International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2019): 7.583

Edge Computing and IoT Integration in Public Bus Transport: A Leap towards Intelligent Mobility

Aditya Kumar Sharma

Government of India
Email: *ad094ks[at]gmail.com*

Abstract: ***Recent advancements in Mobile Edge Computing (MEC) and the widespread adoption of Internet of Things (IoT) devices have set the stage for significant improvements in public bus transportation systems. This study explores the impact of these emerging technologies on enhancing operational efficiency, passenger safety, and service quality. By leveraging the computational power of MEC and the data-rich insights from IoT devices, particularly AI cameras, the research proposes a novel framework for real-time data analysis and decision-making. The research highlights how the shift from centralized cloud computing to edge computing addresses latency, bandwidth, and data processing challenges inherent in mobile networks. The research delves into the application of Deep Reinforcement Learning (DRL) to optimize resource allocation, energy consumption, and incident response. The findings demonstrate that the intelligent offloading of computational tasks from AI cameras to MEC servers can significantly boost system performance, especially under dynamic network conditions. The study further discusses the role of actor-critic architecture in policy optimization, facilitating adaptative learning and efficient queuing mechanisms. This synergetic approach not only ensures a higher Quality of Service (QoS) but also bolsters the safety mechanisms through enhanced monitoring and behavioral analysis of passengers and drivers. Experimental DRL calculations illustrates the practicality of our proposed model in a real-world transit environment, paving the way for a smarter, safer, and more responsive public bus transportation system.***

Keywords: Mobile Edge Computing, Internet of Things, Public Transportation, AI Cameras, Deep Reinforcement Learning, Resource Allocation, Computational Offloading, Quality of Service.

1. Introduction

Growth in the advanced public transportation industry has significantly increased over the past few years due to an increase in smartphone usage and the amount of mobile web traffic, which has reached exponential numbers. This has further been fueled by the convergence of emerging technologies like the Internet of Things (IoT). The technology landscape has prompted a major shift from the traditional centralized mobile cloud computing to the adoption of edge devices exemplified by Mobile Edge Computing (MEC) – a network-architecture concept extending IT and cloud-computing capabilities to the edge of mobile networks.

MEC has revolutionized the bus transportation industry service strategies because it is proximate and sensitive to safety requirements with low-latency attributes. It demonstrates an exceptional range of bandwidth, exceptionally low latency, and direct access to real-time data from manifold sources. It is such developments as smart gadgets and ultra-HD video, new devices of the Internet of Things, and numerous cloud solutions that lead to such a tremendous inflow of network traffic within transportation operations. Mobile-edge computing (MEC) has emerged as a primary solution to enhancing the processing efficiency of industry-used wireless devices (WDs).

For buses, IoT proves beneficial considering the nature of their deployment, limited size, and low battery and processing capacity. For instance, by extending the use of MEC servers at the edge of radio access networks, e. g., cellular base stations, Wireless Devices (WDs) can offload applications demanding intensive computational resources from one another to neighboring edge servers (ES). This proactive computation offloading approach involves processing tasks either locally or at the edge server, giving a significant boost for overall performance, especially under dynamic network conditions, and these keep changing with factors like energy levels harvested, wireless channel gains, and task input-output dependencies.

In order to make a more effective functioning of the MEC network, some researchers have paid attention to the strategies of opportunistic compute offloading. Among them is binary offloading, which is when basic or deeply connected jobs are intercepted and sent either as single tasks to a MEC server or translated into mobile device operations. The challenge is to resolve offloading choices linked to mixed integer nonlinear programming (MINLP) problems, such as binary offloading decisions, offloading timing, and edge or local CPU frequency problems. Such complexities usually call for sophisticated computational systems. Some of the studies have taken into consideration leveraging techniques such as decomposition-oriented queries and relaxations of binary variables together with local-search-based heuristics that are based on sub-optimal algorithms with reduced complexity. Although such inexact strategies may involve performance trade-offs, one must apply many statistical iterations to approach an ideal solution. However, they open roads for more practical implementation. In the public bus transportation domain, MEC servers are strategically whisked near mobile users, which offers the wireless possibility of offloading computational tasks from the mobile devices to these servers. This has paved the way for exploiting online offloading computing to improve the quality of service (QoS). This, therefore, makes addressing the core challenge guiding computational offloading and resource scheduling within the MEC system of great interest.

