The Just-In-Time Cloudlet

James Blakley, Marc Meunier[†], Thomas Eiszler, Jan Harkes,
Mahadev Satyanarayanan

[†]Arm, Inc.

October 2023
CMU-CS-23-138

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work supported by the U.S. Army Research Office and the U.S. Army Futures Command under Contract No. W519TC-23-C-0003, and by the National Science Foundation under grant number CNS-2106862. The content of the information does not necessarily reflect the position or the policy of the government and no official endorsement should be inferred. This work was done in the CMU Living Edge Lab, which is supported by Intel, Arm, Vodafone, Deutsche Telekom, CableLabs, Crown Castle, InterDigital, Seagate, Microsoft, the VMware University Research Fund, IAI, and the Conklin Kistler family fund. Any opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the view(s) of their employers or the above funding sources.

Keywords: edge computing, mobile networks, latency, bandwidth, edge-native applications, virtual machines, virtual desktop infrastructure

Abstract

Using drones to inspect bridges and electric grid towers, mounting a search and rescue operation for a child lost in the woods, hosting a backcountry ski event for a weekend, and providing critical communication and application capabilities for forward military operations – These are just a few situations where there is a need to deploy a modern compute and communication infrastructure very rapidly in areas with poor network coverage. These situations require the ability to provide connectivity for mobile phones, cameras, drones, and other connected devices. These devices need to offload compute-intensive operations that cannot be performed on-device. They also require access to sophisticated applications, possibly proprietary, with too large a footprint to even run partially on-device. Until recent breakthroughs in a wide variety of open technologies, implementing these use cases has required custom hardware and software, proprietary network technologies, communication spectrum licenses, and the expertise to integrate a solution. These breakthroughs enable a new model – the *Just-in-Time (JIT) Cloudlet* – a highly integrated, lightweight, low power platform built from common off-the-shelf (COTS) technologies that provides a standalone private mobile network provisioned with applications for specific use cases. This technical report describes the motivation, requirements, and enablers to build a JIT Cloudlet. It also describes our architecture, implementation, and results for a JIT Cloudlet prototype created at the Carnegie Mellon University Living Edge Lab. This prototype, based around open-source software and COTS hardware can be the base for development platform for application developers and integrators to deliver solutions for the use cases above and many others.

1 Introduction

There has long been a need for telecommunications and internet services in locations far removed from urban environments where connectivity to the global communications network is limited. Small villages and remote temporary events are often unprofitable for telecom operators to serve. But, closed proprietary technologies, unavailable or expensive spectrum, and a lack of expertise make it difficult for "do-it-yourselfers" to create solutions to serve these needs. And, centralization of meaningful services in distant cloud data centers requires not just connection but also high bandwidth. These forces place many potential users in a "digital desert". We believe that a confluence of new technologies, many mature enough for production deployment, have made it feasible for many services to be provided by highly *integrated* and *portable* communications network and application infrastructure.

While there are broad needs for better digital access in places where it is inadequate today, given our focus on edge computing, we have paid particular attention to cases where limited mobile network access and backhaul to the cloud calls for deployment of a low-cost localized mobile network and computing infrastructure. We have also focused on cases where the deployment is intended to be temporary and portable – putting a high premium on small physical footprint, low weight, low power consumption, and fast setup. We believe that these conditions represent some of the most extreme requirements for platform integration in edge computing.

We envision a world where an application developer, system integrator, or enterprise can easily create a turnkey solution that integrates a small-scale private mobile network and an edge-native application suite that addresses an application-specific set of use cases and can be deployed in an environment with limited access or backhaul networks. These deployments must be inexpensive, temporary, and fast to bring online. Solutions should be built on commerical off-the-shelf (COTS) hardware and open-source software using the latest cloud-native and AI technologies.

A system that is designed to be deployed in these situations we refer to as a *Just-in-Time (JIT) Cloudlet*. A cloudlet can be viewed as a "Data Center in a Box" whose goal is to "bring the cloud closer" to the end devices [41]. A JIT Cloudlet adds a bundled mobile network and tight physical constraints on the cloudlet footprint – these two additions turn a cloudlet into an "Edge Computing Hotspot" suitable for quick deployment in constrained environments. This introduction motivates why JIT Cloudlet solutions are more feasible than ever before and are poised for rapid adoption over the coming few years.

