangkat IoT dan pusat data cloud, keterbatasan bandwidth jaringan, serta fluktuasi kondisi jaringan dapat memicu peningkatan latensi end-to-end dan menurunkan stabilitas kualitas layanan [4], [5]. Beberapa penelitian terkini juga melaporkan bahwa ketergantungan penuh terhadap cloud computing belum mampu memenuhi kebutuhan latensi rendah pada layanan IoT real-time, terutama pada lingkungan dengan kepadatan perangkat tinggi dan pola trafik yang dinamis [6].

Sebagai upaya untuk mengatasi keterbatasan pendekatan cloud-sentris, paradigma edge computing diperkenalkan dengan menempatkan kapabilitas komputasi lebih dekat ke sumber data, yaitu pada sisi jaringan akses. Dengan memindahkan sebagian proses pemrosesan ke node edge, sistem IoT dapat mengurangi latensi transmisi, menekan beban pada jaringan backbone, serta meningkatkan keandalan layanan real-time [7]. Berbagai hasil penelitian terkini mengindikasikan bahwa penerapan edge computing mampu memberikan peningkatan performa yang signifikan pada aplikasi IoT yang sensitif terhadap latensi, khususnya pada skenario dengan jumlah perangkat besar dan beban kerja yang berfluktuasi [8], [9].

Meskipun demikian, node edge memiliki keterbatasan sumber daya jika dibandingkan dengan pusat data cloud. Kapasitas CPU, memori, serta kemampuan penanganan antrean pada node edge relatif terbatas dan berpotensi mengalami kelebihan beban ketika terjadi lonjakan jumlah tugas IoT [10]. Kondisi tersebut dapat menyebabkan peningkatan waktu tunggu pemrosesan dan berdampak pada penurunan kualitas layanan apabila tidak dikelola dengan mekanisme yang tepat. Oleh karena itu, pengelolaan sumber daya dan distribusi beban kerja menjadi aspek penting dalam implementasi sistem edge computing untuk IoT.

Salah satu mekanisme utama yang digunakan untuk mengatasi keterbatasan sumber daya pada node edge adalah task offloading, yaitu proses pengambilan keputusan terkait lokasi eksekusi tugas IoT, baik pada node edge tertentu maupun dengan meneruskannya ke

cloud. Keputusan offloading yang tepat berperan penting dalam menjaga kualitas layanan dan stabilitas sistem secara keseluruhan. Namun, dalam praktiknya, proses pengambilan keputusan offloading menjadi permasalahan yang kompleks karena harus mempertimbangkan berbagai faktor, seperti kondisi jaringan, beban komputasi, serta kapasitas antrean pada node edge [11]. Kesalahan dalam menentukan keputusan offloading dapat mengakibatkan ketidakseimbangan beban dan degradasi performa sistem.

Sebagian besar pendekatan task offloading yang telah diusulkan masih mengandalkan mekanisme keputusan statis atau hanya mempertimbangkan satu parameter utama, seperti latensi jaringan. Pendekatan static latency-only relatif sederhana dan ringan, tetapi kurang mampu beradaptasi terhadap perubahan kondisi sistem, terutama ketika utilisasi CPU dan panjang antrean meningkat secara dinamis [12]. Di sisi lain, beberapa penelitian mulai mengintegrasikan lebih dari satu parameter sistem dalam proses pengambilan keputusan, namun masih menggunakan bobot keputusan yang bersifat tetap sehingga fleksibilitas sistem tetap terbatas [13]. Pendekatan berbasis pembelajaran mesin juga telah diusulkan untuk meningkatkan adaptivitas keputusan, tetapi sering kali memiliki kompleksitas komputasi yang tinggi dan kurang sesuai untuk diterapkan pada node edge dengan sumber daya terbatas [14].

Berdasarkan tinjauan terhadap penelitian-penelitian tersebut, dapat diidentifikasi adanya celah penelitian dalam pengembangan strategi task offloading pada edge computing untuk IoT. Sebagian besar pendekatan yang ada belum mampu mengombinasikan karakteristik ringan, adaptif, dan multi-parameter dalam satu kerangka keputusan yang praktis. Secara khusus, masih terbatas penelitian yang secara eksplisit menyesuaikan bobot keputusan offloading berdasarkan kondisi beban sistem, seperti utilisasi CPU dan panjang antrean, secara dinamis.

