DRF: A Software Architecture for a Data Marketplace to Support Advanced Air Mobility

Moustafa Abdelbaky*[1,3], Jiasi Chen†[5], Alexander Fedin‡[1,3], Kenneth Freeman§[1], Mohana Gurram¶[1], Abraham K. Ishihara‖[1,3], Carlee Joe-Wong**[4], Christopher Knight††[1], Kalmanje S. Krishnakumar‡‡[1], Isaias Reyes§§[1,3], Calvin Robinson¶¶[2], Peter Shannon***[6], Sandeep D. Shetye†††[1], Luka Tomljenovic‡‡‡[6] and William R. Van Dalsem§§§[1]

[1]*NASA Ames Research Center, Moffett Field, CA*
[2]*NASA Glenn Research Center, Cleveland, OH*
[3]*KBR Wyle Services, Moffett Field, CA*
[4]*Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA*
[5]*Computer Science and Engineering, University of California, Riverside, Riverside, CA*
[6]*Radius Capital, San Francisco, CA*

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 a digital marketplace, which allows service providers and consumers operating in the airspace ecosystem to securely exchange data and reasoning insights. In this paper, we present the architecture of a decentralized data marketplace that connects data and reasoning service providers to vehicles and other service consumers along the cloud-to-edge continuum. We also present two example use cases to demonstrate the value of our approach.

I. Nomenclature

AAM	=	Advanced Air Mobility
API	=	Application Programming Interface
AV	=	Autonomous Vehicles
CDN	=	Content Delivery Networks
CED	=	Crypto-Economic Design
DLT	=	Distributed Ledger Technology
DRF	=	Data & Reasoning Fabric
FAA	=	Federal Aviation Administration
IoT	=	Internet of Things
ISP	=	Internet Service Providers
MEC	=	Mobile Edge Computing
ML	=	Machine Learning
PCN	=	Payment Channels Networks
$PoPs$	=	Points of Presence

*Computer Scientist, Intelligent Systems Division.
†Assistant Professor, Computer Science and Engineering.
‡Backend Systems Architect, Aviation Systems Division.
§Aerospace Engineer, Aeronautics Projects Office.
¶Computer Scientist, Intelligent Systems Division.
‖Chief Technical Advisor Engineering Systems, AIAA Member.
**Assistant Professor, Electrical and Computer Engineering.
††Computer Engineer, Intelligent Systems Division.
‡‡Associate Division Chief, Intelligent Systems Division, AIAA Associate Fellow.
§§Software Engineer, Aviation Systems Division.
¶¶Data Architect, Information and Applications Division.
***Founder, Managing Director.
†††Computer Scientist, Intelligent Systems Division.
‡‡‡Partner.
§§§Chief Strategy Officer, Aeronautics Directorate.

SDK = Software Development Kit
SLA = Service Level Agreement
UAM = Urban Air Mobility
UAS = Unmanned Aerial Systems
UAV = Unmanned Aerial Vehicle
$UML6$ = UAM Maturity Level 6 (ubiquitous operations)

II. 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 and other urban Unmanned Aerial Systems (UAS) services, which supports a mix of on-board/ground-piloted and increasingly autonomous operations.

Critical to this path is the development of a digital marketplace connecting vehicles, reasoning, data, and infrastructure services across the cloud-to-edge continuum and enabling 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.

Current data platforms are not designed for the projected scale of geographically dispersed, heterogeneous entities with low-latency requirements operating in this new airspace ecosystem. Further, these platforms do not 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, they do not foster innovation and data sharing at the scale required to meet AAM challenges including UML6 and the FAA mission 2045 [1].

The Data & Reasoning Fabric (*DRF*) [2, 3] aims to meet these challenges by employing a decentralized data marketplace, which connects data and reasoning service providers to infrastructure 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 software architecture of the *DRF* marketplace and outline the contributions as follows.

1) *DRF* provides a trusted data and reasoning marketplace that enables secure, consistent, and verifiable transactions for all data and reasoning-related exchanges and supports request-response, publish-subscribe, as well as compute-to-data paradigms.
2) *DRF* provides a lightweight, extensible, and application-agnostic framework to support new applications of aerial mobility by making it easier to add new data and reasoning services.
3) *DRF* leverages open standards to seamlessly connect participants of the marketplace and support interoperability, while ensuring data privacy, protection, democratization, and monetization. Further, *DRF* is platform agnostic, thereby allowing existing data and reasoning platforms to participate in the marketplace.
4) *DRF* increases safety, system-wide monitoring, and operational efficiency by supporting the exchange of embedded information (i.e., metadata) about the quality of the data being exchanged, and measuring system performance and health of other parts of the system.
5) *DRF* lowers barriers to entry while giving participants greater control of their services, which is achieved by providing an SDK for both new and existing services providers to plug-in their services into *DRF* and an API for service providers to submit safety-critical closed feedback loops.

The remainder of this paper is organized as follows. Section III provides a brief background on some technologies relevant to the *DRF*software architecture, namely blockchains and edge computing. In Section IV we describe the overall system requirements, design, and architecture. In Section V we outline some example use case scenarios that demonstrate the effectiveness of our approach. Section VI discusses related work in blockchain-based marketplaces and edge computing. Finally, we conclude the paper in Section VII.

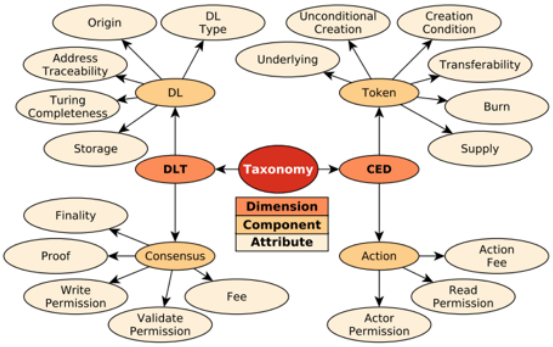

Fig. 1 Taxonomy of Distributed Ledger Technology (DLT) and Crypto-Economic Design (CED) [4].

III. Background

There are a lot of technologies that are relevant to *DRF* (e.g., Machine Learning, 5G communication, advanced control techniques, etc.). In this section, we provide a brief background on some of thee technologies that are relevant to the *DRF* software architecture, namely blockchains and edge computing.

A. Distributed Ledger Technology, Blockchains, and Smart Contracts

Blockchains are a type of distributed ledger technology (DLT), in which multiple workers (often called "miners") collectively store information and run a consensus protocol to synchronize this information [4, 5]. The left side of Figure 1 outlines the typical components of a distributed ledger. The two major components are the distributed ledger itself and the consensus mechanism, each of which has several sub-components, the choice of which may lead to various performance trade-offs. The impossibility result [6] proves that it is impossible to design a system that reaches deterministic consensus with one faulty process, which is unavoidable in distributed systems. Therefore, DLT systems must trade between security, decentralization, and scalability to overcome this limitation [5]. The "blockchain" structure is one such design choice, which specifies the type of data structure (DL Type in Figure 1) used by the DLT. Particular blockchain implementations may be either "permissioned" or "permission-less", which controls who may participate in transactions and/or the consensus protocol in the DLT. For instance, in a permission-less (also known as a public) implementation, any participant can submit transactions to the ledger or be a miner that runs the consensus protocol.