The rest of this document provides the details around the JIT Cloudlet solution developed in the Living Edge Lab (LEL) [3] at Carnegie Mellon University. It outlines the solution design criteria, describes the building block technologies used, presents a solution reference architecture and the prototype built to implement the reference architecture including performance results and lessons learned. It closes with a discussion of future work and opportunities for our program and for the industry as a whole to achieve broad JIT Cloudlet adoption.

1.1 Background

Both edge computing and private mobile networks are growing in industry interest and adoption [14] [28]. However, despite the obvious synergies between the two, the integration of their respective infrastructures has received little attention. In addition, much focus in both areas has been on permanent, relatively large-scale infrastructure (e.g., 10s or 100s of radios and/or edge compute nodes and with substantial backhaul capacity to clouds). There has been some work on temporary "pop-up" networks for sporting events, concerts, and other short-term purposes [13] [45]. There has also been some work on small networks for rural and disconnected communities [29] [4]. Telecom providers and suppliers have speculated about integration of edge computing resources into network equipment (e.g., base stations) but use cases have typically centered on offloading device computing, network function virtualization (NFV) for various purposes (e.g., QOS, traffic shaping) and, rarely, for more general purpose software [20] [39]. Telecom standards (e.g., ETSI MEC [19], GSMA [24]) envision a world with integrated networks and edge computing but these visions imagine integration at a much higher level and scale than we envision for the JIT Cloudlet. This dichotomy is illustrated in Figure 1. The left side of this figure, loosely based on the ETSI-MEC architecture, depicts a design pattern for an operator-scale deployment of edge computing. While the network and user applications share a common architecture, they are partitioned onto separate infrastructure to assure performance, reliability, and security isolation between operator-critical functions and arbitrary user applications. The JIT Cloudlet architecture is depicted on the right side with both network and user application functions sharing infrastructure up through the Mobile Edge Platform level. Since JIT Cloudlets must be portable and easy to manage, sacrificing some stack isolation for power, space, weight, and integration is a reasonable trade-off.