Berdasarkan permasalahan tersebut, penelitian ini bertujuan untuk mengembangkan strategi offloading tugas adaptif berbasis heuristik pada lingkungan edge computing untuk sistem IoT. Strategi yang diusulkan mempertimbangkan beberapa parameter utama, yaitu latensi jaringan, utilisasi CPU, dan panjang antrean, dengan mekanisme penyesuaian bobot keputusan yang bersifat adaptif mengikuti kondisi sistem. Melalui pendekatan tersebut diharapkan sistem mampu melakukan distribusi beban kerja secara lebih seimbang serta meningkatkan kinerja layanan IoT yang sensitif terhadap latensi. Evaluasi terhadap strategi yang diusulkan dilakukan melalui simulasi untuk menganalisis performanya dibandingkan dengan pendekatan offloading konvensional yang hanya mempertimbangkan parameter latensi.

2. TINJAUAN PUSTAKA

Task offloading merupakan salah satu mekanisme penting dalam arsitektur edge computing yang berfungsi untuk menentukan lokasi eksekusi tugas IoT, baik pada node edge, fog, maupun cloud. Mekanisme ini bertujuan untuk menurunkan latensi end-to-end sekaligus menjaga kualitas layanan pada sistem IoT yang bersifat real-time. Sejumlah penelitian awal mengusulkan strategi offloading dengan menjadikan latensi jaringan sebagai parameter utama dalam proses pengambilan keputusan. Pendekatan tersebut dinilai cukup efektif pada kondisi beban sistem yang ringan, namun kinerjanya cenderung menurun ketika jumlah task meningkat dan node edge mulai mengalami kepadatan [2].

Penelitian selanjutnya mulai menyoroti keterbatasan pendekatan cloud-centric computing terhadap performa sistem IoT. Beberapa studi menunjukkan bahwa ketergantungan penuh pada cloud belum mampu memenuhi kebutuhan aplikasi yang sensitif terhadap latensi. Oleh karena itu, edge computing dipandang sebagai solusi yang lebih sesuai untuk mendukung pemrosesan IoT real-time dengan mendekatkan sumber daya komputasi ke perangkat pengguna [5].

Untuk meningkatkan efektivitas keputusan offloading, sejumlah penelitian mulai mengintegrasikan lebih dari satu parameter sistem, seperti latensi, utilisasi CPU, dan ketersediaan sumber daya komputasi. Pendekatan yang bersifat resource-aware ini bertujuan untuk mencegah terjadinya kelebihan beban pada node edge serta meningkatkan efisiensi pemrosesan secara keseluruhan [6]. Hasil-hasil penelitian tersebut menunjukkan bahwa integrasi beberapa parameter sistem mampu menghasilkan performa yang lebih stabil dibandingkan pendekatan latency-only..

Meskipun demikian, sebagian besar pendekatan multi-parameter yang diusulkan masih menggunakan bobot keputusan yang bersifat statis atau ditentukan secara empiris pada tahap awal simulasi. Penggunaan bobot statis menyebabkan sistem kurang mampu beradaptasi terhadap perubahan kondisi beban yang bersifat dinamis, seperti lonjakan utilisasi CPU dan peningkatan panjang antrean pemrosesan [8]. Akibatnya, performa sistem tetap berpotensi mengalami degradasi pada skenario dengan beban tinggi.

Pendekatan heuristik banyak digunakan dalam mekanisme task offloading karena memiliki kompleksitas komputasi yang relatif rendah dan mudah diimplementasikan pada node edge dengan sumber daya terbatas. Beberapa penelitian melaporkan bahwa algoritma heuristik mampu memberikan solusi yang mendekati optimal tanpa menimbulkan overhead komputasi yang signifikan [9]. Namun demikian, pendekatan heuristik konvensional umumnya belum dilengkapi dengan mekanisme adaptasi bobot keputusan yang bersifat dinamis, sehingga tingkat fleksibilitas sistem masih terbatas.

Sebagai alternatif, pendekatan berbasis pembelajaran mesin dan deep reinforcement learning mulai diusulkan untuk meningkatkan kualitas keputusan offloading. Pendekatan ini memungkinkan sistem untuk mempelajari pola kondisi lingkungan secara dinamis dan menyesuaikan keputusan berdasarkan hasil pembelajaran tersebut [10]. Meskipun memiliki keunggulan dari sisi adaptivitas, kompleksitas model dan kebutuhan komputasi yang tinggi menjadikan pendekatan ini kurang sesuai untuk diterapkan secara langsung pada node edge, khususnya pada skenario IoT real-time yang menuntut waktu respons cepat dan overhead rendah [11].

Berdasarkan analisis terhadap penelitian-penelitian terdahulu yang dirangkum pada Tabel 1, dapat disimpulkan bahwa masih terdapat keterbatasan dalam pengembangan strategi task offloading pada edge computing IoT. Sebagian besar pendekatan yang ada masih berfokus pada satu atau dua parameter sistem, menggunakan bobot keputusan yang bersifat statis, atau mengandalkan metode dengan kompleksitas komputasi yang tinggi. Selain itu, sebagaimana ditunjukkan pada Tabel 1, masih terbatas penelitian yang secara eksplisit mempertimbangkan panjang antrean sebagai indikator kepadatan node edge dalam mekanisme offloading adaptif [13].