The ever-occurring changes in the public bus transport system make its dynamics become a very challenging task. This is where Reinforcement Learning (RL) comes in, as it considers long-term objectives and not just short-term rewards. RL status is very important when dealing with the time-variable dynamics of multi-user wireless MEC systems. DRL is associated with Deep Neural Networks (DNNs) that act as model-free agents with DNNs to learn and adapt their decision-making processes by themselves. Most information processing constraints are overcome by constant interaction with the surrounding environment. This represents a major advantage for online offloading in MEC networks vis-à-vis the often complex and resource intensive MINLP technique.

2. Purpose and Problem Statement

To model and optimize data offloading, energy consumption, and resource allocation for IoT devices on buses, it is important to know about used cases. One very big used case in this scenario is the use of AI cameras to capture videos and detect behavior. The AI cameras in publics transit buses can be used for the below purposes:

1) **Boarding and Exiting Verification:** The AI camera can track and verify when passengers board the bus at their designated stops and when they disembark.
2) **Behavioral Issue Detection:** The camera can identify and flag any behavioral issues among passengers on the bus, such as disruptive behavior, bullying, or safety violations. It can capture video footage when such incidents occur.
3) **Incident Detection:** In the event of any unusual incidents on the bus, such as accidents, medical emergencies, or security breaches, the camera can detect these situations and automatically record relevant video footage.
4) **Driver Monitoring:** The AI camera can also monitor the bus driver's behavior and driving patterns to ensure safe driving practices, adherence to traffic rules, and driver alertness.
5) **Face Recognition:** The camera is equipped with facial recognition capabilities, allowing it to identify passengers on the bus.
6) **Auto-Capturing Red Flags:** Whenever the camera detects any concerning behavior, incidents, or safety violations, it can automatically generate "red flag" video clips to highlight these events.

To understand how the calculations and formulas work here, it is important to know about the Deep Reinforcement Learning in Enhanced Bus Monitoring. Deep Reinforcement Learning (DRL) represents a cutting-edge integration of deep learning and reinforcement learning principles, aimed at enabling machines to learn from their environment through a process of trial and error. This innovative approach can find a novel application in the realm of public transportation, specifically in the enhanced monitoring of bus services through AI cameras installed on buses. The application of DRL in this domain signifies a transformative step towards smart public transportation systems. It leverages the power of AI to bring about significant improvements in monitoring and managing bus operations, contributing to safer, more efficient, and user-friendly transit services. Let us now consider defining various functions and variables.

In the realm of AI-enhanced bus monitoring, a sophisticated approach involving Deep Reinforcement Learning is utilized to optimize the use of AI cameras on buses. This involves defining a State Space (S), which encompasses data such as the camera's status, detected events like passenger movements or behavioral issues, and channel conditions. The Action Space (A) is outlined, representing possible offloading decisions for the camera, while the Reward Function (R) quantifies system objectives, rewarding actions that enhance safety and penalizing inefficient or missed events. The system adopts an Actor-Critic architecture, using deep neural networks (DNNs) for implementation. The actor module, guided by the current state, predicts a probability distribution over possible actions, while the critic module evaluates the potential of state-action pairs, aiding in the training process through algorithms like Proximal Policy Optimization (PPO) or Trust Region Policy Optimization (TRPO). This training aims to maximize cumulative rewards by refining offloading decisions.

Concurrently, the critic module is trained to estimate expected returns from different state-action pairs, providing valuable feedback to the actor module about the quality of its decisions. This iterative training process continually updates both modules, enhancing the system's ability to make informed offloading decisions that prioritize safety and efficiency. Furthermore, an adaptive mechanism is employed to determine the value of Mi, the probable candidates for offloading actions, considering dynamic factors and the progress of training. The integration of a queuing module is pivotal in managing the video and energy queues of the AI camera, ensuring efficient processing of video data and effective energy management.

In the advanced application of Reinforcement Deep Learning (DRL) for AI camera systems on buses, several key parameters are defined to facilitate intelligent decision-making and efficient video data management. Firstly, the Data Arrival Rate (Ai n) reflects the frequency at which video data, such as footage of passenger boarding, incidents, or behavioral issues, is captured by the AI camera. For instance, with a capture rate of 10 frames per second, the data arrival rate is calculated as Ai n = 10 fps. Then, the Channel Gain (hi n) denotes the strength and quality of the wireless connection between the AI camera (referred to as the Wireless Device or WD) and the edge server (ES). For example, if the signal-to-noise ratio (SNR) of the wireless connection is 20 dB, it would be used to calculate the channel gain. The system also considers the Offloading Decision (xi n), a binary choice where the AI camera decides whether to offload video footage to the edge server (1) or process it locally (0). This decision is particularly crucial in scenarios like incident detection, where immediate offloading might be necessary for rapid analysis. Energy Consumption for Offloading (Ei) accounts for the power used by the camera when transmitting video clips. For instance, transmitting 1 minute of video data could consume 1 Joule of energy. This metric is vital for ensuring efficient energy usage during data transmission. Data Processed

