# Edge Computing for Aviation Autonomy

Moustafa Abdelbaky[1,2], Jiasi Chen[3], Abraham K. Ishihara[1,2], Carlee Joe-Wong[4] and Sandeep D. Shetye[1]

[1]NASA Ames Research Center, Moffett Field, CA
[2]KBR at Moffett Field, CA (current)
[3]Computer Science and Engineering, University of California, Riverside, Riverside, CA
[4]Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA

Abstract

Advanced Air Mobility is a new aviation vision where unmanned aerial systems will transport passengers and cargo across urban and rural areas. Critical to the realization of this vision is the development of edge computing infrastructure which allows service providers (e.g., data and reasoning services) operating in the airspace ecosystem to deliver data and reasoning insights to vehicles in real-time. In this paper, we present the architecture of a decentralized edge computing platform that connects data and reasoning service providers to vehicles and other service consumers.

## 1 Introduction

We are at the cusp of an aviation revolution whereby flying cyber-physical systems will disrupt and transform entire industries. The convergence of new technologies including electric propulsion and autonomy together with new business models is generating the potential for a new aviation market known as Advanced Air Mobility (AAM). AAM is a safe and efficient system for air passenger and cargo transportation across urban and rural areas, inclusive of small package delivery, Unmanned Aerial Vehicles (UAV), and other urban Unmanned Aerial Systems (UAS), which supports a mix of on-board/ground-piloted and increasingly autonomous operations.

Existing cloud platforms are not designed for the projected scale of geographically dispersed, heterogeneous entities with low-latency requirements that AAM needs. Similarly, they cannot support common and ubiquitous closed feedback loops, which are necessary to ensure the execution of safety-critical operations to support future automated airspace systems. Finally, these platforms do not foster innovation and data sharing at the scale required to meet advanced air mobility challenges.

As a result, critical to the success of AAM is the development of a decentralizes edge platform connecting vehicles, reasoning, and data services to enable real-time and non-real-time decision-making by all users (humans and machines) of the airspace system. Over the next several decades, high density operations will become prevalent in the urban setting motivating the need for more efficient modes of operation leveraging not only traditional airspace assets and strategies, but also emerging ones in autonomous ground transportation and smart cities.

The Data and Reasoning Fabric (DRF ) project aims to meet these challenges by employing a decentralized data marketplace, which connects data and reasoning service providers, edge providers, vehicles, and other service consumers. DRF aims to retain current levels of safety even with increased air travel density, complexity, and user communities while ensuring interoperability and security across the cloud-to-edge continuum. In particular, DRF is an open and scalable framework delivering the interfaces, protocols, tools, and unifying software architecture to connect nodes across vehicles, edge and cloud infrastructure to seamlessly work together, exchanging data as well as reasoning services for real-time and non-real-time decision-making by all users of the airspace system.

In this paper, we present the architecture of the DRF Edge component and outline how it can support use case scenarios for aviation autonomy. The remainder of this paper is organized as follows. In Section2 we provide a brief background on edge computing and related work in using edge computing for autonomous vehicles and smart cities. Section 3 describes the overall system architecture for DRF Edge. In Section 4 we outline some example use case scenarios that demonstrate the effectiveness of the proposed architecture. Finally, we conclude the paper in Section 5.

## 2 Background and Related Work

In this section, we define edge computing and present two areas where edge computing is gaining traction, namely autonomous vehicles and smart cities. Both areas share similar challenges to aviation autonomy, which can be summarized as follows.

- Low latency: AAM consumers (e.g., UAVs) require low latency responses to support their real-time operations. Pushing computing to the edge can reduce network latency, thus improving safety and reliability.
- Massive data: AAM service providers generate massive amounts of data (e.g., video feeds), which require analysis in near real-time. Moving this data to the cloud for further processing is simply not feasible due to latency, bandwidth, and cost.

- Privacy and security: maintaining data locally at the edge is critical for some participants to reduce surface attacks, maintain privacy, and adhere to regulations.
- Reliability: while the network connectivity to the cloud has improved significantly, and will likely continue to improve with 5G, network partitioning remains a challenge (especially for AAM) due to the AAM-mobility requirements. The edge presents a suitable platform for critical AAM applications to continue to run despite network failures.

## 2.1 Edge Definitions and Landscape

There is a growing body of work in computing models and frameworks that focuses on the ability to execute computation outside of a cloud environment—either on end devices themselves or somewhere in the middle, in order to reduce latency [49, 37] . This work has proposed a variety of terms including edge computing [43], fog computing [15], and Mobile Edge Computing (MEC) [29], to name a few.

We do not aim to taxonomize the terms that others have proposed, as the distinctions are not important for our purposes. Instead, we restrict ourselves to four distinct tiers of resources (see Figure 1) that span the continuum from the cloud to the extreme edge. We define these tiers and list their respective properties below.

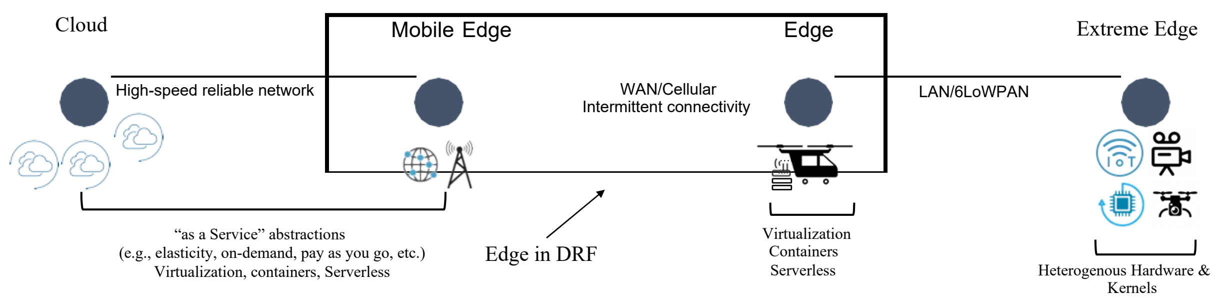

Cloud
Mobile Edge
Edge
Extreme Edge
High-speed reliable network
WAN/Cellular
Intermittent connectivity
LAN/6LoWPAN
"as a Service" abstractions
(e.g., elasticity, on-demand, pay as you go, etc.)
Virtualization, containers, Serverless
Edge in DRF
Virtualization
Containers
Serverless
Heterogenous Hardware &
Kernels