Penelitian sebelumnya mengenai adaptive load balancing pada edge gateway menunjukkan bahwa mekanisme adaptif berperan penting dalam mendukung skalabilitas sistem IoT pada jaringan 5G [14]. Namun, pendekatan tersebut belum mengintegrasikan mekanisme keputusan task offloading berbasis multi-parameter dalam satu kerangka heuristik yang ringan. Oleh karena itu, masih terdapat celah penelitian dalam pengembangan strategi offloading tugas yang adaptif, ringan, dan mampu merespons dinamika beban sistem secara efektif.

Tabel 1. Research Gap

No	Peneliti	Tahun	Fokus Penelitian	Parameter Utama	Keterbatasan
1	Yang et al. [1]	2022	Task offloading MEC berbasis deep learning	Latensi, beban komputasi	Tidak mempertimbangkan panjang antrean
2	Shen et al. [2]	2022	Offloading MEC berbasis kerja sama IoV	Latensi jaringan	Terbatas pada skenario kendaraan
3	Gill et al. [3]	2024	Konsep, visi, dan tantangan komputasi modern	Konsep & arsitektur	Tidak membahas algoritma offloading
4	Ahmed et al. [4]	2025	Tantangan dan peluang fog computing pada jaringan IoT	Latensi, arsitektur fog, QoS	Tidak merumuskan model atau algoritma task offloading adaptif
5	Liu et al. [5]	2022	Offloading dan service caching MEC	Latensi, cache	Tidak adaptif terhadap beban dinamis
6	Kopras et al. [6]	2023	Alokasi komputasi dan komunikasi fog	Energi, latensi	Tidak merumuskan keputusan offloading
7	Markus et al. [7]	2023	Simulasi workflow IoT pada fog/edge	Kinerja sistem	Tidak mengusulkan strategi offloading
8	Alasmari et al. [8]	2023	Offloading hemat energi berbasis multi-classifier	Energi, latensi	Bobot keputusan bersifat statis
9	Shi et al. [9]	2023	Offloading vehicular edge berbasis DRL	Latensi, utilisasi resource	Kompleksitas komputasi tinggi
10	Bi & Zhao [10]	2024	Edge intelligence dua lapis untuk offloading	Latensi, kapasitas komputasi	Tidak mempertimbangkan panjang antrean
11	Benaboura et al. [11]	2023	Task offloading dan energi berbasis DQN	Latensi, energi	Kompleksitas komputasi tinggi
12	Awada et al. [12]	2023	Multi-task offloading	Resource usage, dependensi task	tidak menekankan mekanisme adaptasi bobot offloading
13	Acheampong et al. [13]	2023	Tinjauan komprehensif algoritma dan strategi task offloading	Latensi, energi, resource usage (survey)	Tidak mengusulkan metode baru
14	Amrullah [14]	2024	Adaptive load balancing pada edge gateway	Beban gateway, trafik	Belum membahas mekanisme offloading

3. METODOLOGI PENELITIAN

3.1. Arsitektur Sistem

Arsitektur sistem yang digunakan dalam penelitian ini terdiri dari tiga lapisan utama, yaitu perangkat IoT, node edge, dan cloud. Perangkat IoT berperan sebagai penghasil data dan tugas komputasi yang berasal dari sensor atau aplikasi pengguna. Tugas-tugas tersebut memiliki karakteristik berbeda, seperti ukuran data, kebutuhan komputasi, dan batas waktu pemrosesan (deadline), terutama pada layanan IoT real-time.

Node edge ditempatkan di dekat perangkat IoT dan bertanggung jawab untuk menangani sebagian besar proses komputasi guna menurunkan latensi transmisi. Namun, node edge memiliki keterbatasan sumber daya, seperti kapasitas CPU dan panjang antrean pemrosesan. Cloud berfungsi sebagai lapisan komputasi terakhir dengan sumber daya besar yang digunakan ketika node edge tidak mampu menangani beban kerja secara optimal.

Dalam arsitektur ini, setiap tugas IoT yang masuk akan dievaluasi oleh mekanisme task offloading untuk menentukan lokasi eksekusi yang paling sesuai, baik pada node edge tertentu maupun cloud, dengan tujuan meminimalkan latensi dan menjaga stabilitas sistem.