DLT is often used as a platform for cryptocurrencies (the right side of Figure 1), which are a form of digital currency exchange between consumers and providers. Maintaining cryptocurrency accounts on a DLT helps ensure that users cannot lie about the amount of currency that they possess, as well as providing other security guarantees. Crypto-Economic Design (CED) [4, 7, 8] refers to the design of the options and choices permitted within a given DLT-cryptocurrency system, and consists of two main components: the token, or currency unit; and the actions permitted on these tokens. For example, a cryptocurrency may gain value from an underlying physical action (e.g., in a marketplace it is used to trade physical goods or services), or may simply facilitate general types of transactions as in Bitcoin [9].

In addition to cryptocurrencies, some DLT platforms facilitate the exchange of services between multiple parties without requiring trust among participants. This can be achieved by utilizing smart contracts that are hosted on the DLT. A Smart Contract [5] is a set of promises (i.e., rule-based operations to carry out business logic), specified in digital form (i.e., lines of code that prescribe conditions and outcomes encapsulated in a light-weight container that can self-execute), including protocols (i.e., an algorithm that constitutes consensus on how each party should process data in relation to a smart contract) within which the parties perform on these promises using a hosting environment where smart contracts can be hosted and executed.

Several blockchain implementations have been proposed over the last ten years. Below, we present an overview of some of the most popular ones. Interested readers may consult [5] for additional implementations.

Bitcoin is perhaps the most well-known DLT/blockchain technology and was first proposed in 2008 [9]. Bitcoin is a cryptocurrency that runs on top of a decentralized network of miners, who run a consensus protocol to verify transactions. Each miner receives a transaction fee (in Bitcoin) in exchange for helping to verify a block, i.e., a group of transactions. Security guarantees are provided by the requirement that each miner provide a proof-of-work of the block. Bitcoin is mainly used for financial applications (e.g., wallet applications whose sole goal is to help users track their bitcoins [10], or the Lightning payment network that aims to make Bitcoin more scalable [11]).

Ethereum [12] is a blockchain platform that goes beyond cryptocurrencies and financial applications. Ethereum includes native support for decentralized applications and is built on a cryptocurrency called Ether [12]. Current applications built on the Ethereum platform include not only payment applications but also a variety of games, crowd-sourcing platforms, and financial applications [13]. Ethereum is specifically designed to facilitate a variety of applications built on smart contracts, providing virtual machines that allow smart contracts to be executed and a programming language called Solidity [14, 15] for developers to write such applications.

Hyperledger is an open-sourced, modular blockchain platform that aims to allow businesses to easily deploy their own blockchain solutions without developing the technology from scratch [16, 17]. It offers several DLT platforms with different applications in mind, e.g., Hyperledger Fabric focuses on enterprise DLT, such as supply chain applications; while Hyperledger Iroha focuses on mobile and IoT applications; and Hyperledger Indy provides decentralized identity management. All ledgers are cryptocurrency-agnostic and interoperable, with different performance trade-offs.

B. State Channels and Sidechains

Digital marketplaces, in which multiple consumers transact with multiple providers, introduce specific performance requirements for DLT operations. In particular, they must facilitate a large volume of transactions between many different parties, and thus introduce significant scalability challenges that existing platforms like Bitcoin and Ethereum cannot meet. In this section, we review existing scalability solutions and discuss their potential advantages and disadvantages.

State channels are a widely used mechanism that takes advantage of repeated transactions between two users (e.g., exchanges of payment promises) by avoiding the cost of storing all transactions between these users on the blockchain. Instead, some initial overhead is incurred in setting up and depositing funds into a pairwise channel between the users. This channel is then used for subsequent transactions between them (i.e., off the main blockchain), significantly reducing transaction latency. Such channels were first proposed for Bitcoin [18] and are used, e.g., by the Raiden Network [19]. However, pairwise channels are not sufficient for marketplaces where consumers may connect with unknown providers for only a few payments.

A variant on state channels uses *Payment Channels Networks* (PCN), which include Payment Hubs. With PCNs [20–22], payment senders rely on non-custodial intermediaries that provide indirect routes (composed of state channels) to the payment recipient. Thus, even if a pairwise channel is not available between a sender and recipient, a set of channels can be used to transfer the appropriate payment. Though this allows consumers to establish state channels (with locked-in funds) with a limited number of intermediaries in order to pay providers, significant limitations still exist. For instance, in large decentralized marketplaces with sporadical online users, such routes may not exist for some consumer-provider pairs. Further, the intermediaries must commit funds to the route's outgoing channel, which then cannot be used to pay other users. Finally, recent work has exposed incentive compatibility issues with payment intermediaries [23].

Payment Hubs are PCNs where an intermediary is dedicated to providing a 1-hop route between consumers and providers, thus ensuring that a route always exists. In comparison with other PCNs, this facilitates pooled liquidity by allowing consumers to pool their capital (intended for use in marketplace orders) in a single unidirectional channel with the intermediary, e.g. as with Plasma Debit [24] and Plasma Cash [25]. However, other PCN challenges such as incentive compatibility remain an issue. Custodial versions of Payment Hubs allow the intermediary operator to receive consumer payments in dedicated state channels with consumers and periodically initiating root chain transactions to make corresponding payments to each provider. However, this may be inefficient since the number of on-chain transactions grows with the number of providers (receiving payments) every period.

Sidechains offer an alternative to payment channel networks, using the same general idea of moving some transactions off-chain, though utilizing a significantly different method. They enable uses well beyond pairwise payments. Proposed designs [26, 27] are in various stages of completeness and development, some similar to payment channels. Most sidechains are similar to the originally proposed Plasma-style sidechains [28], with Minimum Viable Plasma [29] and its improved version More Viable Plasma [30] as one of the earliest constructions to use periodic root chain notarization of off-chain payment transactions using *short* commitments.

Snappy [31] and PayPlace [32] are two recently proposed protocols for marketplace payments. With Snappy, consumers directly send payments to providers on the root-chain, but are unrestrained by the root-chain's transaction confirmation latency.However, at least one root-chain transaction is made for each payment; hence Snappy does not scale well with the number of transactions. PayPlace does not require a root-chain transaction for each payment, and instead has the operator act as a custodian of funds paid by the consumer to a provider. The funds are released to the provider(s) upon successful verification through the smart contract, which occurs at regular time intervals.

C. Edge Computing

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 [33, 34] . This work has proposed a variety of terms including edge computing [35], fog computing [36], and Mobile Edge Computing (MEC) [37], 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 2) that span the continuum from the cloud to the extreme edge. We define these tiers and list their respective properties below.

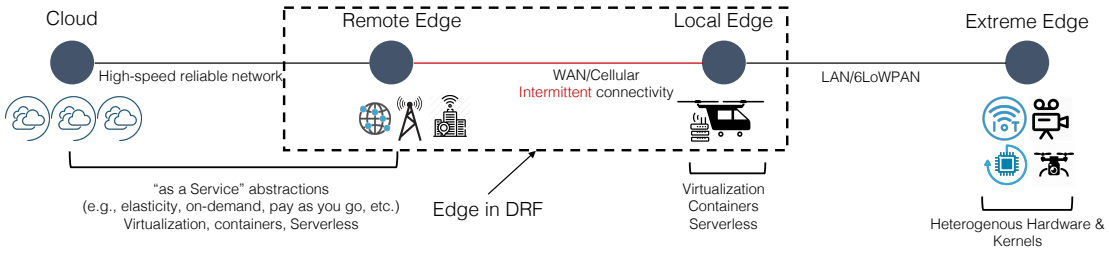

Fig. 2 The four tiers in our system model that form a computing continuum

- **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. 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 [38], 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. Local 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) [37]), refers to resources between the cloud and the edge such as Content Delivery Networks (CDNs), Internet Service Providers (ISP) local stations, network Points of Presence (PoPs), cell tower base stations, and Road Side Units. 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 [39–42] 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.