Figure 1: The four tiers in our system model that form a computing continuum

- The extreme edge refers to end devices where data is generated and/or consumed (e.g., IoT sensors and actuators, cameras, UAV sensors). These devices can be powered or battery operated. Their processing power is very limited, but it can be enhanced by custom hardware (e.g., FPGAs). The extreme edge is closest to data sources (lowest latency) but exhibits the most heterogeneity in terms of hardware and kernels. The extreme edge also has very limited capacity and low memory and computing power.
- The local edge refers to local resources that are always-on and one network hop (usually wireless but possibly wired) from the extreme edge (e.g., cloudlets [41], gateways, micro clusters, UAV on- board computers). These resources have higher capacity and computational power (when compared to the extreme edge), but are still very limited. Edge resources can take advantage of virtualization, containers, or a serverless platform to simplify resource management.
- The remote edge (also known as Mobile Edge Computing (MEC) [29]), refers to resources between the cloud and the edge (e.g., Content Delivery Networks (CDNs), Internet Service Providers (ISP) local stations, network Points of Presence (PoPs), cell tower base stations, Road Side Units (RSUs)). The remote edge can be one or more tiers (from a network perspective) which have closer proximity to the extreme edge (i.e., lower latency) but higher cost and limited capacity (compared to the cloud). These resources can take advantage of virtualization, containers, or a serverless platform to simplify resource management. Further, a few commercial offerings [2, 6, 5, 4] provide "as a Service" abstractions for edge infrastructure (i.e., compute, memory, storage, etc.), where users can take advantage of a pay-as-you-go model, on-demand access, and elasticity as they would in the cloud.
- The cloud refers to large data centers provided as a service over the Internet. The cloud provides the illusion of infinite capacity, elasticity, state-of-the-art hardware, pay-as-you-go billing, and on-demand access. However, cloud resources are furthest from data sources (at the extreme edge), thereby introducing additional latency when processing requests.

## 2.2 Edge Computing for Autonomous Vehicles

Autonomous vehicles (AVs) are often considered a "killer" use case for edge computing, due to their requirements for low-latency computations that can involve massive amounts of data [40]. In order to successfully deploy edge-based AV systems, however, edge systems must comply with AVs' safety and performance requirements. These include the need to process large amounts of data, up to 4 TB per hour (on the order of one GB per second) [45, 36], satisfy strict energy constraints on battery-powered AVs, and provide security for all layers of the AV stack to avoid compromising passenger safety [36].

AVs today are in widespread testing, with Waymo recently announcing that driverless vehicles would be available for ride hailing in the Phoenix area [48]. Self-driving cars, however, are not yet widely available to the public, in part due to the challenges of making them fully autonomous due to the stringent requirements in sensing, perception, localization, control, decision making, and map generation. In addition, communication, coordination, and cooperation among vehicles required for efficiency and safety raises further challenges. Vehicle to X (V2X) broadly encompasses communication systems involving Vehicle to Vehicle (V2V), Vehicle

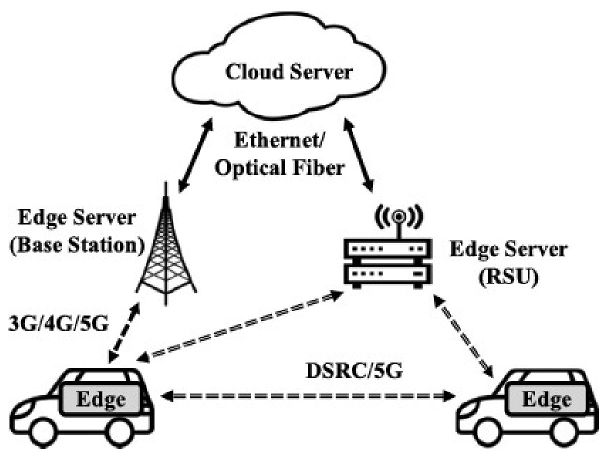


Figure 2: Overview of an edge computing system designed for autonomous vehicles, taken from [36].

to Infrastructure (V2I), Vehicle to Grid (V2G), Vehicle to Pedestrian (V2P), and so forth. A single vehicle may encounter a bump on the road due to unknown debris or deteriorating road conditions, and broadcast information to X while simultaneously receiving potentially redundant and/or unimportant information from X. Considering the number of on-board vehicle sensors and the myriad of potential environmental conditions encountered on the road, the volume of V2X traffic transmitted over a peer-to-peer mesh network could potentially overwhelm a poorly equipped or overly (computationally) burdened vehicle [22].

2020: Existing Technologies. As noted above, AVs today are just beginning to be deployed to the general public, and today's AV deployments do not leverage edge servers, which themselves are not widely deployed. Edge computing is, however, a topic of great interest to mobile network operators, who already have the back haul infrastructure to support a large network of geographically distributed servers. In 2017, AT&T and NTT both announced plans to build networks of edge data centers [45]. In 2019, Amazon (through a partnership with Verizon) and Microsoft (through a partnership with AT&T) announced commercial offerings for edge computing [2, 6]. Similarly, automakers have also started exploring edge computing as a potential complement to their vehicles, with Porsche [32] recently citing several prototypes of edge computing use cases for their products. In the remainder of this section, we first outline the components of an edge-based AV system before discussing the current deployment status in more detail.

Components of an edge-based AV system. Figure 2 shows a high-level overview of edge computing infrastructure for autonomous vehicles. We observe that, in addition to edge servers located near vehicles, the vehicles themselves can act as edge nodes with limited computing power. Stationary edge nodes would likely be placed at cellular base stations or as roadside units (RSUs) along roadways. The communication network connecting these edge servers is just as crucial to edge computing deployments, and also has multiple parts. Since AVs are generally mobile, they would need some form of wireless connectivity (cellular connectivity to base stations and 5G or DSRC vehicle-to-vehicle connectivity). Wired networks would then act as back haul links from the edge servers at base stations to a cloud server.

An edge computing system for AVs also includes software components that enable applications to make use of edge resources. Figure 3 illustrates some of the application components that are necessary for AV navigation. The vehicle itself is equipped with multiple sensors that continuously collect data, which must be analyzed in order to make decisions in real-time. Recent work suggests that localization and perception are the bottleneck functions in AVs [36], suggesting that specialized hardware may be needed for these algorithms. Cloud servers supplement on-device analysis, for example by training machine learning models for object tracking, storing historical data, and producing high-dimensional maps. Runtime managers can allocate these tasks to different hardware resources so as to ensure good performance.

Existing edge deployments. Edge computing deployments are not yet widely available, and neither are AVs. Some companies, however, have begun pilot AV deployments. Despite a lack of edge servers, Waymo [48], Cruise [26], and Tesla [27] have begun trial roll outs of vehicles that do not require driver control. In these deployments, much of the data processing is done on the AV itself. However, introducing more advanced features like coordination between AVs may require external edge servers. Such coordination may be even more important for aerial vehicles compared to terrestrial vehicles, due to the lack of predefined roadways for aerial vehicles. Aerial vehicles may also have more restrictive on-board computing constraints, i.e., due to weight limitations and may not support the high workload of camera and LIDAR perception.

Autonomous vehicles that do not carry human passengers have also made recent progress. Driverless trucks, for example, are being actively developed by several companies, including a partnership between Waymo and Daimler [28]. The farming industry is an early adopter of autonomous equipment for use in private farms [33]. These vehicles generally offload some of their processing and analytics tasks to cloud instead of edge servers. Edge computing deployments in general are also making progress, with Walmart starting a venture to compete with Amazon by locating edge servers at its super centers, effectively creating a highly geo-distributed cloud [25]. Such infrastructure could be exploited by AVs, as both Walmart stores and AVs would likely be concentrated in relatively highly populated areas [21].