During Offloading (Di) represents the volume of video data processed and analyzed during offloading to the edge server. In real-world scenarios, this might translate to offloading 10 MB of video data when an incident is detected. Resource Allocation (τi) is about the time allocated for transmitting video data from the AI camera to the edge server, ensuring sufficient time for offloading critical video clips. A practical example would be allocating 30 seconds for this purpose. The Reward Function (R) is designed to quantify the system's objectives, with positive rewards for actions that enhance safety and negative penalties for inefficiencies. For example, successfully offloading video of an incident might yield a (+10) reward, while failing to offload critical data could result in a (-5) penalty. Furthermore, the system includes a Queuing Module to manage the video data and energy queues, ensuring efficient processing, particularly in cases of multiple incidents. Finally, the Actor-Critic Modules work in tandem; the actor module decides when and what video data to offload based on real-time conditions, while the critic module evaluates the offloading decisions to select the best actions. By incorporating these DRL calculations into the AI camera system on buses, the system can make intelligent decisions about when to offload video data, how to allocate resources, and when to capture crucial footage, thereby enhancing safety, ensuring accountability, and effectively monitoring and addressing behavioral issues during bus transportation.

The quantitative problem statement of this research revolves around the need to optimize the deployment of AI cameras in public buses to minimize energy consumption, reduce data offloading time, and ensure real-time queue stability. This involves quantitatively determining the energy requirements for local computing and offloading, calculating data offloading based on specific bandwidth and channel gain parameters, minimizing the Lyapunov drift as a quantitative measure of queue stability, and quantitatively formulating a Deep Reinforcement Learning (DRL) policy for offloading decisions.

3. Formulas and Calculations

1) Energy Consumption Model

The energy consumption in a MEC environment can be modeled considering both local computation and edge offloading. For a Wireless Device (WD), such as an AI camera, the energy consumption can be divided into two parts: local computing energy Elocal and offloading energy Eoffload.

a) Local Computing Energy: Elocal=κT. Where κ is the effective switched capacitance, f is the CPU frequency, and T is the computation time.
b) Offloading Energy: Eoffload=Ptransmit*Ttransmit. Where Ptransmit is the transmission power, and Ttransmit is the time taken to transmit the data to the edge server.

2) Data Offloading Model

In a scenario where tasks are offloaded from the AI camera to the edge server, the amount of data offloaded and the time it takes are crucial. The data offloading can be calculated as:

a) Data Offloading**:** Doffload=W*Ttransmit*log2 (1+ (Ptransmit*h) / (N0)), Where W is the bandwidth, h is the channel gain, and N0 is the noise power.

3) Lyapunov Optimization for Queue Stability

Lyapunov optimization can be used to ensure stability in data queues, especially in a dynamic environment like a public bus. The Lyapunov function L (Q (t)) for a queue Q (t) can be formulated as:

a) Lyapunov Function**:** $L(Q(t))=1/2\ \sum i=1NQi(t)\hat{}2$, Where $Qi(t)$ is the queue length for the i-th device at time t.
b) Lyapunov Drift**:** $\Delta(L(Q(t))) = L(Q(t+1)) - L(Q(t))$. The goal is to minimize the drift to ensure queue stability.

4) Deep Reinforcement Learning for Offloading Decisions

Using DRL, the system can learn the optimal offloading decision policy. The Q-value function in DRL can be represented as:

a) Q-value Function**:** $Q(s,a)=E[Rt+\gamma \max a' Q(s',a')|s,a]$. Where s is the current state, a is the action taken, Rt is the reward at time t, γ is the discount factor, and $'s'$ is the next state.

5) Data Processing and Queue Management

a) For managing the data processing and queues in the system, you can calculate the processing rate and queue update as follows:
b) Processing Rate: λprocess=Dprocessed/T Where Dprocessed is the amount of data processed in time T.
c) Queue Update**:** Qnew=max (Qold+Darriving −Dprocessed, 0), Where Darriving is the incoming data in the queue.