Edge computing addresses many challenges that are relevant to *DRF*, which can be summarized as follows.

- **Low latency:** *DRF* 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:** *DRF* participants can 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 *DRF* 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 UAVs due to their mobility requirements. The edge presents a suitable platform for critical UAV applications to continue to run despite network failures.

IV. System Architecture

A. Software Architecture Principles

Future air mobility introduce many challenges due to far greater density of heterogeneous vehicles that must operate safely in all geographical areas within the ecosystem. The ecosystem must accommodate routine as well as contingency situations, which translates to far more decisions that are made more rapidly to support the real-time operations of these vehicles. These decisions rely on accurate, reliable, and up-to-date data, which must be made available to a combination of vehicles and human operators.

DRF will allow thousands of small and large enterprises with different assets to participate in creating a data and reasoning marketplace. To ensure that the *DRF* marketplace supports the requirements presented above, we outline the following ten key principals that guide the design of the framework.

1) **Decentralized** – *DRF* adopts a decentralized peer-to-peer architecture to build resilient software for data economy.
2) **Open** – *DRF* is neutral, service and platform agnostic, and user driven to foster innovation by adopting new technology trends.
3) **Sovereign** – *DRF* ensures data and service sovereignty in terms of ownership, control, and exposure of data and reasoning services.
4) **Secure** – security is built into the foundation not an afterthought.
5) **Transparent and Trusted** – *DRF* authenticates participants and provides irrevocable, immutable, transparent, and verifiable transactions, which is available for participants to audit.
6) **Connected** – *DRF* enables a mesh network of data and reasoning services.
7) **Scalable** – *DRF* service providers autonomously deploy and manage services, which will be automatically part of the marketplace.
8) **Governed** – *DRF* includes a policy framework for decentralized collaborative rules.
9) **Accuracy** – *DRF* provides validation mechanisms for data and reasoning services in terms of accuracy, integrity, lineage, and meta data.
10) **Reliability and Timeliness** – *DRF* ensures reliable delivery of low latency messages to support mission critical airspace applications and operational needs.

B. System Overview

The overarching goal of the *DRF* software architecture is to create the necessary mechanisms and abstractions to support complex operations among and within the smart airspace, smart cities, smart aerial vehicles, data services, compute services, and infrastructure services. In order to accomplish this goal, we evaluated and used a range of proven methods and practices to develop and organize the software architecture components. We tracked several architectural patterns and corresponding technologies. Some of these technologies such as centralized data pipelines and platforms (e.g., data lakes, data warehouses, and data brokers) satisfy some of the design requirements. However, based on our industry survey, their centralized approach is not desired, not scalable, and inhibits innovation. Instead, careful analysis of the use cases and the assessment of existing capabilities led us to develop a real-time software architecture with a peer-to-peer decentralized backbone, which is needed for future aviation. The software architecture development was primarily focused on four key drivers: usability, availability, performance, and security.

The *DRF* software architecture is composed of two main components: *DRF* Core and *DRF* Edge as shown in Figure 3. *DRF* Core supports the following functionalities.

- **Identity management**, which leverages a decentralized identity management framework, a participant registry, and a permission graph to ensure proper access to services based on access roles.
- **Transaction management**, which leverages smart contracts to record all transactions for service exchange between service providers and consumers in immutable verifiable logs.
- **Service discovery and monitoring**, which leverages a service catalog to enable consumers to discover the proper services necessary for their operation as well as monitor these services during execution to ensure their correct and safe operation.
- **Service deployment and migration**, which monitors cloud/edge infrastructure and recommends deployment and migration strategies for services providers to meet the SLAs required by service consumers.
- **Policy management**, which regulates the ecosystem, enforces governing policies, and disseminates these policies to participants.

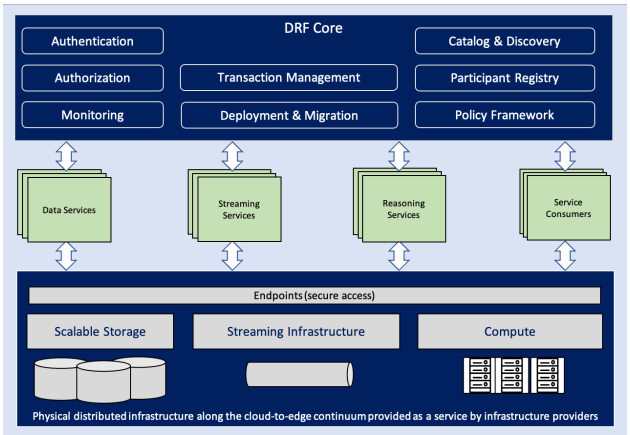

Fig. 3 The *DRF* software architecture connects service providers and consumers using a decentralized peer-to-peer pattern, while maintaining security and recording all transactions in an immutable verifiable log.

DRF Edge allows low latency access to data and reasoning services. Physical infrastructure along the cloud-to-edge continuum, available as a service from infrastructure providers, is used to support data storage and streaming, and computing capabilities necessary for reasoning services. All transactions within the system are secure and encrypted to ensure the safety and integrity of data exchange.

All participants use a portal and/or APIs to register with the marketplace. Data, reasoning, and infrastructure services are included in the service catalog, which service consumers can use to discover available services. Upon discovery, a smart contract is instantiated between a provider and a consumer. The service provider may use the deployment and migration manager to find recommendations for suitable cloud or edge infrastructure for their services. Once a service is deployed, an authenticated data exchange between the provider and consumer is carried out (without *DRF* being in the critical path). Finally, once a consumer acknowledges the receipt of the service, the monetary exchange is complete and the contract is terminated and stored in an immutable verifiable log for future auditing.

C. DRF Core

In this section, we outline in more details the main components of the *DRF* Core and how they operate together to achieve the desired functionality. The overall architecture of the *DRF* Core is presented in Figure 4.

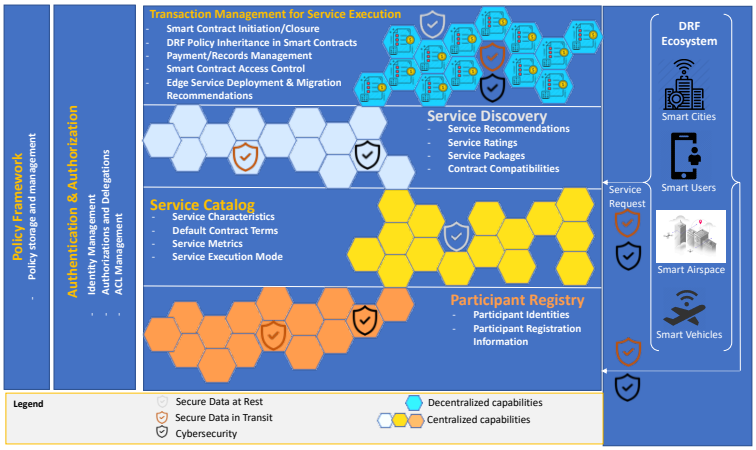

Fig. 4 Overview of the *DRF* Core architecture.

Participant registration and validation. A core functionality of *DRF* is to validate *DRF* participants and maintain secure access to services during service deployment and exchange. To achieve these guarantees, the *DRF* Core includes a decentralized identity management framework and a participant registry.