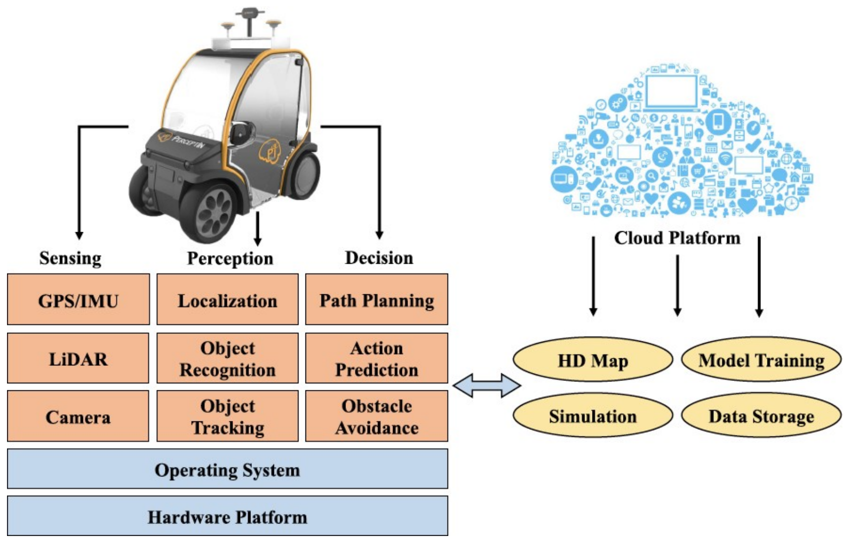


Figure 3: Overview of an edge computing system designed for autonomous vehicles, taken from [36].

2035: Deployment Vision. Over the next fifteen years, many industry players expect AVs to achieve widespread deployment, and to make extensive use of edge computing ideas. Since our work focuses on the edge infrastructure required for aerial vehicles in urban air mobility, we focus our discussion of the predicted status of AVs on aspects related to edge computing. We first discuss the predicted state of vehicle-to-everything (V2X) communications, and then discuss specific use cases and applications that are envisioned to be enabled by edge computing. We conclude by discussing stakeholder incentives to deploy the infrastructure needed to realize this vision.

V2X communications. Wireless connectivity for autonomous vehicles is an active area of research, with the 3GPP cellular network standardization association including 5G-enabled vehicle-to-everything (V2X) communication in its release 16 [14]. This work includes V2V (vehicle-to-vehicle), vehicle-to-infrastructure (V2I), and vehicle-to-network (V2N) communications. Recent work comparing cellular (LTE), WiFi, and DSRC-based communication interfaces shows that none of them conclusively dominate the others, e.g., WiFi has better end-to-end latency for image transfers, but not basic safety messages. None of the interfaces is particularly energy-efficient, suggesting that energy efficiency is an important area of future research on V2X communications [35] and future deployments may utilize a combination of these technologies in a "heterogeneous vehicular network" [36].

30], since false or intercepted messages may have severe safety consequences for the AV. Most current solutions rely on either cryptographic schemes that guard against outside attackers, or on trust-based schemes that guard against "insider" attacks. Privacy attacks may try to infer location or identity information of an AV. The specific threats for V2X, and thus the required defense mechanisms, may depend on the interface used (e.g., 5G or LTE versus DSRC).

Edge-AV applications. Allowing AVs to offload some computations to edge servers effectively expand the computing power available to AVs. Several frameworks have been proposed to optimize the required data sharing and compute offloading and scheduling [24]. Such offloading, however, may be more effective for some applications than others. We review some particularly compelling AV applications that may require edge servers below.

Multi-sensor fusion underlies much of the functionality behind AVs, which may collect radar, camera, LIDAR, GPS, and other data at the same time [47, 32]. Fusing this data together then enables applications like object tracking and environment reconstruction. While several algorithms have been proposed for such data fusion, safety concerns remain [47]. Sophisticated models based on deep learning may improve the fusion performance, and external edge servers (e.g., RSUs) that allow vehicles to access data from further away vehicles (i.e., cooperative sensing [36]) may also improve safety by improving the accuracy of the fused data model. Edge nodes at vehicles and edge servers can also preprocess data to reduce the communication requirements of analyzing it.

Finally, applications that fall under cooperative autonomous driving [36] will often benefit from the presence of an external edge server, as it allows for better coordination between vehicles. Similarly, edge servers in our approach may be used for applications like cooperatively assigning flight plans to multiple vehicles in an area. Some applications may be able to execute without assistance from roadside units, e.g., "seeing through" vehicles can be done through message passing between vehicles on the road in order to determine if it is safe to pass [40]. Others, like intersection management or virtual traffic lights, may require external edge servers that take as input data from multiple vehicles and then optimize over these vehicles' movements accordingly [36].

Deployment incentives. While autonomous driving is projected to be a multi-billion-dollar industry, realizing the vision of widespread AV deployments requires investments from multiple stakeholders. These include the 5G industry (including telecom operators and vendors), users, the automotive industry, policy makers, and road infrastructure operators [14]. While the automotive industry has a clear incentive to produce autonomous cars due to the potential for directly making revenue from them, other stakeholders like the 5G industry may not have clear incentives to provide the necessary investments.

The cost of deploying 5G V2X networks is anticipated to be particularly large, though it may be partially offset by some form of infrastructure sharing. An analysis of the AV market in Europe suggests that network

| | | | | Administrative Domain | | | |
|---|---|---|---|---|---|---|---|
| Name | Developer | App. Domain | Sensors | Multiple producers | Multiple consumers | Latency | Computing |
| Apache Edgent [3] | Apache, IBM | General | General | yes | yes | real-time | Edge, cloud |
| Akraino Edge Stack [1] | Linux Foundation, Intel, AT&T | General | General | yes | yes | real-time | Edge |
| SAGE [9] | Argonne National Labs | Smart city (Chicago) | Air, weather, noise, light, cameras | no | no | real-time | Device, cloud |
| WIFIRE [13] | UC San Diego | Wildfire monitoring | Weather, cameras, satellite | yes (university, park service, utility companies) | no | real-time, 10 minutes (weather), quarter day (satellite) | Cloud |

Table 1: Smart city research prototypes and general edge open-source platforms

operators may reach profitability within 5 to 10 years under various deployment scenarios.

## 2.3 Edge Computing for Smart Cities

2020: Existing Technologies. Smart city research prototypes are being deployed in public spaces, resulting in a massive number of low-powered devices collecting sensor data. The types of sensor data being collected depends on the specific use case within smart cities, ranging from environmental sensors such as air quality and noise, to infrastructure and transportation sensors such as WiFi hot-spot usage or GPS sensors. These deployments share a similar concept with our vision, where multiple producers (e.g., temperature sensors in a building), produce data that can be consumed by one or more consumers (e.g., a city building that wishes to automatically control its HVAC electricity consumption). Below we discuss several examples of research prototypes, civic deployments, and edge platforms for smart cities.