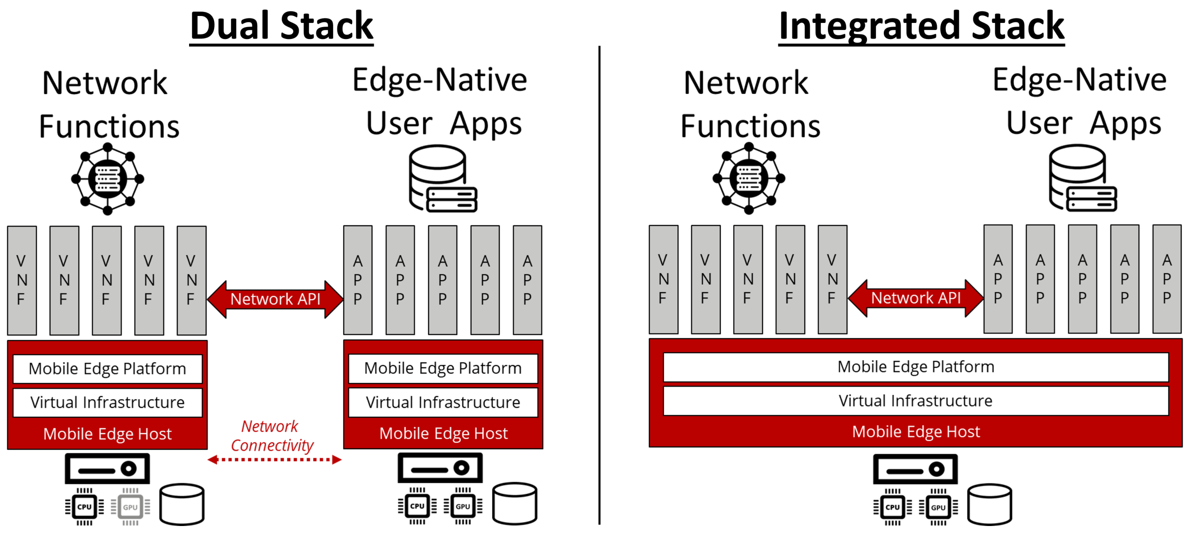


Figure 1: Dual and Integrated Edge Stacks

While cloud computing and, by inheritance, edge computing have evolved with a focus on COTS hardware, open-source software, and a cloud-native paradigm [6], communication networks have historically been implemented with closed, proprietary systems. Software defined networking (SDN) [27] and NFV [36] are undergoing broad adoption in the telecom space. However, cloud-native open-source software for mobile networks is only slowly making progress with initiatives like the Magma [30] and ONF [35] Projects (e.g., OMEC, SD-RAN, SD-Core). These initiatives are at the forefront of the cloudification of mobile networks and are important building blocks for JIT Cloudlet solutions.

1.2 Use Cases

Using drones to inspect bridges and electric grid towers, mounting a search and rescue operation for a child lost in the woods, hosting a backcountry ski event for a weekend, and providing critical communication and application capabilities for forward military operations – These are just a few situations where there is a need to deploy a modern compute and communication infrastructure very rapidly in areas with poor network coverage. These situations require the ability to provide connectivity for mobile phones, connected cameras, drones, and other connected devices. These devices need to offload compute-intensive operations that cannot be performed on-device. They also require access to sophisticated applications, possibly proprietary, with too large a footprint to even run partially on-device. Since these use cases are by definition in remote and possibly rugged locations, transportation of infrastructure from a home base to the actual site demands lightweight and compact infrastructure. Access to the electric grid may be limited so the system may need to run on generator power for hours or days. Given the time critical nature and limited duration of deployments, set up and tear down must be simple and fast for on-site operators to perform. With their potential to be deployed in unsecured physical environments, special consideration needs to be given to security and data privacy characteristics of the solution. Figure 2 shows an illustrative search and rescue JIT Cloudlet solution with all infrastructure other than the antenna and radio deployed in a single desktop-sized server.

In this representative use case, an on-site command center operator, a drone pilot, and two first responders collaborate to rescue an injured climber in the mountains. The edge-native application that supports this must provide computer vision to find the climber and video and audio conferencing between the four actors and the drone feed to coordinate the rescue. Overall system operation must be monitored at the command center.