An *Identity Management Framework* allows participants to create identities and corresponding credentials, which can be used for any interactions with *DRF*. A *Participant Registry* allows both service providers and service consumers to register to become part of the *DRF* marketplace. Together the identity management framework and the participant registry are used to provide the following functionalities:

- *Registration.* Every *DRF* participant registers with *DRF* to ensure only validated users can participate in a *DRF* exchange. Registration utilizes decentralized identifiers (DIDs) that uniquely identify each participant. The participant registry interacts with the *DRF* Policy Framework to enable policy associations in real time while registering a participant based on the participant's permissions.
- *Authentication.* Once registered, *DRF* validates participants whenever an exchange is initiated to ensure only allowed users can participate using the decentralized identifiers provided at registration.
- *Authorization and Delegation.* Similarly, *DRF* maintains and enforces access control for participants according to policies and predefined agreements. It also supports delegation of access to authorized services across multiple administrative domains when necessary.

Service registration and discovery. *DRF* Core service discovery allows service consumers to identify and find service provides. To achieve these goals, *DRF* includes service registries, service catalog, and a smart discovery mechanism.

Service Registries. The *DRF* service registries enable service providers (e.g., data, reasoning, cloud/edge infrastructure) to register services and define their characteristics such as performance, deployment environments, deployment constraints, price models, infrastructure requirements, availability, contract definitions, applicable policy definitions, and decentralized identifiers. The service registries leverage APIs to ingest characteristic information provided by service providers. The registries also provide health check capabilities for service providers.

Service Catalog. The service catalog is driven by intuitive user interfaces that allow users to efficiently navigate the characteristics of services and its associations with policies managed by governing entities using the policy management framework. The service catalog acts as a knowledge base for users to navigate and query public information associated with *DRF* registered services. The policy framework enforces that private information in the catalog are only accessible with proper access.

Smart Service Discovery. There are different approaches to providing service discovery. In consumer-side discovery, consumers are responsible for finding the necessary service by downloading and searching through the catalog. In marketplace-side discovery, the marketplace operator is responsible for determining the necessary service based on the consumer's request. *DRF* will implement a hybrid approach where *DRF* provides smart search capabilities that narrows down the search results. These results are then passed to the consumer who is responsible for determining the necessary service. The service discovery capability is fully automated to understand service consumers' requirements and provide the consumers with options based on their needs. Each option provided by the service discovery may include multiple services bundled as a package to meet the consumer needs. The service discovery will include smart algorithms to identify compatibility between different service providers to bundle them together based on the service providers' characteristics. The service discovery exposes APIs for consumers to identify and discover service packages.

Transaction management. All transactions and agreements for service exchange are recorded in *DRF* using immutable verifiable logs. Further, given *DRF*'s need to facilitate multiple types of applications, smart contracts are a particularly promising solution for consumer-provider interactions. In particular, *DRF* will utilize smart contracts to programmatically enforce and guarantee that service exchanges are being carried out according to prearranged agreements.

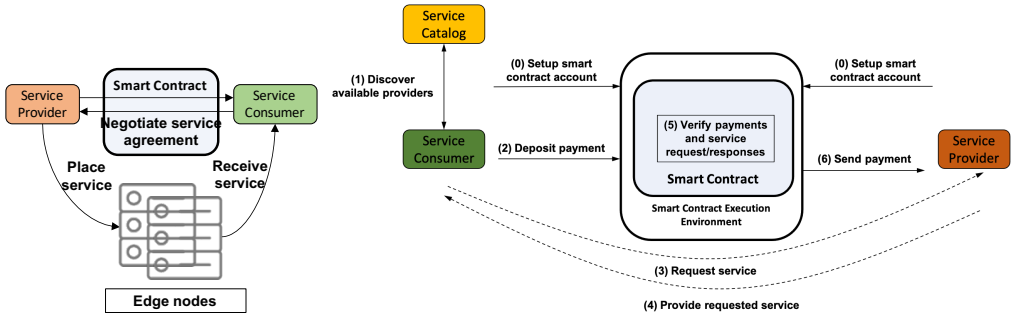

Fig. 5 An overview of a consumer-provider smart contract negotiation sequence.

Smart contracts can be used to negotiate the terms of agreement for service exchange between a service provider and a service consumer. Moreover, these contracts can decouple the process of providers' requesting compute resources and negotiating an agreement to provide services to consumers. Figure 5 gives a high-level overview of this process. We next outline the steps required to negotiate a smart contract. Note that, prior to doing so, both the service providers and service consumers should register as participants with *DRF*and setup their smart contract account (Step 0 in Figure 5).

There are three types of entities present in the consumer-provider negotiation: a provider, a consumer, and *DRF* as a marketplace operator. Multiple consumers and providers may be present in a single smart contract. However, for simplicity, we consider a single provider and single consumer below to outline the sequence of steps needed to establish an agreement. The role of *DRF* is to facilitate interactions between consumers and providers, as well as provide recommendations on which cloud or edge resources a provider should use. As a marketplace operator, *DRF* is the entity in charge of smart contract management (i.e., storing and executing the contracts). Figure 5 shows an example sequence of interactions between these entities, which we elaborate on below.

1) *Consumer discovers available providers.* The first step for service exchange is for consumers to request a particular service, which *DRF* smart service discovery mechanisms would facilitate using the service catalog.
2) *Consumer requests service.* Upon receiving a list of potential providers from *DRF*, the consumer chooses one or more and sends service requests to the chosen providers. Upon receiving the request, the provider can negotiate the payment amount or terms with the consumer. The format and protocol underlying the negotiation is out of the *DRF* scope; however, the provider may query the *DRF* service catalog as to whether sufficient compute resources are available during the negotiation. A successful negotiation concludes with the deployment of one or more smart contracts (between the consumer, the service provider, and the infrastructure (e.g, edge) provider).
3) *Provider provides service, and consumer pays.* Once an agreement has been reached, the provider arranges with the edge provider to obtain access to edge resources, and provides the agreed-on service. The service provider sends a payment to the edge provider and the consumer sends a payment to the service provider in return. These payments may take place before, after, or during the service exchange. A common technique to limit the risk for the consumer or provider is to use micro-payments, in which incremental amounts of service are provided in exchange for incremental payments [32]. Thus, if a provider stops providing the service or a consumer stops paying, the other party can break off the transaction without significant losses. Micro-payments may not be viable for certain types of applications in *DRF*, or when multiple providers or consumers are involved. Reputation systems [43] or verification from third-parties may instead be used to verify service receipt.
4) *Smart contract verifies payments and service requests.* After or at periodic intervals during the consumer-provider interaction, the smart contract must verify the payments and service requests. The exact mechanisms of this interaction depend on the marketplace technology used. If the consumer and provider are connected through payment channels, the verification can happen immediately. If a sidechain solution is used, then periodic notarization will be required, each of which will require a verification step.
5) *Provider receives payment.* Upon successful verification, the provider receives payment and the transaction concludes.

Policy framework. A policy framework allows governing entities to regulate access and disseminate policy updates to participants. A policy defines a set of permissions associated with a role enforced on a group of *DRF* registered services. Roles are assigned to systems managed by governing bodies. The *DRF* Policy Framework will have machine-to-machine adapters to interface with these systems. Service consumers can search and retrieve applicable enforced policies and associated documents to use *DRF* registered services.

Permissions and Discovery Policies. The policy framework will define a set of permissions associated with a role enforced on a group of *DRF* registered participants. The roles are assigned to participating systems and are managed by governing bodies. This can be achieved using a pub-sub model, where participants can search and subscribe to retrieve applicable policies and associated documents.