Research prototypes: Several smart city research prototypes are summarized in Table 1. SAGE [9] (and its predecessor, the Array of Things) is an environmental monitoring platform led by Northwestern University and originally deployed in Chicago. Its sensors include air, weather, noise, light, and cameras, with the aim of providing open data to researchers and to the government. WIFIRE [13] is a wildfire monitoring platform run by UC San Diego with local utility partners, and captures weather information, satellite imagery, and camera data. Its goal is to detect, monitor, and predict wildfires, and provide this information to the public through a web-based interface. The multiple sensors run by different administrators (e.g., National Park Service weather, San Diego Gas & Electric cameras) are examples of multiple data providers in the aviation autonomy context. Processing in SAGE and WIFIRE currently runs on the devices or in the cloud, due to their philosophies of centralized information collection and distribution to the public. Civic deployments. Alongside these research prototypes, there are several civic deployments that are more tightly integrated with city operations, as summarized in Table 2. SmartCity PHL [12] is deployed in Philadelphia, and covers a wide variety of use cases (not all yet deployed), including snowplow monitoring, HVAC control in government buildings, language translation to communicate with city residents, and more. Toronto Quayside [8] was a planned neighborhood in Toronto led by Sidewalk Labs (a subsidiary of Alphabet), and included air quality, water, noise, and traffic monitoring. It was cancelled in May 2020 in part due to the COVID-19 pandemic and the resulting economic uncertainty, as well as privacy concerns from local residents. San Jose's smart city plans [11] target a variety of use cases, including crime, housing, sustainability, transportation, etc. In general, "smart city" is a broad term for these civic deployments, encompassing digital technologies across many domains (e.g., transportation, infrastructure). Computing is generally considered in a centralized context, with little mention being made of whether these computations reside on the cloud or the edge. Furthermore, privacy and data transparency are consistently highlighted as key requirements for these real deployments, with cities affirming that they will uphold the NYC Guidelines for the Internet of Things [7] which establishes guidelines for privacy and transparency [10, 12].

General edge platforms. Moving away from specific smart city deployments, there are also several generic edge platforms that are designed for low-power sensors, which thus may be applicable to the smart city scenario. These platforms are included in Table 1. Apache Edgent [3] is an open-source platform originating from IBM, focusing on real-time edge and cloud computations. It includes the concept of multiple producers and consumers. Akraino Edge Stack [1] was started by an industry consortium including Intel and AT&T, and is now open-source, focusing on edge processing. Other edge platforms are further described in [34]. While these platforms are relatively recent (e.g., Akraino Edge Stack was launched in 2018), and the specific smart city prototypes and deployments described above are generally older and appear to rely on custom-made software, as general edge platforms become more stable and mature, smart city deployments in the future may choose to adopt such platforms.

|  |  |  |  | Administrative Domain |  |  |  |
|---|---|---|---|---|---|---|---|
| Name | Developer | Application Domain | Sensors | Multiple producers | Multiple consumers | Latency | Computing |
| SmartCity PHL [12] | city; university, industry, nonprofit partners | snowplow monitoring, centralized HVAC control, language translation, etc. | WiFi hotspots, automatic vehicle locations, HVAC, microphone | no | no | 15 s (vehicles), Real-time (voice) | Centralized |
| Toronto Quayside (cancelled) [8] | Alphabet | general smart city | Air, water, noise, traffic counters | no | yes (e.g., city departments) |  | Edge |
| San Jose [11] | city; university, industry, nonprofit partners | Crime, housing, sustainability, transportation, etc. | Air, water, noise, etc. | no | no | App dependent | Centralized |

Table 2: Smart city civic deployments

2025 and Beyond: Deployment Visions. With smart city deployments underway in many cities, and a large number of use cases in the design phase, smart cities are expected to persist into 2025 and beyond. There are several research trends that can enhance existing deployments or be incorporated into the design of future deployments. Below, we outline several research topics at the intersection of smart cities and edge computing; namely, privacy and security, sensor communications, and data processing.

Privacy and security. Security and privacy are emerging as some of the key concerns for widespread deployment of smart cities, with push back from local communities based on these concerns already having significant deployment impact [16, 20]. To help alleviate privacy concerns, edge computing provides a possible solution, either by filtering data before forwarding to a central collection point, or by restricting information flow and keeping data close to the data source. Public access to data can also be restricted to local regions. However, even when transferring data locally between a sensor and its destination, security should be preserved, which can be challenging because sensors and edge nodes lack computational resources for traditional security protocols, and may be more physically accessible than traditional cloud data centers in remote locations [44]. Thus, designing lightweight schemes that are resistant to attack is a key concern moving forward.

Sensor communications. Sensor data collected in smart cities can range from relatively delay-tolerant data, such as temperature readings, to data with real-time requirements, such as an image from a traffic camera. The majority of the data will be passed between machines (e.g., a sensor and an edge node) without human intervention, with humans interacting with the data only after it has been communicated, filtered/processed, and displayed. The large amount of sensors, their density, and the associated traffic is expected to present a challenge to existing cellular networks. To help address this, cellular standards include special modes to handle these types of traffic, called massive machine type communications (mMTC) for smart cities, as well as an ultra-reliable low-latency communications (URLLC) mode for critical data traffic [42]. With 5G deployments just beginning and lacking mMTC and URLLC support, designing smart city infrastructure with software and hardware that can leverage new 5G technology is an ongoing challenge.

Data processing. Smart cities already encompass a bewildering array of applications, as described above, and the pool of possible applications will only continue to grow. These new applications will put more strain on edge infrastructures, especially those applications that rely on artificial intelligence or deep learning, which require significant computational resources [18]. For example, an augmented reality (AR) framework for road repair would require compute power (for deep learning to identify the repair type), local processing (to prevent camera data from leaking to outside observers), high bandwidth (to offload the video stream for processing), and low latency on the order of 100 ms (for real-time AR user experience [19]). Multiple workers collaborating on the repair would only further increase the computational resources required. Such requirements could be satisfied by a local edge server optimized to handle multiple incoming data streams [39], or by increasingly powerful sensors that can perform processing on the sensor itself (e.g., Amazon DeepLens, which can perform deep learning on the camera), reducing the need to communicate with an edge server.