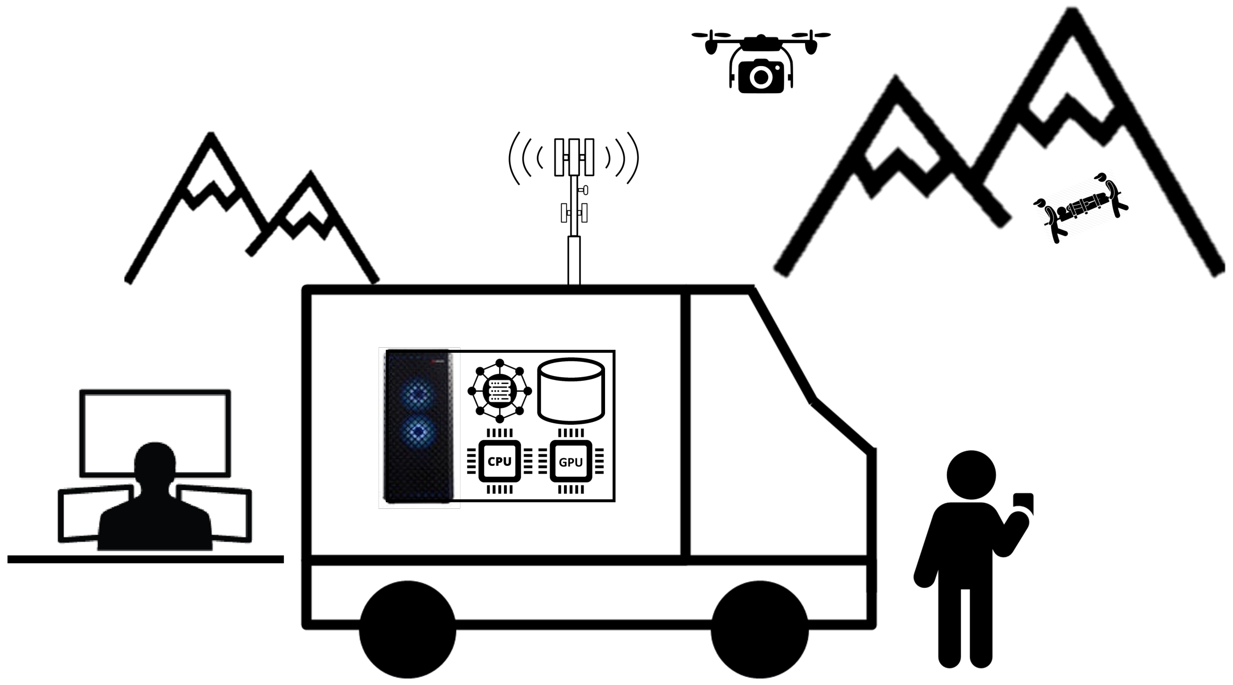


Figure 2: Search and Rescue JIT Cloudlet

1.3 Open Technologies

Solutions to address many of these use cases have been previously deployed (see Section 1.1). However, these solutions have typically been built using closed, proprietary, and bespoke hardware and software and commercial or special purpose mobile networks – making them expensive, inflexible and requiring substantial expertise to develop, deploy, and maintain. Innovation waves driven by cloud, IoT, 5G, and edge computing have stimulated open software, hardware, and network technologies that enable JIT Cloudlet solutions to be built entirely with open components. These open components make creation of solutions easier, faster, and possible for developers with mainstream technology skills. These enablers include:

- *Open-source Software* in the domains of Cloud-Native Computing, Edge-Native Applications, Software Defined Networking, Artificial Intelligence and Computer Vision, Operating Systems, and Development Tools
- *COTS Hardware* including standard high volume servers based on x86 and ARM, accelerated computing hardware based on GPUs, mobile network equipment built to standardized interfaces, and technologies like ethernet, PCIe, and NVMe
- *Private Mobile Network Technologies* like Citizen's Band Radio Spectrum (CBRS) [21] and 5G

1.4 Integration for Footprint

Addressing the space, weight, power, and rapid deployment constraints of a JIT Cloudlet requires a solution with fewer and smaller components. Component reduction is accomplished via the integration of multiple solution functions into fewer components. The open trends discussed in Section 1.3 make this possible by providing common and compatible building blocks across the solution functions. However, there is still effort involved in integration process itself. Integration, in this sense, involves a number of tasks during the continuous integration and deployment (CI/CD) process:

1. Component and version selection and management
2. Individual component assembly, configuration, and test
3. Connection, configuration, and test of the end-to-end solution
4. End-to-end performance benchmarking against user requirements
5. Design and deployment of solution management tools
6. Provisioning of system, network, application, and user data
7. On-going maintenance of the above

2 The Just-In-Time Cloudlet

In the context of the use cases, technologies, and integration goals described in Section 1, we set out to develop a reference architecture and a prototype implementation for a JIT Cloudlet solution. This reference architecture is intended as proof of the validity of the concept given the state of current technologies. It is not intended as a finished, complete product or even as a fully defined development platform for general use. However, all solution building blocks are widely available at a production or near production level.

To ground our reference architecture, we set the design criteria and their *key performance indicators (KPIs)* in Section 2.1, identified the solution building blocks in Section 2.2, built the prototype in Section 2.4 and derived a more general reference architecture from the prototype in Section 2.3. The prototype testing results against the design KPIs are shown in Section 2.5.

2.1 Design Criteria

The design criteria in Table 1 were set to inform the choices made during the development of the prototype. They are based on subjective but common sense requirements to satisfy the use cases described in Section 1. They are not intended as exhaustive but rather as key performance indicators (KPI) for a generalized platform prototype. Fully detailed requirements can only be sensibly established for a production platform intended for a specific use case (e.g., the search and rescue use case in Section 1.2).