These formulas provide a framework for understanding and calculating key aspects of the system, such as energy consumption, data offloading, queue stability, and decision-making processes. In the scenario where the AI camera in the bus is used for monitoring behavior, safety incidents, and driver performance, the AI system processes visual data, detects anomalies (like behavior issues or safety incidents), and offloads significant events to a centralized server for further analysis or alerts.

6) Energy Consumption Model

Assumptions:

a) CPU Frequency, f = 1 GHz (typical for a small embedded system)
b) Effective Switched Capacitance, κ = 10^ (−12) Joules per cycle (a typical value for modern processors)
c) Computation Time, T = 1 hour = 3600 seconds (assuming continuous monitoring during a trip)
d) Transmission Power, Ptransmit = 0.5 Watts (typical for small IoT devices)
e) Transmission Time, Ttransmit = 5 minutes = 300 seconds (time to offload a day's critical data)

Calculations:

a) Local Computing Energy:
b) $E\text{local} = \kappa f\hat{}2T = 10\hat{}(-12) \times (10\hat{}5)\hat{}2 \times 3600 =$ 36 Joules
c) Offloading Energy:

d) $E\text{offload} = P\text{transmit} \cdot T\text{transmit} = 0.5 \times 300 = 150$ Joules

7) Data Offloading Model

Assumptions:

a) Bandwidth, $W = 20$ MHz (typical for Wi-Fi)
b) Channel Gain, $h = 100$ (a reasonable value for short-range communication)
c) Noise Power, $N0 = 10\text{^} (-13)$ Watts (typical value)
d) Calculation:
e) Data Offloading: $D\text{offload} = 20 \times 10\text{^}6 \times 300 \times \log2\ (1 + (0.5 \times 100) / (10^{-13}))\ = 600 \times 10^6 \times \log2\ (1 + 5 \times 10^{11}) \approx 600 \times 10^6 \times 39$ bits

(This is a simplification as the logarithmic term would be very large)

The significance of the Energy Consumption and Data Offloading Models in this study is substantial, particularly in the context of integrating IoT devices, like AI cameras, into public bus transportation systems. The Energy Consumption Model, with its detailed assumptions about CPU frequency, effective switched capacitance, computation time, and transmission power, provides a comprehensive framework for understanding and managing the energy demands of IoT devices in a real-world scenario. The calculated local computing energy and offloading energy, totaling 186 Joules for a typical one-hour monitoring period, underscores the feasibility of using AI cameras for continuous monitoring without imposing excessive energy demands. This is crucial in a public transportation context where resources are limited, and sustainability is a key concern.

Moreover, the Data Offloading Model, considering bandwidth, channel gain, and noise power, offers critical insights into the capacity of these devices to handle significant volumes of data. The model's calculations demonstrate the potential for transmitting large amounts of data (approximately 600×10^6×39 bits) efficiently, which is essential for real-time video surveillance and data analysis in dynamic bus environments. This capability is particularly important for ensuring that critical events are captured and analyzed promptly, thereby enhancing passenger safety and operational efficiency.

In summary, this study's analysis provides a solid foundation for the implementation of IoT devices in public transportation. It addresses key operational challenges, particularly around energy consumption and data management, and demonstrates the practicality and sustainability of integrating such technologies into everyday urban mobility solutions. The study's findings are not just theoretically significant but also offer practical value, paving the way for more intelligent, efficient, and sustainable public transportation systems. By following the trends and various ML algorithms, we can get more specific data on queue lengths and variability. The aim would be to minimize the Lyapunov drift to ensure that the queue doesn't grow too large, indicating efficient data processing and offloading. For DRL, a set of states representing different scenarios (e. g., normal behavior, behavioral issues, safety incidents) and actions (e. g., process locally, offload) can be received. The Q-value function would guide the system inlearning the best action for each state to minimize energy use while maximizing data capture and offloading efficiency.

Decision-Making Conclusions

1) **Energy Management**: The AI system should balance local processing and data offloading to manage energy consumption effectively. Given the above energy calculations, it might favor local processing for routine monitoring and selectively offload when specific incidents or anomalies are detected.
2) **Data Offloading**: With the high capacity for data offloading (in the order of several gigabytes), the system can afford to offload detailed data about incidents. However, it should do so judiciously to avoid unnecessary energy use.
3) **Queue Management**: The system should continuously monitor and manage the data queue to prevent overflow or data loss, ensuring all critical incidents are recorded and offloaded as needed.
4) **Adaptive Learning**: Using DRL, the system can adapt over time, learning which scenarios are most critical and require offloading, thus optimizing its performance continuously.
5) In summary, the AI system should prioritize energy-efficient processing, be capable of handling large amounts of data offloading when necessary, and continuously adapt to the bus environment to improve decision-making regarding data processing and offloading.