Distributing Policies. The *DRF* policy framework will communicate and distribute policies across the *DRF* ecosystem such that participants in the system receive and retain up-to-date policies relevant to them and their roles in the airspace system. The participants ingesting these policies should be able to use the *DRF* policy framework to correctly interpret the policy and their role in its execution. The *DRF* policy framework should likewise be able to track which participants are part of which policies.

Executing Policies. The *DRF* policy framework will communicate to participants the necessary information or signaling regarding the execution of policies, when policies are put into effect, external conditions or circumstances that have a bearing on how policies are executed (some policies may have different operating modes depending on external conditions), and other group signaling necessary to support policy execution.

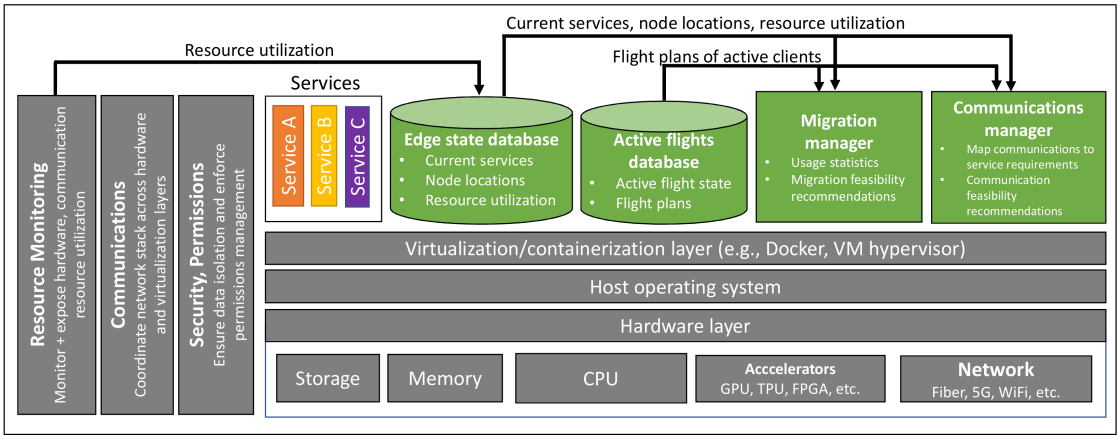

Fig. 6 Edge node architecture.

Feedback Loops. The *DRF* policy framework will include support for establishing feedback loops as part of policies such that execution parameters related to policies can be measured and monitored across the system. These feedback loops will ensure that the characteristics or performance of execution of a policy is understood by other system elements, whether under nominal or off-nominal conditions or whether the policy is executing within intended performance parameters or whether execution is deviating due to some factor. With such feedback loops, the overall system has the facility to implement adaptive measures and better ensure safe and effective operations.

Governing Entities. The policy framework will allow governing entities (e.g., FAA) to regulate access and disseminate policy updates to participants. This can be achieved using machine-to-machine adapters to interface with existing (e.g., ARNIC) or new systems.

Monitoring. *DRF* will also monitor data, reasoning, and infrastrtucture services performance. In particular, *DRF* will monitor reasoning service provider performance and SLAs, which may include response time, availability, and consumer ratings. *DRF* consumers will have the ability to rate reasoning services to ensure their correct operations according to the established agreements. *DRF* will also monitor data service performance as well as data quality. Initially *DRF* will rely on consumers to validate data by allowing them to provide feedback and ranking. *DRF* will also allow trusted third party data validator services. *DRF* will monitor infrastructure performance and availability. Data and reasoning providers can rate infrastructure providers based on the delivered services. Finally, *DRF* will utilize this information to provide smart service discovery based upon consumer feedback as well as third party validations.

Part of the monitoring of *DRF* related services and infrastructure, is the establishment of a framework that can exchange metadata for monitoring system health, operations execution, and safety. This is an important closed feedback loop for the safe operation of UASs. In particular, the data generated and made available by the *DRF* monitoring framework will allow airspace system operations managers and regulators to much better understand performance of the airspace system. This can improve robustness and safety, support more automated air mobility flight operations, and make positive enhancements to policies based on real data and operating experience.

D. DRF Edge

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 more advanced navigation features.

DRF does not directly 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 edge providers and *DRF* data and reasoning 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). Infrastructure providers for these types of resources can opt in to install *DRF* libraries, which will allow *DRF* to monitor and provide access control to these resource.

Note – *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.

Edge Architecture. The architecture of an edge node is shown in Figure 6. 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 6. 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 to the virtualization layer.
- *Hardware layer:* The hardware layer consists of storage, memory, CPU, any accelerators (*e.g.,* GPU, TPU, FPGA), 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 6. 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* marketplace 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 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.
- *Active Flights Database:* The purpose of this database is to maintain information about which 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 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 vehicle. For example, if a consumer located on a vehicle enters an urban area with unreliable 5G signals due to wireless interference, the Communications Manager can recommend the vehicle switch to a higher-reliability but lower throughput network protocol.

Note – the outputs of the migration and communications managers are simply provided as recommendations to providers. Also note that a *DRF* implementation could function without the Migration Manager and Communications Manager, but their operation can enhance the overall *DRF* system performance by assisting providers to satisfy their contracts with consumers.

V. Example Use Case Scenarios

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

A. 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 *DRF* provider that has cameras close to the water surface, providing better visibility. Another *DRF* 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 [44], hence the need for an edge node. Figure 7 shows an example of the interactions between the consumer (the vehicle) and the camera capture and machine learning providers. These steps are:

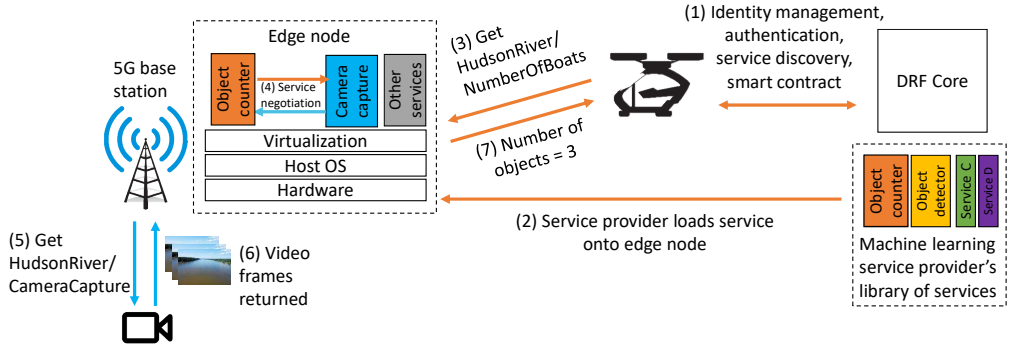

Fig. 7 An edge deployment example for an emergency water landing.

1) *Identity management, authentication, service discovery, smart contract:* The consumer learns about the available providers and negotiates a contract with a machine learning (ML) service provider. Similarly the ML provider searches for available camera providers and edge infrastructure providers and negotiates contracts with a camera and an edge infrastructure service providers.
2) *Service provider loads service onto an edge node:* The ML service provider 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 ML service provider (shown in orange) starts 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 [45] 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.

B. 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 an ML 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 8 shows an example of the interactions between the consumer (the vehicle) and the ML provider.