Criteria	Nominal	KPI
Weight and Space	*“Fit in the weight and space of a single normal checked bag on a typical US airline”*	Less than 50 pounds; fit in suitcase less than 30 in x 20 in x 12 in (United Airlines requirements)
Power Consumption	*“Run on household power”*	Consume no more than 20A@120V or 2500 running watts
On-site Deployment Time	*“Up and running quickly”*	Less than 3 hours from arrival to unpack, set up, bring up, and test the complete system
Accelerated Computing and Graphics	*“Needs a graphics processing unit (GPU) for AI and graphics rendering”*	Hardware includes a GPU to support computer vision, other AI inferencing tasks, and mixed reality graphics rendering
Backhaul	*“Work with no or narrowband backhaul”*	Fully operational with limited backhaul to offsite systems including the cloud/internet
Software Licensing	*“Everything open-source”*	Software building blocks from widely supported, widely used community-driven open-source projects *(preferred)* or smaller community or non-community open-source repositories *(acceptable)*
Hardware Technologies	*“Everything COTS”*	Hardware building blocks that are 1) commercial products with substantially equivalent systems available from multiple vendors and 2) based on widely used standards (e.g., ethernet, PCIe, NVMe)
Security and Privacy	*“Meets all security and data protection requirements given the JIT Cloudlet deployment model”*	Address supply chain and operational risk in the context of the ecosystem and deployment model of JIT Cloudlet solutions (e.g., disaggregated supply chain, unsecured deployment site)

Table 1: Design Criteria

2.2 The Building Block Technologies

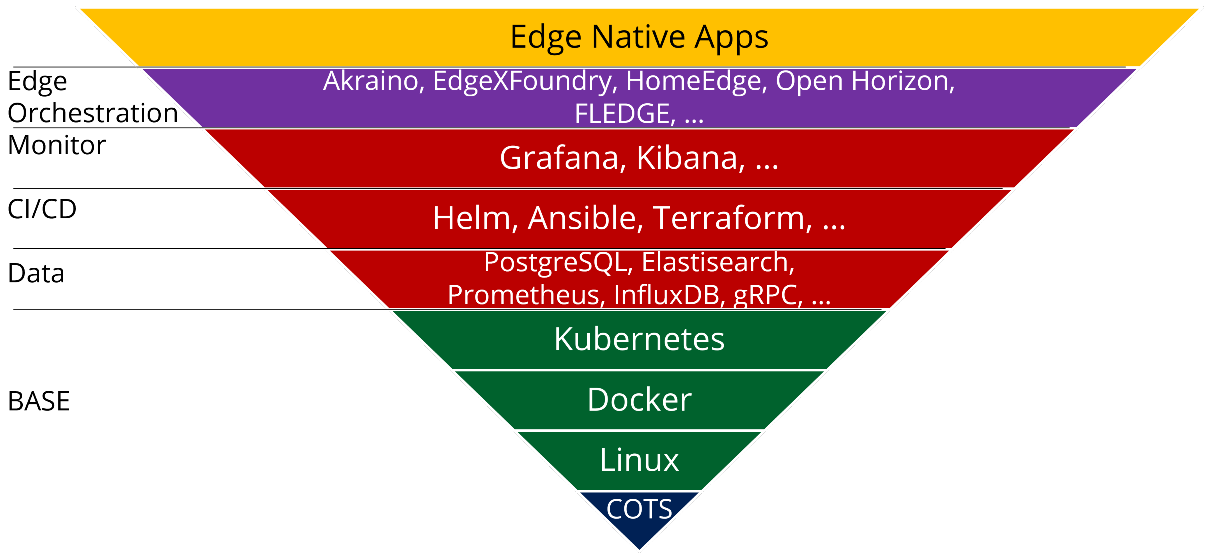
Within the space of the design criteria and open technologies described in Section 1, there are many choices for the building blocks of a JIT Cloudlet solution. In general, our selection of components weighed factors including: 1) fit with the design criteria, 2) maturity and breadth of industry adoption, 3) compatibility with other building blocks, 4) ease of integration into the solution, and 5) our experience and expertise with the building block.

The following sections describe our rationale for some of the major building block choices in the platform. Our base platform started from Ubuntu 20.04 running on a modern server. This platform has been our standard for cloudlets in the Living Edge Lab for several years. On the base platform, we layered software building blocks.

COTS Hardware – The availability of standards-based COTS hardware enables a choice between solution components using criteria beyond baseline functionality. In our case, it allows component selection for power efficiency, space, weight, performance of our software stack, and system ease of use. Most modern servers available from major original equipment manufacturers (OEM) and original design manufacturers (ODM) can be considered COTS – built using open standards based components and interfaces for networking, storage, memory, peripherals, and compute. Because of this, our primary choice was selecting which COTS server to use. Our low-power requirement led us to select an ARM64 server and SSD storage. We chose a desktop form factor to support an accelerated computing PCIe card and because the dimensions were somewhat more compatible with our "suitcase" paradigm. Similarly, the other components in the solution (e.g., switch, router, eNodeB) were available from a variety of commercial suppliers.