## 3 Architecture

Edge computing provides infrastructure that is closer than cloud resources and may be used for navigation of aerial vehicles, which has similar safety requirements to navigation of terrestrial vehicles. Vehicles in DRF will likely have a similar technology stack to terrestrial vehicles to enable their navigation, though new modules, e.g., for platooning, may be implemented at edge servers or on the vehicles themselves to enable

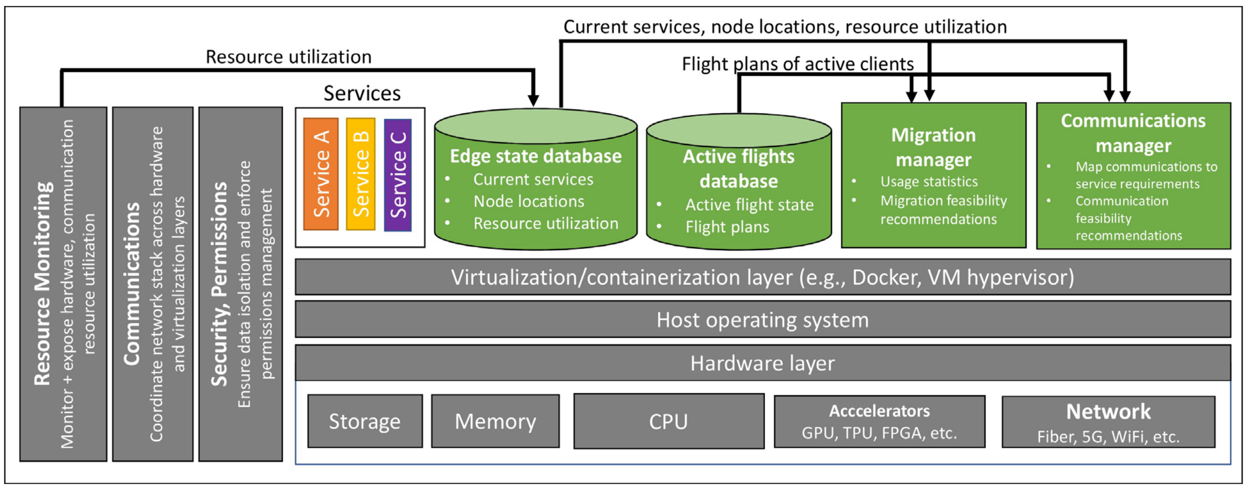

Current services, node locations, resource utilization
Resource utilization
Flight plans of active clients
Services
Resource Monitoring
Monitor + expose hardware, communication resource utilization
Communications
Coordinate network stack across hardware and virtualization layers
Security, Permissions
Ensure data isolation and enforce permissions management
Service A
Service B
Service C
Edge state database
Current services
Node locations
Resource utilization
Active flights database
Active flight state
Flight plans
Migration manager
Usage statistics
Migration feasibility recommendations
Communications manager
Map communications to service requirements
Communication feasibility recommendations
Virtualization/containerization layer (e.g., Docker, VM hypervisor)
Host operating system
Hardware layer
Storage
Memory
CPU
Accelerators
GPU, TPU, FPGA, etc.
Network
Fiber, 5G, WiFi, etc.

Figure 4: Edge node architecture.

more advanced navigation features.

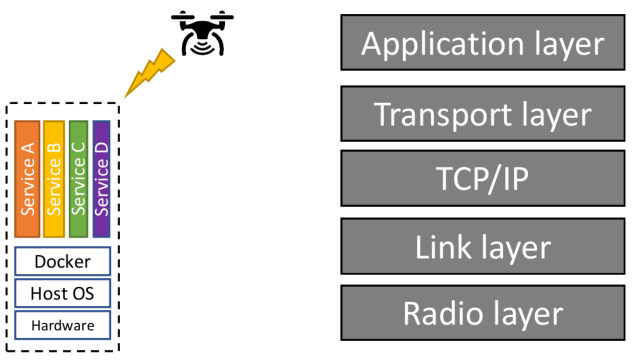
DRF does not provide edge infrastructure and is not responsible for its management or operation. Instead, there are two types of edge resources in DRF. The first type is provided as a service by external providers (e.g., AWS Wavelength). In this case, DRF is not involved in the exchange between the edge provider and DRF service providers (i.e., it does not monitor these resources or suggest deployment or migration plans). The other type of edge resources is more focused on smaller edge providers with finite capacity (e.g., roof top units, roadside units, in vehicle nodes). These resource providers can opt in to install DRF libraries, which will allow DRF to monitor and provide access control to this resource. In this case, DRF can provide suggestions to DRF service providers on where to deploy or migrate services but it is the responsibility of the service provider to ensure that the recommended resources satisfy their requirements and will not impact their SLAs. In the remainder of this section, we discuss the edge architecture for the second type of edge resources, as well as the communication layer between vehicles and edge nodes.

The architecture of an edge node is shown in Figure 4. Recall that the edge nodes are responsible for hosting DRF provider services. Consumers can connect to an edge node to access these services. A library of services, denoted by example Services A, B, and C, is shown in Figure 4. The grey boxes denote the standard components of an edge node. The green boxes denote the non-standard components of an edge node, which are unique to DRF. The arrows denote information flow between different components of the edge node. We first briefly describe the standard components below:

- Resource monitoring: This component monitors and exposes relevant information about the hardware layer, communication module, and resource utilization.
- Communications: This component runs the appropriate application and transport layers of the network stack. The lower layers of the network stack are handled by the virtualization and hardware layers.
- Security, permissions: This component is responsible for maintaining security and appropriate data permissions.
- Virtualization/containerization layer: This component is the virtualization layer of the chosen virtualization technology (e.g., the virtual machine hypervisor, Docker container layer).
- Host operating system sits above the hardware layer and exposes the hardware functionality to the virtualization layer.
- Hardware layer: The hardware layer consists of storage, memory, CPU, any accelerators (e.g., GPU, TPU, FGPA), and the network hardware (e.g., wired fiber connection, wireless 5G and/or WiFi).

The DRF -specific components of the edge node architecture are shown in green in Figure 4. At a high level, the Edge State Database provides general information about edge nodes, while the Active Flights Database, Migration Manager, and Communication Manager work together to provide additional intelligence when more information (e.g., flight plans) about the current state of the DRF ecosystem is available. This additional information enables more intelligent decision making to improve the overall system performance of DRF, in terms of metrics such as migration latency, communication throughput, etc., thus helping DRF providers meet their service guarantees. These components while logically part of the edge architecture, do not necessarily need to reside at the edge. For example, this information can be centrally aggregated to provide a global view of edge infrastructure status, as well as leverage centralized optimization techniques to find optimum placement and migration plans. We describe each DRF -specific component in details below:

- Edge State Database: The purpose of this database is to maintain general information about edge nodes. For each edge node, it contains a list of currently running services, geographic location, and resource utilization (e.g., CPU, memory). This information can be accessed by other components on the edge node, or by DRF providers. For example, a provider could request this information to determine the geographic distance (and corresponding network latency) between itself and a consumer, and hence whether to enter a contract with that consumer or not.



(a) Communication between a UAM vehicle and an edge node

(b) Layers of the network stack

Figure 5: Communication layer.

- Active Flights Database: The purpose of this database is to maintain information about which AAM vehicles are currently active, and their flight plans (if permissible). This information can be used by providers to discover, place, and/or migrate services. For example, a provider could use this information to predict that the edge node currently hosting its service will soon become congested, and preemptively migrate the service to a less congested edge node.