3.2. Model Task Offloading

Model task offloading dalam penelitian ini dirancang sebagai proses pengambilan keputusan berbasis multi-parameter. Untuk setiap tugas IoT yang masuk, sistem mengevaluasi kondisi beberapa kandidat node tujuan berdasarkan parameter berikut:

a. Latensi end-to-end (L), yang mencakup waktu transmisi data, waktu tunggu antrean, dan waktu eksekusi.
b. Utilisasi CPU (C) pada node edge, yang merepresentasikan tingkat beban komputasi saat ini.
c. Panjang antrean (Q), yang menunjukkan jumlah tugas yang sedang menunggu untuk diproses.

Ketiga parameter tersebut dipilih karena secara langsung memengaruhi kualitas layanan pada sistem IoT real-time dan mencerminkan kondisi beban node edge secara lebih komprehensif dibandingkan pendekatan latency-only.

3.3. Strategi Offloading Tugas Adaptif Berbasis Heuristik

Untuk menjawab keterbatasan pendekatan offloading existing yang menggunakan bobot statis atau parameter tunggal, penelitian ini mengusulkan strategi offloading tugas adaptif berbasis heuristik. Strategi ini menghitung skor keputusan untuk setiap kandidat node tujuan dengan mengombinasikan parameter latensi, utilisasi CPU, dan panjang antrean dalam satu fungsi evaluasi.

Skor keputusan untuk node ke-i dirumuskan sebagai:

$$Skor(i) = wL.L'(i) + wC.C'(i) + wQ.Q'(i) \quad (1)$$

dengan:

- $L'(i)$, $C'(i)$, dan $Q'(i)$ merupakan nilai parameter yang telah dinormalisasi,
- w_L, w_C, dan w_Q adalah bobot keputusan untuk masing-masing parameter.

Keunikan dari pendekatan ini terletak pada mekanisme adaptasi bobot. Ketika sistem berada pada kondisi beban ringan, bobot latensi lebih dominan. Sebaliknya, ketika utilisasi CPU atau panjang antrean meningkat, bobot untuk parameter tersebut diperbesar secara adaptif agar sistem menghindari node edge yang sedang padat. Dengan demikian, keputusan offloading tidak hanya berorientasi pada kedekatan jaringan, tetapi juga mempertimbangkan kondisi internal node edge.

3.4. Algoritma Heuristik Adaptif

Algoritma heuristik adaptif bekerja pada setiap kedatangan tugas IoT dengan langkah-langkah sebagai berikut:

a. Membaca status terkini setiap node kandidat, meliputi latensi estimasi, utilisasi CPU, dan panjang antrean.
b. Menghitung indikator beban sistem berdasarkan rata-rata utilisasi CPU dan panjang antrean.
c. Menyesuaikan bobot keputusan secara adaptif berdasarkan kondisi beban sistem.
d. Melakukan normalisasi parameter dan bobot.
e. Menghitung skor keputusan untuk setiap node kandidat.
f. Memilih node dengan skor terendah sebagai lokasi eksekusi tugas.

Pendekatan heuristik ini dipilih karena memiliki kompleksitas komputasi yang rendah, sehingga sesuai untuk diterapkan pada node edge dengan sumber daya terbatas dan mendukung kebutuhan respons cepat pada sistem IoT real-time.

3.5. Skenario dan Setup Simulasi

Evaluasi kinerja strategi offloading yang diusulkan dilakukan melalui simulasi pada skenario layanan IoT real-time. Sistem disimulasikan dengan sejumlah perangkat IoT yang menghasilkan tugas secara periodik dengan tingkat kedatangan yang bervariasi. Node edge dikonfigurasi dengan kapasitas CPU dan panjang antrean yang terbatas untuk merepresentasikan kondisi nyata pada lingkungan edge computing.

Sebagai pembanding, penelitian ini menggunakan metode static latency-only, yaitu pendekatan offloading konvensional yang menentukan lokasi eksekusi tugas berdasarkan latensi jaringan terendah tanpa mempertimbangkan utilisasi CPU dan panjang antrean. Kedua metode dievaluasi pada kondisi sistem yang identik untuk memastikan perbandingan kinerja yang adil.

3.6. Metrik Evaluasi

Kinerja strategi offloading dievaluasi menggunakan beberapa metrik utama, yaitu:

a. Latensi rata-rata end-to-end, untuk mengukur kualitas layanan IoT real-time.
b. Utilisasi CPU node edge, untuk mengevaluasi keseimbangan beban sistem.
c. Panjang antrean rata-rata, sebagai indikator kepadatan node edge.

Metrik-metrik tersebut digunakan untuk menilai sejauh mana strategi offloading adaptif berbasis heuristik mampu meningkatkan performa sistem dibandingkan pendekatan baseline.