Cloud-native Software Platforms – For the software platform and the applications, we heavily favored open-source cloud-native building blocks. These include those from Cloud-Native Computing Foundation (CNCF) [5] (e.g., Kubernetes [9], Helm [8], Prometheus [10], gRPC [7]) and other widely used projects (e.g., Docker [15], Grafana [23], Kibana [17], Elasticsearch [16], Logstash [18]). Cloud-native brings with it a paradigm of containerized microservices with representational state transfer (REST) and streaming application programming interfaces (API). Adoption of this paradigm and the selection of compatible building blocks resulted in overall architectural simplicity – *everything is a service, everything is in a container, and Kubernetes manages everything*.

This "Cloud-native at the Edge" paradigm is depicted in Figure 3. This figure shows an extensive but not exhaustive list of cloud-native open-source software projects that are applicable at the edge. In Section 2.3, we outline the specific components used in the JIT Cloudlet reference architecture.



Edge Native Apps
Edge Orchestration
Akraino, EdgeXFoundry, HomeEdge, Open Horizon, FLEDGE, ...
Monitor
Grafana, Kibana, ...
CI/CD
Helm, Ansible, Terraform, ...
Data
PostgreSQL, Elastisearch, Prometheus, InfluxDB, gRPC, ...
BASE
Kubernetes
Docker
Linux
COTS

Figure 3: Cloud-native At The Edge

Open Software Defined Networking – The goal of integrating the mobile network and the applications onto a single COTS server without any proprietary hardware led us to software defined networking. There were three important building blocks to consider – the Evolved Packet Core (EPC), the Radio Access Network (RAN), and the User Plane Function (UPF) within the EPC. Our open-source requirement led us to the Magma EPC [30] which embeds the Open vSwitch [31] to implement the UPF. See [33] for more on Magma's role in opening the opportunity for private networks. At the time of this work, there was no proven LTE open software RAN building block and we used COTS hardware eNodeBs in the solution.

Open Access Mobile Wireless Spectrum A critical but straightforward choice was the use of unlicensed Citizen's Band Radio Spectrum (CBRS) [21]. In the US, CBRS is the only viable option for an LTE or 5G mobile network to be offered by a non-licensed private network operator. The CBRS General Authorized Access (GAA) tier enables this capability but the GAA requirement for a Spectrum Access System (SAS) requires narrowband backhaul (e.g., satellite).

Unfortunately, there are few equivalent spectrum models outside the US. Canada [37] and Germany [38] have recently announced new spectrum models that enable private mobile networks. Wi-Fi, while unlicensed, has limited outdoor range, poor congestion control, and little interference protection from other transmitters. In some places, experimental spectrum licenses could be used for solution development but those solutions don't have a clear path to production.

Usable Artificial Intelligence and Computer Vision – Many edge-native applications use Artificial Intelligence (AI) and computer vision technologies to implement key parts of the application. These applications typically use an AI framework to abstract the underlying AI models and hardware from the application itself. Our applications often use the PyTorch [32], TensorFlow [22], and OpenCV [11] frameworks with common AI models like YOLO, COCO, and ResNet.

Achieving desired AI application performance usually requires accelerated computing hardware. Accelerated computing hardware is generally not COTS in the strictest sense. While physical form factor and interfaces are generally standarized, software interfaces and programming capabilities are substantially different between suppliers unless abstracted by a higher level library, (e.g., PyTorch). We chose the NVIDIA GPU PCIe card family because, as the most commonly used, they have the broadest support and therefore are the easiest to integrate.

Edge-Native Applications – Containerized microservices has been our edge-native application development paradigm for many years. That practice allow us to use several of our existing applications with no modifications.