We assume that the consumer-provider relationship has already been established (similar to Section V.A). Having finished searching the area around vertipad 1 with a negative results, the UAV begins to navigate to its next waypoint at vertipad 2. However, vertipad 2 is far from edge node A where the ML 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:

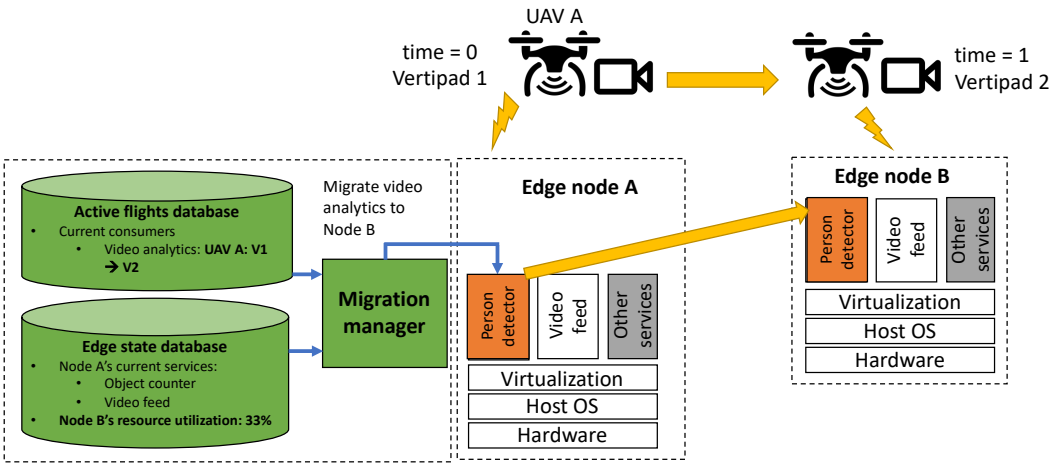

Fig. 8 An edge migration example for an aerial search and rescue mission.

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.

VI. Related Work

In section III, we gave a brief background on some of the technologies relevant to the *DRF* software architecture. In this section, we cover some of the related work in using blockchains to provide a decentralized marketplace or using edge computing to support autonomous vehicles and smart cities.

A. Blockchain-based Marketplaces

Blockchain-based marketplace solutions have been proposed or deployed for a wide variety of marketplaces, including platforms for computing [47, 48] and network connectivity [49, 50]. Below, we review the literature on two marketplaces particularly relevant to *DRF*: mobility and Internet-of-Things (IoT) applications.

Mobility Applications. While mobility-based blockchain applications today largely focus on autonomous vehicles (AVs), unmanned aerial vehicles (UAV) may face similar challenges. This work advocates the use of payment channels [51] and pegged sidechains [52] to address scalability challenges such as the need to support large number of transactions and multiple applications, potentially running on different blockchains. [51] provides a detailed architecture of how payment channels can be used for electric charging, which introduces latencies in the order of a few minutes. [52] proposes the use of pegged sidechains to provide reliable and secure peer-to-peer communication in robotic swarms. They use multi-signatures to enforce collaboration and multiple side chains if the swarms must execute different behaviors.

Blockchains can also be used for autonomous intersection management. [53] proposes using blockchains to ensure private, authenticated, reliable, and scalable exchange of information between an intersection manager and vehicles. The intersection manager can run a smart contract to optimally schedule traffic lights to maximize the flow of traffic, which requires data on where vehicles are going.

More broadly, intelligent transportation systems can leverage blockchains to mitigate the security risks of centralized data sharing, facilitating a decentralized data architecture [54]. Like the typical networking stack, the authors propose a seven layer model from application to physical layers (including contract, incentive, consensus, network, and data layers). Vehicles can then act as consensus nodes with data on transportation logistics directly stored in the blockchain.

Data and IoT Applications. Securing data collected by IoT devices presents a significant challenge due to the number of devices and scale of the collected data. *DRF* faces a similar challenge, although instead of simply providing data, *DRF* providers may offer a richer array of services, e.g., including analysis of this data, which requires not only data storage but also edge computing resources to run the analysis algorithms. Blockchains can facilitate shared access to data [55] and storing transactions on this data between devices. Encryption and signature verification may be used to provide security guarantees.

Smart contracts have been proposed as a means to set up marketplaces for IoT data [43, 56–58], since they provide a natural way to handle access control policies and remuneration requests in a decentralized manner. Typically, such architectures utilize an off-chain provider to store the encrypted data, and separate key authorities manage the data access keys so as to prevent malicious activity from the data provider [59]. Transactions on the smart contract are thus separated from the physical exchange of data to reduce latency.

In [43], providers can publish data offerings on the blockchain, where consumers can evaluate them and decide whether to purchase each piece of data. Periodic check-pointing deters double-spend attacks. A reputation model that runs on the blockchain helps participants assess the risk of entering into agreements with each other, and a similar mechanism might be used by *DRF* to help enforce consumer-provider agreements. An Ethereum implementation shows that the data transfers take only a few seconds, although it is unclear how well this solution scales.

An alternative model is proposed in [58], which puts the access policy logic on the blockchain smart contract, ensuring that data access is auditable and that data owners are correctly remunerated for access to their data. As in [43], providers make data offers on the blockchain and consumers can decide whether to accept them. Storage providers act as blockchain nodes enforcing these access policies, and data owners control the access keys, which may also be distributed among multiple entities for further security guarantees. A Hyperledger Fabric implementation shows that data transfers in this architecture take only a few seconds, scaling well with the data size. However, it is unclear how this architecture scales with the number of nodes, as the implementation only looked at five data owners.

B. 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 [60]. 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) [61, 62], satisfy strict energy constraints on battery-powered AVs, and provide security for all layers of the AV stack to avoid compromising passenger safety [62].

Edge computing can provide a natural coordination point that can minimize the amount of communication, offload computation, and improve cooperation among vehicles by taking into account locally aggregated vehicle data, road conditions, smart-city data, and infrastructure sensors. In the remainder of this section, we review existing testbed deployments and activities in edge computing for AVs.

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 [61]. In 2019, Amazon (through a partnership with Verizon) and Microsoft (through a partnership with AT&T) announced commercial offerings for edge computing [39, 40]. Similarly, automakers have also started exploring edge computing as a potential complement to their vehicles, with Porsche [63] recently citing several prototypes of edge computing use cases for their products.

Some companies have begun pilot AV deployments. Despite a lack of edge servers, Waymo [64], Cruise [65], and Tesla [66] 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 in *DRF* 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 [67]. The farming industry is an early adopter of autonomous equipment for use in private farms [68]. 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 supercenters, effectively creating a highly geo-distributed cloud [69]. Such infrastructure could be exploited by AVs, as both Walmart stores and AVs would likely be concentrated in relatively highly populated areas [70].

C. Edge Computing for Smart Cities

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 *DRF*, 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: SAGE [71] (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 [72] 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 even predict wildfires, and provide this information to the public through a web-based interface. 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. The multiple sensors run by different administrators (*e.g.,* National Park Service weather, San Diego Gas & Electric cameras) are examples of data service providers in *DRF*.

Civic deployments. Alongside these research prototypes, there are several civic deployments that are more tightly integrated with city operations. SmartCity PHL [73] is deployed in Philadelphia, and covers a wide variety of use cases, including snowplow monitoring, HVAC control in government buildings, language translation to communicate with city residents, and more. Toronto Quayside [74] 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 [75] 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, etc.). 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 [76] which establishes guidelines for privacy and transparency [73, 77].

General edge platforms. There are also several generic edge platforms that are designed for low-power sensors, which thus may be applicable to the smart city scenario. Apache Edgent [78] 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 [79] 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 [80]. 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.