3.7. Justifikasi Pemilihan Parameter Offloading

Strategi offloading tugas yang diusulkan dalam penelitian ini mempertimbangkan tiga parameter utama, yaitu latensi end-to-end, utilisasi CPU, dan panjang antrean. Pemilihan ketiga parameter tersebut

didasarkan pada karakteristik layanan IoT real-time serta hasil analisis penelitian terdahulu yang dirangkum pada Tabel 1.

Latensi end-to-end dipilih sebagai parameter utama karena secara langsung merepresentasikan kualitas layanan (Quality of Service/QoS) pada aplikasi IoT real-time. Keterlambatan pemrosesan data dapat berdampak signifikan terhadap keandalan sistem, terutama pada aplikasi seperti pemantauan kesehatan dan sistem kontrol berbasis sensor. Sebagian besar penelitian awal juga menjadikan latensi sebagai parameter utama dalam pengambilan keputusan offloading, meskipun masih bersifat statis atau tunggal.

Utilisasi CPU digunakan untuk merepresentasikan tingkat beban komputasi pada node edge. Parameter ini penting karena node edge memiliki sumber daya terbatas dan rentan mengalami kelebihan beban ketika jumlah tugas meningkat. Beberapa pendekatan existing mempertimbangkan utilisasi sumber daya, namun umumnya belum mengintegrasikannya secara adaptif dalam mekanisme keputusan offloading. Dengan memasukkan utilisasi CPU, strategi yang diusulkan mampu menghindari pemilihan node edge yang sedang berada pada kondisi beban tinggi.

Panjang antrean dipilih sebagai indikator kepadatan node edge yang secara langsung memengaruhi waktu tunggu pemrosesan tugas. Berbeda dengan latensi jaringan dan utilisasi CPU, panjang antrean mencerminkan kondisi internal sistem yang sering diabaikan pada pendekatan offloading konvensional. Integrasi parameter antrean memungkinkan sistem untuk mendistribusikan beban kerja secara lebih merata dan meningkatkan stabilitas pemrosesan pada kondisi beban dinamis.

Parameter konsumsi energi tidak dimasukkan dalam penelitian ini karena fokus utama diarahkan pada optimasi latensi dan stabilitas sistem IoT real-time. Selain itu, pengukuran energi yang akurat pada lingkungan edge membutuhkan model perangkat keras yang lebih kompleks dan spesifik. Pendekatan ini sejalan dengan tujuan penelitian untuk merancang strategi offloading yang ringan dan mudah diimplementasikan, dibandingkan metode berbasis pembelajaran mesin yang memiliki kompleksitas komputasi dan kebutuhan pelatihan yang tinggi.

3.8. Analisis Kompleksitas Algoritma

Algoritma offloading adaptif berbasis heuristik yang diusulkan dirancang dengan mempertimbangkan keterbatasan sumber daya pada node edge. Kompleksitas komputasi algoritma ini bersifat linear terhadap jumlah node kandidat, yaitu O(n), dengan n menyatakan jumlah node edge dan cloud yang tersedia sebagai tujuan offloading.

Pada setiap kedatangan tugas IoT, algoritma hanya melakukan operasi pembacaan status node, normalisasi parameter, penyesuaian bobot sederhana, serta perhitungan skor keputusan. Tidak terdapat proses iteratif yang kompleks maupun fase pelatihan model seperti pada pendekatan berbasis deep reinforcement learning. Dengan demikian, overhead komputasi yang dihasilkan relatif rendah dan sesuai untuk diterapkan pada node edge dengan kapasitas komputasi terbatas.

Jika dibandingkan dengan pendekatan berbasis pembelajaran mesin, seperti DQN-based task offloading, metode yang diusulkan tidak memerlukan proses training, state exploration, maupun reward optimization yang bersifat iteratif. Hal ini menjadikan strategi heuristik adaptif lebih stabil dan dapat memberikan respons keputusan secara cepat, yang merupakan kebutuhan utama pada sistem IoT real-time.

4. HASIL DAN PEMBAHASAN

4.1. Parameter Simulasi

Simulasi dilakukan pada skenario layanan IoT real-time dengan asumsi sejumlah perangkat IoT yang menghasilkan tugas pemrosesan data secara periodik. Node edge dikonfigurasikan dengan keterbatasan sumber daya untuk merepresentasikan kondisi nyata pada lingkungan edge computing. Parameter utama simulasi ditunjukkan pada Tabel 2.

Tabel 2. Parameter Simulasi

Parameter	Nilai
Jumlah perangkat IoT	100–500
Jumlah node edge	5
Kapasitas CPU node edge	2–4 GHz
Kapasitas antrean	50 task
Deadline task	100–300 ms
Metode pembanding	Static latency-only

4.2. Perbandingan Latensi Rata-rata

Latensi end-to-end digunakan sebagai metrik utama untuk mengevaluasi kinerja strategi offloading pada sistem IoT real-time. Tabel 3 menunjukkan perbandingan latensi rata-rata antara metode baseline static latency-only dan strategi offloading adaptif berbasis heuristik yang diusulkan.