- Migration Manager: The purpose of the Migration Manager is to provide recommendations of migration decisions based on the current load of the edge nodes. The Migration Manager uses its knowledge of flight plans (from the Active Flights Database) and edge node resource information (from the Edge State Database) to predict when service violations may occur (e.g., due to a AAM vehicle flying out of range of an edge node). Based on this information, it can warn a provider that a contract violation is about to occur.

- Communications Manager: The goal of the Communications Manager is similar in spirit to the Migration Manager, but rather than making migration decisions, it makes communication decisions. That is, to help providers satisfy their contracts, it uses information about the flight plans and edge nodes' resource utilization to adjust the communication between an edge node and a AAM vehicle. For example, if a consumer located on a AAM vehicle enters an urban area with unreliable 5G signals due to wireless interference, the Communications Manager can recommend the AAM vehicle switch to a higher-reliability but lower throughput network protocol.

Edge-UAV Communication Layer. In this section, we describe the communication link between an AAM vehicle and an edge node, as shown in Figure 5a. There are various choices for the communication link at various layers of the network stack (illustrated in Fig. 5b). We first describe various options available at the radio and link layers, and their pros and cons. We then discuss possible application layer protocols. The transport layer is assumed to be TCP or UDP, as alternative transport layer protocols are not widely supported. The final selection of radio, link, transport, and application layers is chosen by a DRF provider/consumer based on the service agreement in the smart contract. The network is assumed to operate on a TCP/IP-based network (clean-slate designs such as information-centric networks are an alternative, but outside the scope of this project).

Radio and link layers. The radio and link layers are typically bundled together into a protocol, for example WiFi or LTE. Table 3 summarizes several possibilities for the link between an edge node and a AAM vehicle, in terms of their typical throughput, latency, use by AAM-type vehicles, and upcoming features. The options include WiFi, cellular (including 4G/5G), satellite, and LoRa for long-range, low-power communication. To summarize, WiFi and cellular networks can provide the highest throughputs and lowest latencies, and are already used by some existing AAM vehicles such as UAVs and passenger aircraft. However, their range can be limited, with cellular base stations covering approximately 1 square km. On the other hand, long-range communication methods such as satellite or LoRa can fill those gaps in coverage but have much lower throughput and longer latency.

Application layer. Options at the application layer include HTTP, HTTP/2, WebSockets, WebRTC, and MQTT. HTTP is one of the most commonly used options on the Internet today, and HTTP/2 provides additional useful features such as server push and multiplexing between data streams on a single connection. Furthermore, the commonly used REST APIs are built on top of HTTP. WebSockets provides full-duplex streams, while WebRTC targets real-time applications. Finally, MQTT is a lightweight publish-subscribe based protocol for lightweight sensor devices. The choice of protocol depends on the type of data being transferred between a consumer and a provider; for example, a high bandwidth camera stream may be sent using HTTP, while small, infrequent weather data may use MQTT. Furthermore, because the application layer can have unforeseen interactions with the lower layers [31, 23], the application layer should be carefully chosen based on the selected radio/link layer.

| Radio/link layer | Throughput | Latency | Currently used by | Upcoming |
|---|---|---|---|---|
| WiFi (802.11ax) | 1 Gbps | 10 ms | UAVs | WiFi 7 in 2-5 years with 30 Gbps |
| Cellular | 4G: 100 Mbps 5G: 1 Gbps | 4G: 50 ms 5G: <10 ms | 4G: UAVs, passenger aircraft | 6G in 10 years |
| Satellite | 1 Mbps | 500 ms | Passenger aircraft | high-altitude balloons with 1 Mbps |
| LoRa | 1 Kbps | | agriculture, smart city | |

Table 3: Comparison of wireless communication technologies currently used by AAM-type vehicles.

## 4 Example Use Cases

In order to demonstrate the effectiveness of our approach, we present two example use case scenarios that illustrate the required steps for deploying and migrating services using the aforementioned architecture.

### 4.1 Emergency Landing

This scenario is an Unmanned Aerial Vehicle (UAV) attempting to make an emergency landing in the Hudson River, while avoiding any boats currently present. Due to poor weather conditions, visibility from the air is poor, but there is a service provider that has cameras close to the water surface, providing better visibility. Another service provider operates a machine learning service that can count the number of objects (i.e., boats) in a scene. Note that object counting is a challenging computer vision problem, which is often done using deep learning and requires significant computational resources to run with low latency [38], hence the need for an edge node. Figure 6 shows an example of the interactions between the consumer (the vehicle) and the camera capture and machine learning providers. These steps are:

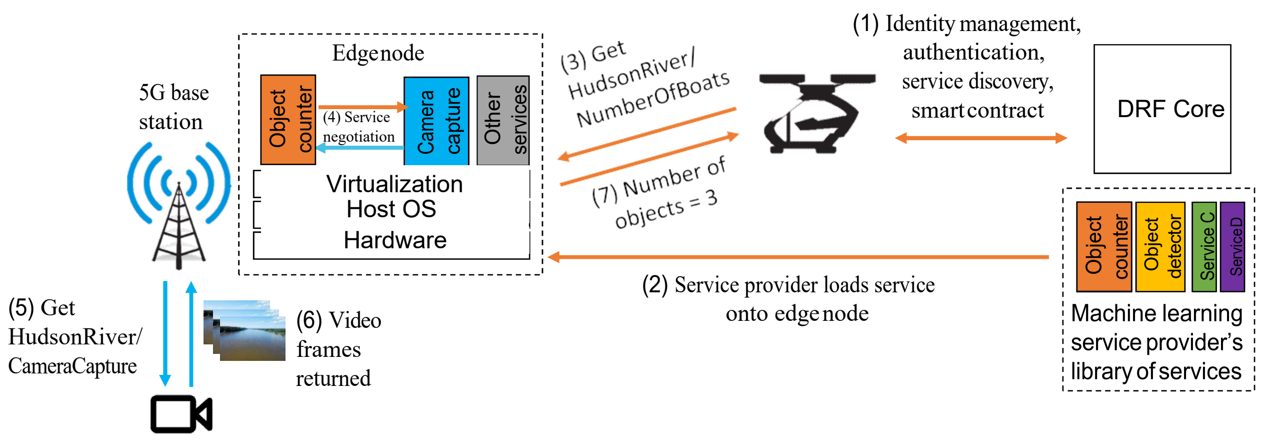

Edgenode
Object counter
(4) Service negotiation
Camera capture
Other services
Virtualization
Host OS
Hardware
5G base station
(3) Get HudsonRiver/ NumberOfBoats
(7) Number of objects = 3
(1) Identity management, authentication, service discovery, smart contract
DRF Core
Object counter
Object detector
Service C
Service D
Machine learning service provider's library of services
(2) Service provider loads service onto edge node
(5) Get HudsonRiver/ CameraCapture
(6) Video frames returned

Figure 6: An initial edge deployment example for an emergency water landing.

1. Identity management, authentication, service discovery, smart contract: The consumer learns about the available providers and negotiates a contract with the machine learning service provider.

2. Service provider loads service onto an edge node: The machine learning service provider negotiates a contract with an edge provider and loads the object counter service onto the edge node closest to the camera service provider.