VII. Conclusion

In this paper, we presented an overview of the software architecture for the Data & Reasoning Fabric, a digital data services marketplace for Advance Air Mobility. Further, this architecture also supports future air mobility for traditional manned air crafts. The architecture leverages a decentralized peer-to-peer design pattern to support secure and low latency exchange of all data and reasoning services necessary for future air mobility. These exchanges are monitored and recorded in a transparent immutable verifiable log. The services are deployed across the cloud-to-edge continuum to support real-time requirements. We have also presented two driving examples that demonstrate how the system components work together to achieve the desired results.

Distributed ledger technology has come a long way since the introduction of Bitcoin in 2008. While there are inherent limitations to its usage, active research in academia and industry is focused on addressing these limitations [5]. Within *DRF*, DLTs present an opportunity for providing an immutable verifiable log for transaction management in the marketplace. Research on improving security and scalability of blockchains will only improve the performance of the *DRF* software architecture.

Similarly, while the vision for edge computing has been around for a few years, only recently did this vision start materializing, with the introduction of commercial edge computing offerings from large scale providers. As these offerings become more ubiquitous and available, they will strengthen the *DRF* software architecture by enabling low-latency access to data and reasoning services.

Acknowledgments

The DRF activity is funded by the NASA Aeronautics Research Mission Directorate (ARMD) Transformative Aeronautics Concepts Program (TACP) Convergent Aeronautics Solutions (CAS) Project.

References

[1] "Urban Air Mobility Concept of Operations," , 2021. `https://nari.arc.nasa.gov/sites/default/files/attachments/UAM_ConOps_v1.0.pdf`.

[2] Van Dalsem, W., Shetye, S., Das, A. N., Krishnakumar, K. S., Lozito, S., Freeman, K., Swank, A., Shannon, P., and Tomljenovic, L., "A Data & Reasoning Fabric to Enable Advanced Air Mobility," *AIAA Scitech 2021 Forum*, 2021, p. 2033. https://doi.org/10.2514/6.2021-2033.

[3] Van Dalsem, W., Freeman, K., Shetye, S., Das, A., Krishnakumar, K., Swank, A., Shannon, P., Tomljenovic, L., Lozito, S., Euler, H., Kaur, S., Wong, C., and O'Brien, J., "Data & Reasoning Fabric Minimum Viable Product," *AIAA Aviation 2021 Forum*, 2021, to appear.

[4] Ballandies, M. C., Dapp, M. M., and Pournaras, E., "Decrypting distributed ledger design-taxonomy, classification and blockchain community evaluation," *arXiv preprint arXiv:1811.03419*, 2018.

[5] Kolb, J., AbdelBaky, M., Katz, R. H., and Culler, D. E., "Core Concepts, Challenges, and Future Directions in Blockchain: A Centralized Tutorial," *ACM Computing Surveys (CSUR)*, Vol. 53, No. 1, 2020, pp. 1–39.

[6] Fischer, M. J., Lynch, N. A., and Paterson, M. S., "Impossibility of distributed consensus with one faulty process," *Journal of the ACM (JACM)*, Vol. 32, No. 2, 1985, pp. 374–382.

[7] Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., and Felten, E. W., "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," *2015 IEEE symposium on security and privacy*, IEEE, 2015, pp. 104–121.

[8] Tschorsch, F., and Scheuermann, B., "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Communications Surveys & Tutorials*, Vol. 18, No. 3, 2016, pp. 2084–2123.

[9] Nakamoto, S., "Bitcoin: A peer-to-peer electronic cash system," Tech. rep., Working Paper, 2008.

[10] Hurst, A., "The best Bitcoin apps," Information Age, 2019. https://www.information-age.com/best-bitcoin-apps-123467034/.

[11] Poon, J., and Dryja, T., "The bitcoin lightning network: Scalable off-chain instant payments," , 2016.

[12] Wood, G., et al., "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, Vol. 151, No. 2014, 2014, pp. 1–32.

[13] ConsenSys, "90+ Ethereum Apps You Can Use Right Now," ConsenSys Blog, 2019. https://consensys.net/blog/news/90-ethereum-apps-you-can-use-right-now/.

[14] "Solidity Docs," https://solidity.readthedocs.io/en/v0.5.13/, ???? Accessed: 2019-12-07.

[15] Dannen, C., *Introducing Ethereum and solidity*, Vol. 1, Springer, 2017.

[16] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al., "Hyperledger fabric: a distributed operating system for permissioned blockchains," *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1–15.

[17] Cachin, C., et al., "Architecture of the hyperledger blockchain fabric," *Workshop on distributed cryptocurrencies and consensus ledgers*, Vol. 310, 2016.

[18] Decker, C., and Wattenhofer, R., "A fast and scalable payment network with bitcoin duplex micropayment channels," *Symposium on Self-Stabilizing Systems*, Springer, 2015, pp. 3–18.

[19] "Raiden Network," Tech. rep., Brainbot Labs Est., ???? URL https://raiden.network/101.html.

[20] Miller, A., Bentov, I., Kumaresan, R., Cordi, C., and McCorry, P., "Sprites and state channels: Payment networks that go faster than lightning," *arXiv preprint arXiv:1702.05812*, 2017.

[21] Sivaraman, V., Venkatakrishnan, S. B., Alizadeh, M., Fanti, G., and Viswanath, P., "Routing cryptocurrency with the spider network," *arXiv preprint arXiv:1809.05088*, 2018.

[22] Dziembowski, S., Eckey, L., Faust, S., and Malinowski, D., "PERUN: Virtual Payment Channels over Cryptographic Currencies." *IACR Cryptology ePrint Archive*, Vol. 2017, 2017, p. 635.

[23] Béres, F., Seres, I. A., and Benczúr, A. A., "A cryptoeconomic traffic analysis of bitcoins lightning network," *arXiv preprint arXiv:1911.09432*, 2019.

[24] "Plasma Debit: Arbitrary Denomination Payments in Plasma Cash," https://ethresear.ch/t/plasma-debit-arbitrary-denomination-payments-in-plasma-cash/2198, ???? Accessed: 2019-12-07.

[25] "Plasma Cash: Plasma with Much Less Per User Data Checking," https://ethresear.ch/t/plasma-cash-plasma-with-much-less-per-user-data-checking/1298, ???? Accessed: 2019-12-07.

[26] Johnson, S., Robinson, P., and Brainard, J., "Sidechains and interoperability," *arXiv preprint arXiv:1903.04077*, 2019.

[27] Adler, J., and Quintyne-Collins, M., "Building Scalable Decentralized Payment Systems," *arXiv preprint arXiv:1904.06441*, 2019.

[28] Poon, J., and Buterin, V., "Plasma: Scalable autonomous smart contracts," *White paper*, 2017, pp. 1–47.

[29] "Minimum Viable Plasma," https://ethresear.ch/t/minimal-viable-plasma/426, ???? Accessed: 2019-12-07.

[30] "More Viable Plasma," https://ethresear.ch/t/more-viable-plasma/2160, ???? Accessed:2020-07-30.

[31] Mavroudis, V., Wüst, K., Dhar, A., Kostiainen, K., and Capkun, S., "Snappy: Fast On-chain Payments with Practical Collaterals," *arXiv preprint arXiv:2001.01278*, 2020.

[32] Harishankar, M., Iyer, S. V., Laszka, A., Joe-Wong, C., and Tague, P., "PayPlace: A Scalable Sidechain Protocol for Flexible Payment Mechanisms in Blockchain-based Marketplaces," 2020.

[33] Yi, S., Li, C., and Li, Q., "A survey of fog computing: concepts, applications and issues," *Proceedings of the 2015 workshop on mobile big data*, 2015, pp. 37–42.