Tabel 3. Latensi Rata-Rata End-to-End (ms)

Jumlah Task	Static Latency-only	HAWD (Usulan)
100	120	98
200	165	132
300	210	162
400	275	205
500	340	255

Berdasarkan Tabel 3, terlihat bahwa metode yang diusulkan secara konsisten menghasilkan latensi yang lebih rendah dibandingkan pendekatan baseline pada seluruh tingkat beban sistem. Penurunan latensi rata-rata berada pada kisaran 15–30%, dengan peningkatan kinerja yang semakin signifikan pada kondisi beban tinggi. Hal ini menunjukkan bahwa mekanisme adaptasi bobot mampu menghindari pemilihan node edge yang sedang padat, sehingga waktu tunggu

antrean dapat ditekan.

4.3. Analisis Utilisasi CPU Node Edge

Selain latensi, utilisasi CPU node edge dianalisis untuk mengevaluasi keseimbangan distribusi beban sistem. Tabel 4 menyajikan perbandingan utilisasi CPU rata-rata antara metode baseline dan metode usulan.

Tabel 4. Rata-rata Utilisasi CPU (%)

Jumlah Task	Static Latency-only	HAWD (Usulan)
100	42	38
200	68	55
300	85	70
400	95	78
500	99	82

Hasil pada Tabel 4 menunjukkan bahwa pendekatan static latency-only cenderung menyebabkan peningkatan utilisasi CPU yang tajam hingga mendekati kondisi jenuh pada beban tinggi. Sebaliknya, metode offloading adaptif berbasis heuristik menghasilkan utilisasi CPU yang lebih terkendali. Hal ini mengindikasikan bahwa integrasi parameter utilisasi CPU dan panjang antrean dalam keputusan offloading mampu mendistribusikan beban kerja secara lebih merata antar node edge.

4.4. Analisis Panjang Antrean

Panjang antrean dianalisis sebagai indikator kepadatan node edge dan stabilitas sistem. Pada metode baseline, peningkatan jumlah task menyebabkan akumulasi antrean yang signifikan pada node edge dengan latensi jaringan terendah. Sebaliknya, metode usulan mampu menjaga panjang antrean tetap lebih stabil dengan mengalihkan tugas ke node yang memiliki kondisi beban lebih ringan.

Secara kualitatif, mekanisme adaptasi bobot pada metode usulan mencegah terjadinya bottleneck pada node tertentu dan mengurangi risiko keterlambatan pemrosesan akibat antrean yang panjang. Temuan ini sejalan dengan tujuan penelitian, yaitu mendukung layanan IoT real-time pada kondisi beban sistem yang dinamis.

4.5. Pembahasan

Hasil simulasi menunjukkan bahwa strategi offloading tugas adaptif berbasis heuristik mampu mengatasi keterbatasan pendekatan offloading existing yang bersifat statis dan berorientasi pada satu parameter. Dengan mempertimbangkan latensi, utilisasi CPU, dan panjang antrean secara simultan, metode usulan menghasilkan keputusan offloading yang lebih responsif terhadap kondisi sistem.

Dibandingkan dengan pendekatan berbasis pembelajaran mesin yang memiliki kompleksitas tinggi, strategi heuristik adaptif yang diusulkan tetap ringan dan sesuai untuk diterapkan pada node edge dengan sumber daya terbatas. Dengan demikian, metode ini memberikan keseimbangan antara peningkatan performa dan efisiensi komputasi, sebagaimana dibutuhkan pada sistem IoT real-time.

4.6. Diskusi Tren Kinerja pada Kondisi Beban Dinamis

Berdasarkan hasil simulasi pada Tabel 3 dan Tabel 4, dapat diamati bahwa perbedaan kinerja antara metode static latency-only dan strategi offloading adaptif berbasis heuristik semakin meningkat seiring bertambahnya jumlah task IoT. Pada kondisi beban ringan, kedua metode menunjukkan kinerja yang relatif mendekati karena sebagian besar node edge masih memiliki kapasitas komputasi dan antrean yang memadai. Namun, ketika jumlah task meningkat, pendekatan static latency-only cenderung terus memilih node edge dengan latensi jaringan terendah tanpa mempertimbangkan kondisi internal node tersebut.