3. Get HudsonRiver/NumberOfBoats: The consumer sends a request to retrieve the number of boats near its current location on the Hudson River.

4. Service negotiation: The machine learning service provider (shown in orange) negotiates a connection with the camera capture service provider (shown in blue), if the connection has not been pre-established.

5. Get HudsonRiver/CameraCapture: The camera capture service requests the video feed from the camera over a 5G connection.

6. Video frames returned: The video frames are returned to the camera capture service, which forwards them to the object counter service.

7. Number of objects = 3: After determining that there are three boats present in the scene, the result is returned to the consumer, and the UAV must search for another location to land.

Moreover, the above steps must take place quickly in order for the returned results to be useful to the UAV. Prior work [17] suggests that feedback at least every 200 ms allows enough time for the autonomous control to update its flight path and land safely, although more recent work may further reduce that requirement.

## 4.2 Search and Rescue

In this use case scenario, we give an example of service migration across edge nodes. The scenario is an UAV ("UAV A") performing search and rescue for a missing hiker in a park. The UAV has an on-board camera and wishes to analyze the image to search for the missing hikers, who may blend into the background. Detecting specific objects (i.e., the missing hiker) in a scene is a challenging computer vision problem that is often tackled with deep learning, which requires computational resources that might not be available on the UAV [46]. Thus, the UAV (i.e., the consumer) contracts with a machine learning service provider, which performs object detection on the camera frames to detect whether the missing hiker is present and a rescue should be initiated at that location. Figure 7 shows an example of the interactions between the consumer (the vehicle) and the machine learning provider.

We assume that the consumer-provider relationship has already been established (similar to Section 4.1). Having finished searching the area around vertipad 1 with a negative result, the UAV begins to navigate to its next waypoint at vertipad 2. However, vertipad 2 is far from edge node A where the machine learning provider currently hosts the video analytics object detection service, and the wireless link between UAV A and edge node A would be disconnected upon arrival at vertipad 2. The migration steps are outlined below:

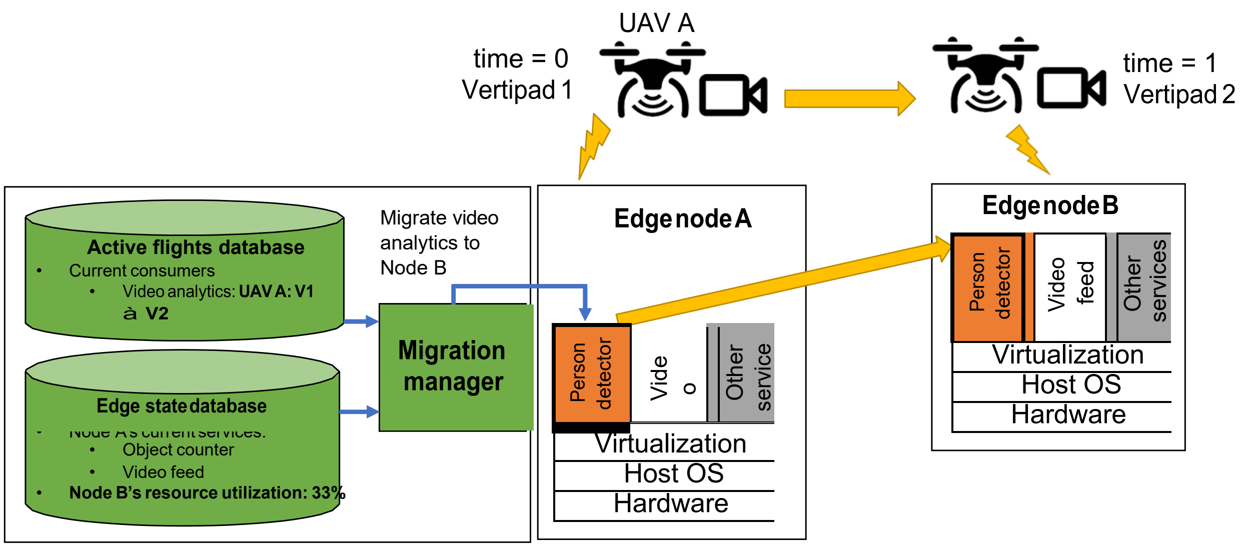


Figure 7: An edge migration example for an aerial search and rescue mission.

1. The Active Flights Database has information about this flight plan ("UAV A: V1↦V2"), and notifies the Migration Manager.
2. The Migration Manager retrieves information from the Edge State Database that edge node B is close to vertipad 2, and has low resource utilization ("Node B's resource utilization: 33%").
3. Based on this information, the Migration Manager initiates the migration of the video analytics service from node A to B in advance of the UAV's arrival at vertipad B, timing it so that the migration finishes just as the UAV arrives at node B and establishes a wireless connection. This enables the video analytics service to proceed seamlessly without missing a frame.

## 5 Conclusion

In this paper, we presented a decentralized edge architecture for aviation autonomy. The DRF Edge architecture enables data and reasoning providers to run their applications with the desired low latency required for real-time operations. Unmanned Aerial Systems can leverage edge computing resources to complement their on-board computing power by interacting with data and reasoning services that are deployed on top of this infrastructure. We presented two use case scenarios to demonstrate how this architecture can support operations such as emergency landing or search and rescue, with the underlying services being automatically deployed and/or migrated across edge infrastructure as needed.

## References

[1] Akraino edge stack. https://www.lfedge.org/projects/akraino/.

[2] Amazon aws wavelength. https://aws.amazon.com/wavelength/.

[3] Apache edgent. https://edgent.incubator.apache.org/.

[4] Cloudflare workers. https://workers.cloudflare.com.

[5] Fastly serverless at the edge. https://www.fastly.com/products/edge-compute/beta.

[6] Microsoft azure edge zones. https://azure.microsoft.com/en-us/solutions/ low-latency-edge-computing/.

[7] Nyc guidelines for the internet of things. https://iot.cityofnewyork.us/.

[8] Quayside - sidewalk toronto. https://www.sidewalktoronto.ca/plans/quayside/.

[9] Sage: Cyberinfrastructure for ai at the edge. http://sagecontinuum.org/.

[10] Smart city pdf. https://www.smartcitypdx.com/.

[11] Smart city vision — city of san jose. https://www.sanjoseca.gov/your-government/departments/ information-technology/smart-city-vision.

[12] Smartcity phl roadmap. https://www.phila.gov/media/20190204121858/SmartCityPHL-Roadmap. pdf.

[13] Wifire: Workflows integrating collaborative hazard sciences. https://wifire.ucsd.edu/.

[14] 5G PPP Automotive Working Group. Business feasibility study for 5g v2x deployment, 2019.

[15] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pages 13–16, 2012.

[16] L Cecco. Surveillance capitalism": Critic urges toronto to abandon smart city project. The Guardian, 6, 2019.