[34] Mahmud, R., Kotagiri, R., and Buyya, R., "Fog computing: A taxonomy, survey and future directions," *Internet of everything*, Springer, 2018, pp. 103–130.

[35] Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L., "Edge computing: Vision and challenges," *IEEE internet of things journal*, Vol. 3, No. 5, 2016, pp. 637–646.

[36] Bonomi, F., Milito, R., Zhu, J., and Addepalli, S., "Fog computing and its role in the internet of things," *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13–16.

[37] Hu, Y. C., Patel, M., Sabella, D., Sprecher, N., and Young, V., "Mobile edge computing—A key technology towards 5G," *ETSI white paper*, Vol. 11, No. 11, 2015, pp. 1–16.

[38] Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N., "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, Vol. 8, No. 4, 2009, pp. 14–23.

[39] "Amazon AWS Wavelength," , ???? https://aws.amazon.com/wavelength/.

[40] "Microsoft Azure Edge Zones," , ???? https://azure.microsoft.com/en-us/solutions/low-latency-edge-computing/.

[41] "Fastly Serverless at the edge," , ???? https://www.fastly.com/products/edge-compute/beta.

[42] "CloudFlare Workers," , ???? https://workers.cloudflare.com.

[43] Bajoudah, S., Dong, C., and Missier, P., "Toward a Decentralized, Trust-less Marketplace for Brokered IoT Data Trading using Blockchain," *2019 IEEE International Conference on Blockchain (Blockchain)*, IEEE, 2019, pp. 339–346.

[44] Onoro-Rubio, D., and López-Sastre, R. J., "Towards perspective-free object counting with deep learning," *European Conference on Computer Vision*, Springer, 2016, pp. 615–629.

[45] Cesetti, A., Frontoni, E., Mancini, A., Zingaretti, P., and Longhi, S., "A vision-based guidance system for UAV navigation and safe landing using natural landmarks," *Journal of intelligent and robotic systems*, Vol. 57, No. 1-4, 2010, p. 233.

[46] Szegedy, C., Toshev, A., and Erhan, D., "Deep neural networks for object detection," *Advances in neural information processing systems*, 2013, pp. 2553–2561.

[47] Golem, "Golem," , 2020. https://docs.golem.network/#/.

[48] Edge Network, "Network Architecture," , 2020. https://edge.network/en/architecture/.

[49] Nodle, "The Nodle Network: A New Economic Model to Free the Mobile Internet," Tech. rep., Nodle, ???? URL https://docsend.com/view/gjtn4jc.

[50] S. Cannell, J., Sheek, J., Freeman, J., Hazel, G., Rodriguez-Mueller, J., Hou, E., and J. Fox, B., "Orchid: A Decentralized Network Routing Market," Tech. rep., Orchid Labs, 2019. URL https://www.orchid.com/assets/whitepaper/whitepaper.pdf.

[51] Pedrosa, A. R., and Pau, G., "ChargeltUp: On blockchain-based technologies for autonomous vehicles," *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, 2018, pp. 87–92.

[52] Ferrer, E. C., "The blockchain: a new framework for robotic swarm systems," *Proceedings of the future technologies conference*, Springer, 2018, pp. 1037–1058.

[53] Buzachis, A., Celesti, A., Galletta, A., Fazio, M., and Villari, M., "A secure and dependable multi-agent autonomous intersection management (MA-AIM) system leveraging blockchain facilities," *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, IEEE, 2018, pp. 226–231.

[54] Yuan, Y., and Wang, F.-Y., "Towards blockchain-based intelligent transportation systems," *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2016, pp. 2663–2668.

[55] Singh, M., Singh, A., and Kim, S., "Blockchain: A game changer for securing IoT data," *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, IEEE, 2018, pp. 51–55.

[56] Shafagh, H., Burkhalter, L., Hithnawi, A., and Duquennoy, S., "Towards blockchain-based auditable storage and sharing of iot data," *Proceedings of the 2017 on Cloud Computing Security Workshop*, 2017, pp. 45–50.

[57] Kaaniche, N., and Laurent, M., "A blockchain-based data usage auditing architecture with enhanced privacy and availability," *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*, IEEE, 2017, pp. 1–5.

[58] Truong, H. T. T., Almeida, M., Karame, G., and Soriente, C., "Towards secure and decentralized sharing of IoT data," *2019 IEEE International Conference on Blockchain (Blockchain)*, IEEE, 2019, pp. 176–183.

[59] Yakubov, A., Shbair, W., Wallbom, A., Sanda, D., et al., "A blockchain-based pki management framework," *The First IEEE/IFIP International Workshop on Managing and Managed by Blockchain (Man2Block) colocated with IEEE/IFIP NOMS 2018, Tapei, Tawain 23-27 April 2018*, 2018.

[60] Sabella, D., Moustafa, H., Kuure, P., Kekki, S., Zhou, Z., Li, A., Thein, C., Fischer, E., Vukovic, I., Cardillo, J., Young, V., Tan, S. J., Park, V., Vanderveen, M., Runeson, S., and Sorrentino, S., "Toward fully connected vehicles: Edge computing for advanced automotive communications," 5G Automotive Association, 2017.

[61] Sverdlik, Y., "Do Driverless Cars Really Need Edge Computing?" DataCenter Knowledge, 2019. https://www.datacenterknowledge.com/edge-computing/do-driverless-cars-really-need-edge-computing.

[62] Liu, S., Liu, L., Tang, J., Yu, B., Wang, Y., and Shi, W., "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, Vol. 107, No. 8, 2019, pp. 1697–1716.

[63] Hub, M., "Seven Edge Computing Use Cases for Vehicles," Porsche Newsroom, 2019. https://newsroom.porsche.com/en/2019/digital/porsche-edge-computing-19509.html.

[64] Waymo, "Be an Early Rider," , 2020. https://waymo.com/apply/.

[65] Hawkins, A. J., "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.

[66] Hawkins, A. J., "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.

[67] Hawkins, A. J., "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.

[68] Linthicum, D., "Where autonomous vehicles and edge computing have arrived," InfoWorld, 2020. https://www.infoworld.com/article/3518784/where-autonomous-vehicles-and-edge-computing-have-arrived.html.

[69] Gray, P., "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/.

[70] Eliot, L., "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-questions/.

[71] "Sage: Cyberinfrastructure for AI at the Edge," http://sagecontinuum.org/, ????

[72] "WIFIRE: Workflows Integrating Collaborative Hazard Sciences," https://wifire.ucsd.edu/, ????

[73] "SmartCity PHL Roadmap," https://www.phila.gov/media/20190204121858/SmartCityPHL-Roadmap.pdf, ????.

[74] "Quayside - Sidewalk Toronto," https://www.sidewalktoronto.ca/plans/quayside/, ????

[75] "Smart City Vision | City of San Jose," https://www.sanjoseca.gov/your-government/departments/information-technology/smart-city-vision, ????

[76] "NYC Guidelines for the Internet of Things," https://iot.cityofnewyork.us/, ????

[77] "Smart City PDF," https://www.smartcitypdx.com/, ????.

[78] "Apache Edgent," https://edgent.incubator.apache.org/, ????

[79] "Akraino Edge Stack," https://www.lfedge.org/projects/akraino/, ????

[80] Liu, F., Tang, G., Li, Y., Cai, Z., Zhang, X., and Zhou, T., "A survey on edge computing systems and tools," *Proceedings of the IEEE*, Vol. 107, No. 8, 2019, pp. 1537–1562.