Akibatnya, node edge tertentu mengalami akumulasi beban yang berlebihan, yang ditunjukkan oleh peningkatan utilisasi CPU dan panjang antrean secara signifikan. Kondisi ini menyebabkan peningkatan waktu tunggu pemrosesan dan berdampak langsung pada kenaikan latensi end-to-end. Sebaliknya, strategi offloading adaptif berbasis heuristik mampu merespons kondisi tersebut dengan menyesuaikan bobot keputusan, sehingga tugas-tugas baru dialihkan ke node edge yang memiliki beban relatif lebih ringan. Mekanisme ini menjelaskan mengapa laju peningkatan latensi pada metode usulan lebih landai dibandingkan pendekatan baseline pada kondisi beban tinggi.

4.7. Analisis Keseimbangan Beban Sistem

Distribusi beban yang lebih merata pada metode usulan menunjukkan bahwa integrasi parameter utilisasi CPU dan panjang antrean berperan penting dalam menjaga stabilitas sistem. Dengan mempertimbangkan kedua parameter tersebut, keputusan offloading tidak hanya berorientasi pada kedekatan jaringan, tetapi juga pada kemampuan node edge dalam memproses tugas secara efektif. Hal ini mencegah terjadinya hotspot komputasi pada node tertentu yang sering muncul pada pendekatan berbasis latensi saja.

Hasil ini konsisten dengan tujuan penelitian, yaitu merancang strategi offloading yang adaptif namun tetap ringan. Berbeda dengan pendekatan berbasis pembelajaran mesin yang membutuhkan proses pelatihan dan komputasi tambahan, metode heuristik adaptif yang diusulkan mampu mencapai keseimbangan beban sistem melalui mekanisme sederhana yang mudah diimplementasikan. Dengan demikian, pendekatan ini lebih sesuai untuk lingkungan edge computing IoT yang memiliki keterbatasan sumber daya dan menuntut waktu respons cepat.

4.8. Perbandingan Konseptual dengan Penelitian Terkait

Jika dibandingkan dengan pendekatan berbasis deep reinforcement learning, seperti metode task offloading berbasis DQN, strategi yang diusulkan memiliki kompleksitas komputasi yang jauh lebih rendah. Pendekatan berbasis DQN umumnya memerlukan fase pelatihan, eksplorasi ruang status, serta optimasi fungsi reward yang bersifat iteratif. Meskipun mampu menghasilkan keputusan yang adaptif, kompleksitas tersebut berpotensi menimbulkan overhead komputasi yang signifikan pada node edge.

Di sisi lain, penelitian yang berfokus pada resource-aware atau dependency-aware offloading pada skenario tertentu, seperti Internet of Vehicles, menunjukkan peningkatan kinerja pada domain spesifik. Namun, pendekatan tersebut sering kali kurang fleksibel untuk diterapkan pada skenario IoT yang lebih umum. Strategi offloading adaptif berbasis heuristik yang diusulkan dalam penelitian ini bersifat lebih generik dan dapat diterapkan pada berbagai layanan IoT real-time tanpa bergantung pada karakteristik domain tertentu.

4.9. Implikasi terhadap Layanan IoT Real-Time

Hasil dan analisis yang diperoleh menunjukkan bahwa strategi offloading adaptif berbasis heuristik memiliki implikasi praktis yang signifikan bagi pengembangan layanan IoT real-time. Dengan latensi yang lebih rendah dan distribusi beban yang lebih seimbang, sistem edge computing dapat memberikan kualitas layanan yang lebih stabil, terutama pada kondisi lonjakan jumlah perangkat atau trafik data.

Pendekatan ini juga membuka peluang penerapan pada berbagai skenario IoT, seperti pemantauan kesehatan, sistem industri cerdas, dan aplikasi perkotaan cerdas, yang memerlukan pemrosesan data secara cepat dan andal. Dengan karakteristiknya yang ringan dan adaptif, strategi yang diusulkan dapat dijadikan sebagai solusi praktis untuk mendukung skalabilitas dan kinerja sistem IoT berbasis edge computing.

5. KESIMPULAN DAN SARAN

Penelitian ini mengusulkan strategi offloading tugas adaptif berbasis heuristik pada edge computing IoT dengan mempertimbangkan parameter latensi, utilisasi CPU, dan panjang antrean secara simultan. Pendekatan yang diusulkan dirancang untuk mengatasi keterbatasan strategi offloading existing yang umumnya bersifat statis atau hanya berfokus pada satu parameter keputusan.

Hasil simulasi pada skenario layanan IoT real-time menunjukkan bahwa metode yang diusulkan mampu menurunkan latensi rata-rata end-to-end sebesar 15–30% dibandingkan pendekatan static latency-only. Selain itu, strategi ini juga menghasilkan distribusi beban yang lebih seimbang pada node edge, yang ditunjukkan oleh penurunan utilisasi CPU dan stabilitas panjang antrean pada kondisi beban sistem yang tinggi. Temuan ini mengindikasikan bahwa mekanisme adaptasi bobot berbasis heuristik efektif dalam meningkatkan kualitas layanan dan stabilitas sistem edge computing.