[17] Andrea Cesetti, Emanuele Frontoni, Adriano Mancini, Primo Zingaretti, and Sauro Longhi. A vision-based guidance system for uav navigation and safe landing using natural landmarks. Journal of intelligent and robotic systems, 57(1-4):233, 2010.

[18] Jiasi Chen and Xukan Ran. Deep learning with edge computing: A review. Proceedings of the IEEE, 107(8):1655–1674, 2019.

[19] Zhuo Chen, Wenlu Hu, Junjue Wang, Siyan Zhao, Brandon Amos, Guanhang Wu, Kiryong Ha, Khalid Elgazzar, Padmanabhan Pillai, Roberta Klatzky, et al. An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance. In Proceedings of the Second ACM/IEEE Symposium on Edge Computing, pages 1–14, 2017.

[20] Sopan Deb. Following outcry, hudson yards tweaks policy over use of vessel pictures. New York Times, 2019.

[21] Lance Eliot. News of walmart aiming to setup edge computing for self-driving cars generates both surprise and questions. Forbes, 2019. https://www.forbes.com/sites/lanceeliot/2019/12/23/ news-of-walmart-aiming-to-setup-edge-computing-for-self-driving-cars-generates-both-surprise-and-

[22] Lance Eliot. Breakthroughs in roadway edge computing are the missing link for self-driving car collaboration. Forbes, 2020. https://www.forbes.com/sites/lanceeliot/2020/03/24/ breakthroughs-in-roadway-edge-computing-are-the-missing-link-for-self-driving-car-collaboration/.

[23] Jeffrey Erman, Vijay Gopalakrishnan, Rittwik Jana, and Kadangode K Ramakrishnan. Towards a spdy'ier mobile web? IEEE/ACM Transactions on Networking, 23(6):2010–2023, 2015.

[24] Jingyun Feng, Zhi Liu, Celimuge Wu, and Yusheng Ji. Ave: Autonomous vehicular edge computing framework with aco-based scheduling. IEEE Transactions on Vehicular Technology, 66(12):10660–10675, 2017.

[25] Patrick Gray. Leveraging assets: Walmart's new venture has lessons for us all. TechRepublic, 2019. https://www.techrepublic.com/article/ leveraging-assets-walmarts-new-venture-has-lessons-for-us-all/.

[26] Andrew J. Hawkins. Cruise is doubling down on shared autonomous rides with new covid protocols. The Verge, 2020. https://www.theverge.com/2020/10/21/21527014/ cruise-self-driving-origin-covid-safety-ruls-nhtsa-exemption.

[27] Andrew J. Hawkins. Tesla's 'full self-driving' beta is here, and it looks scary as hell. The Verge, 2020. https://www.theverge.com/2020/10/22/21528508/ tesla-full-self-driving-beta-first-reaction-video.

[28] Andrew J. Hawkins. Waymo and daimler are teaming up to build fully driverless semi trucks. The Verge, 2020. https://www.theverge.com/2020/10/27/21536048/ waymo-daimler-driverless-semi-trucks-cascadia-freightliner.

[29] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. Mobile edge computing—a key technology towards 5g. ETSI white paper, 11(11):1–16, 2015.

[30] Jiaqi Huang, Dongfeng Fang, Yi Qian, and Rose Qingyang Hu. Recent advances and challenges in security and privacy for v2x communications. IEEE Open Journal of Vehicular Technology, 1:244–266, 2020.

[31] Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. Confused, timid, and unstable: picking a video streaming rate is hard. In Proceedings of the 2012 internet measurement conference, pages 225–238, 2012.

[32] Matthias Hub. Seven edge computing use cases for vehicles. Porsche Newsroom, 2019. https:// newsroom.porsche.com/en/2019/digital/porsche-edge-computing-19509.html.

[33] David Linthicum. Where autonomous vehicles and edge computing have ar- rived. InfoWorld, 2020. https://www.infoworld.com/article/3518784/ where-autonomous-vehicles-and-edge-computing-have-arrived.html.

[34] Fang Liu, Guoming Tang, Youhuizi Li, Zhiping Cai, Xingzhou Zhang, and Tongqing Zhou. A survey on edge computing systems and tools. Proceedings of the IEEE, 107(8):1537–1562, 2019.

[35] Liangkai Liu, Baofu Wu, and Weisong Shi. A comparison of communication mechanisms in vehicular edge computing. In 3rd {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 20), 2020.

[36] Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. Edge computing for autonomous driving: Opportunities and challenges. Proceedings of the IEEE, 107(8):1697–1716, 2019.

[37] Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. Fog computing: A taxonomy, survey and future directions. In Internet of everything, pages 103–130. Springer, 2018.

[38] Daniel Onoro-Rubio and Roberto J López-Sastre. Towards perspective-free object counting with deep learning. In European Conference on Computer Vision, pages 615–629. Springer, 2016.

[39] Xukan Ran, Haolianz Chen, Xiaodan Zhu, Zhenming Liu, and Jiasi Chen. Deepdecision: A mobile deep learning framework for edge video analytics. In IEEE INFOCOM 2018-IEEE Conference on Computer Communications, pages 1421–1429. IEEE, 2018.

[40] Dario Sabella, Hassnaa Moustafa, Pekka Kuure, Sami Kekki, Zheng Zhou, Alice Li, Christoph Thein, Edwin Fischer, Ivan Vukovic, John Cardillo, Valerie Young, Soo Jin Tan, Vince Park, Michaela Vanderveen, Stefan Runeson, and Stefano Sorrentino. Toward fully connected vehicles: Edge computing for advanced automotive communications. 5G Automotive Association, 2017.

[41] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. IEEE pervasive Computing, 8(4):14–23, 2009.

[42] Mansoor Shafi, Andreas F Molisch, Peter J Smith, Thomas Haustein, Peiying Zhu, Prasan De Silva, Fredrik Tufvesson, Anass Benjebbour, and Gerhard Wunder. 5g: A tutorial overview of standards, trials, challenges, deployment, and practice. IEEE journal on selected areas in communications, 35(6):1201–1221, 2017.

[43] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. IEEE internet of things journal, 3(5):637–646, 2016.

[44] Mehdi Sookhak, Helen Tang, Ying He, and F Richard Yu. Security and privacy of smart cities: a survey, research issues and challenges. IEEE Communications Surveys & Tutorials, 21(2):1718–1743, 2018.

[45] Yevgeniy Sverdlik. Do driverless cars really need edge computing? DataCenter Knowledge, 2019. https://www.datacenterknowledge.com/edge-computing/ do-driverless-cars-really-need-edge-computing.

[46] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In Advances in neural information processing systems, pages 2553–2561, 2013.

[47] Zhangjing Wang, Yu Wu, and Qingqing Niu. Multi-sensor fusion in automated driving: A survey. IEEE Access, 8:2847–2868, 2019.

[48] Waymo. Be an early rider, 2020. https://waymo.com/apply/.

[49] Shanhe Yi, Cheng Li, and Qun Li. A survey of fog computing: concepts, applications and issues. In Proceedings of the 2015 workshop on mobile big data, pages 37–42, 2015.