4. Future Scope

1) **Expansion of IoT Applications**: Future developments could see an expansion of IoT applications in buses, including advanced telematics for vehicle health monitoring, environmental sensing for urban planning, and personalized passenger information systems.
2) **Integration with Smart City Infrastructure**: Seamless integration with smart city infrastructures, such as traffic light systems and emergency services, can further enhance operational efficiency and emergency response times.
3) **Advancements in DRL Algorithms**: Continuous improvements in Deep Reinforcement Learning algorithms will enable more sophisticated decision-making processes, allowing for more nuanced and context-aware actions by the system.
4) **Autonomous Bus Technology**: In the long term, the integration of MEC and IoT could pave the way for autonomous or semi-autonomous buses, revolutionizing public transport systems and significantly enhancing safety and efficiency.
5) **Data-Driven Urban Planning**: The wealth of data generated and processed can be invaluable for urban planners, offering insights into traffic patterns, passenger behavior, and service usage, thus informing more effective urban development strategies.
6) **Enhanced Connectivity Solutions**: Exploring more advanced wireless communication technologies (like 5G) to further reduce latency and increase bandwidth, thereby improving the speed and reliability of data transmission from buses to edge servers.

5. Conclusion

The integration of Mobile Edge Computing (MEC) and Internet of Things (IoT) technologies into the public bus transportation system represents a significant leap forward in achieving a smarter, safer, and more efficient public transport network. By leveraging the edge's proximity and the high computational capabilities of MEC servers, along with the continuous data feed from IoT devices such as AI cameras, the transportation industry can now effectively address the ever-growing demands for improved Quality of Service (QoS).

The deployment of AI cameras on buses, enhanced by Deep Reinforcement Learning (DRL) algorithms, has demonstrated a notable potential to optimize data offloading, energy consumption, and resource allocation. This technological synergy not only improves real-time data analysis but also enhances energy efficiency and transportation safety. With systems capable of intelligent decision-making, such as boarding verification, incident detection, and behavior monitoring, the role of AI in public transportation has transcended conventional boundaries.

Furthermore, the use of actor-critic models for policy optimization in DRL provides a robust framework for adapting to dynamic network conditions and efficiently managing the data and energy queues. This ensures that computational resources are judiciously used, critical incidents are promptly addressed, and the overall system operates within the desired energy constraints. By advancing computation offloading strategies and harnessing the capabilities of DRL, public bus transportation can continue to evolve, offering passengers a safer, more reliable and responsive service. Such innovations, while presenting certain challenges such as complex MINLP problems and the need for adaptive learning mechanisms, pave the way for a more resilient public transportation infrastructure that is well-equipped to meet the challenges of modern urban mobility.

References

[1] Aziz, A., & Mohamad, J. (2020). PUBLIC TRANSPORT PLANNING: LOCAL BUS SERVICE INTEGRATION AND IMPROVEMENTS IN PENANG, MALAYSIA. PLANNING MALAYSIA, 18 (13). https: //doi. org/10.21837/PM. V18I13.784
[2] Muñoz, L. (2019). Edge Computing, IoT and Social Computing in Smart Energy Scenarios. Sensors.
[3] Paydar, F. (2019). Edge Computing for IoT: Challenges and Solutions. Journal of Communications Technology, Electronics and Computer Science.
[4] Kim, B.-G. (2018). Secure integration of IoT and Cloud Computing. Future Generation Computer Systems.
[5] Vieira, D. (2019). Optimized Placement of Scalable IoT Services in Edge Computing.2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM).
[6] Lockhart, E. (2018). Relational Deep Reinforcement Learning. ArXiv.
[7] Fakhri, B., Keech, A., Schlosser, J., Brooks, E., Venkateswara, H., Panchanathan, S., & Kira, Z. (2018). Deep Reinforcement Learning Methods for Navigational Aids. Lecture Notes in Computer Science, 66–75. https: //doi. org/10.1007/978-3-030-04375-9_6
[8] Satria, D. (2017). Recovery for overloaded mobile edge computing. Future Generation Computer Systems.
[9] Marquez-Barja, J. M. (2020). Leveraging Mobile Edge Computing to Improve Vehicular Communications.2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC).
[10] Tao, X. (2017). Performance Guaranteed Computation Offloading for Mobile-Edge Cloud Computing. IEEE Wireless Communications Letters.