Sebagai saran untuk penelitian selanjutnya, strategi offloading yang diusulkan dapat dikembangkan dengan mengintegrasikan mekanisme pembelajaran mesin untuk menyesuaikan bobot keputusan secara otomatis, serta dievaluasi pada skenario aplikasi IoT lain seperti smart transportation dan smart industry. Selain itu, pengujian pada lingkungan nyata atau testbed juga diperlukan untuk memvalidasi kinerja metode ini di luar skenario simulasi.

DAFTAR PUSTAKA

[1] S. Yang, G. Lee, and L. Huang, “Deep Learning-Based Dynamic Computation Task Offloading for Mobile Edge Computing Networks,” *Sensors*, vol. 22, no. 11, 2022, doi: 10.3390/s22114088.

[2] X. Shen, Z. Chang, and S. Niu, “Mobile Edge Computing Task Offloading Strategy Based on Parking Cooperation in the Internet of Vehicles,” *Sensors*, vol. 22, no. 13, 2022, doi: 10.3390/s22134959.

[3] S. S. Gill *et al.*, “Modern computing: Vision and challenges,” *Telematics and Informatics Reports*, vol. 13, p. 100116, 2024, doi: https://doi.org/10.1016/j.teler.2024.100116.

[4] Z. R. Ahmed, S. Askar, D. H. Hussein, and M. A. Ibrahim, “Fog Computing Challenges and Opportunities in IoT Networks: A Review,” *Procedia Comput. Sci.*, vol. 259, pp. 1749–1764, 2025, doi: https://doi.org/10.1016/j.procs.2025.04.130.

[5] X. Liu, X. Zhao, G. Liu, F. Huang, T. Huang, and Y. Wu, “Collaborative Task Offloading and Service Caching Strategy for Mobile Edge Computing,” *Sensors*, vol. 22, no. 18, 2022, doi: 10.3390/s22186760.

[6] B. Kopras, F. Idzikowski, B. Bossy, P. Kryszkiewicz, and H. Bogucka, “Communication and Computing Task Allocation for Energy-Efficient Fog Networks,” *Sensors*, vol. 23, no. 2, 2023, doi: 10.3390/s23020997.

[7] A. Markus, A. Al-Haboobi, G. Kecskemeti, and A. Kertesz, “Simulating IoT Workflows in DISSECT-CF-Fog,” *Sensors*, vol. 23, no. 3, 2023, doi: 10.3390/s23031294.

[8] M. K. Alasmari, S. S. Alwakeel, and Y. A. Alohali, “A Multi-Classifiers Based Algorithm for Energy Efficient Tasks Offloading in Fog Computing,” *Sensors*, vol. 23, no. 16, 2023, doi: 10.3390/s23167209.

[9] W. Shi, L. Chen, and X. Zhu, “Task Offloading Decision-Making Algorithm for Vehicular Edge Computing: A Deep-Reinforcement-Learning-Based Approach,” *Sensors*, vol. 23, no. 17, 2023,

doi: 10.3390/s23177595.

[10] X. Bi and L. Zhao, "Two-Layer Edge Intelligence for Task Offloading and Computing Capacity Allocation with UAV Assistance in Vehicular Networks," *Sensors*, vol. 24, no. 6, 2024, doi: 10.3390/s24061863.

[11] A. Benaboura, R. Bechar, W. Kadri, T. D. Ho, Z. Pan, and S. Sahmoud, "Latency-Aware and Energy-Efficient Task Offloading in IoT and Cloud Systems with DQN Learning," *Electronics (Basel).*, vol. 14, no. 15, 2025, doi: 10.3390/electronics14153090.

[12] U. Awada, J. Zhang, S. Chen, S. Li, and S. Yang, "Resource-aware multi-task offloading and dependency-aware scheduling for integrated edge-enabled IoV," *Journal of Systems Architecture*, vol. 141, p. 102923, 2023, doi: https://doi.org/10.1016/j.sysarc.2023.102923.

[13] A. Acheampong, Y. Zhang, X. Xu, and D. A. Kumah, "A Review of the Current Task Offloading Algorithms, Strategies and Approach in Edge Computing Systems," *CMES - Computer Modeling in Engineering and Sciences*, vol. 134, no. 1, pp. 35–88, 2022, doi: https://doi.org/10.32604/cmes.2022.021394.

[14] A. Amrullah, "Adaptive Load Balancing Model on Edge Gateways to Support IoT Scalability in 5G Networks," *Intellithings Journal*, vol. 1, no. 1, Feb. 2024, [Online]. Available: https://e-jurnal.unisda.ac.id/index.php/intellithings/article/view/10657