2.3 Reference Architecture

A generalized JIT Cloudlet reference architecture, based on the prototype described in Section 2.4, is shown in Figure 4. This reference architecture shows the major building blocks of a JIT Cloudlet solution for a 4G LTE Network. This reference architecture and the first generation prototype focuses on a 4G LTE solution because of its proven *solution readiness*. Namely, the major components of the solution – UEs, eNodeBs, EPC had already been deployed successfully in our campus private LTE network. However, we are in process of developing a prototype and reference architecture for a 5G JIT Cloudlet solution. See Section 2.6 for more on our goals for 5G.

- The **Cloudlet** is the primary workhorse for the JIT Cloudlet solution. It is responsible for the time critical elements (e.g., UPF) of the 4G LTE Evolved Packet Core (EPC) as implemented by the Magma Access Gateway. It also hosts the use case specific applications required for a particular solution. Each major component is deployed in its own dedicated namespace in a single Kubernetes cluster running on Ubuntu. The Open vSwitch provides accelerated

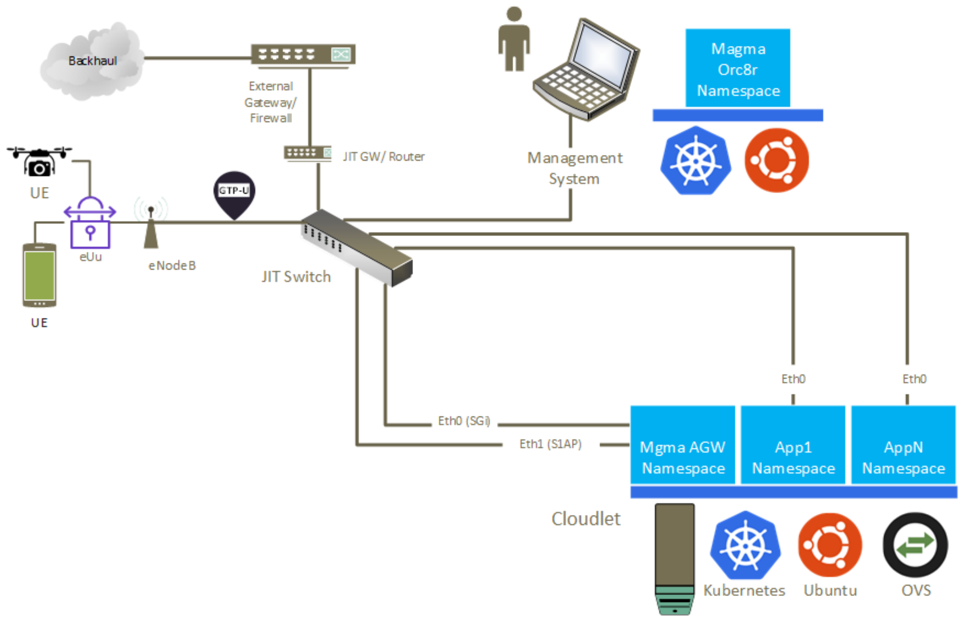


Figure 4: JIT Cloudlet Reference Architecture

forwarding for user plane traffic from UEs. All system deployment on the cloudlet is managed using common cloud-native tools such as Helm, Docker, and Ansible [40]. The cloudlet also requires an accelerated computing platform like a GPU to support any graphics or AI application workloads. Docker and Kubernetes are configured to support use of accelerated computing.

- *Kubernetes Cluster* – Kubernetes [9] is an open-source container orchestration system for automating software deployment, scaling, and management. Originally designed by Google, the project is now maintained by the Cloud-Native Computing Foundation. For the JIT Cloudlet, Kubernetes provides a proven, well-supported, common, open-source platform for the deployment and management of network functions and user applications in an integrated single node system. Some use cases may require coordination across multiple JIT Cloudlets. In those cases, a cross-tier orchestration system like our Sinfonia project [42] working with Kubernetes may be needed. These use cases are out-of-scope of this reference architecture but are discussed in Section 3.
- *Magma Access Gateway (AGW)* – Magma [30] is an open-source software platform that gives network operators an open, flexible and extendable mobile core network solution supporting 4G, 5G, WiFi, and other access networks. In an LTE network, the AGW implements an evolved packet core (EPC), and a combination of a Packet Gateway (PGW) and authentication, authorization, and accounting (AAA) services. It works with existing, unmodified commercial radio hardware. Magma is maintained by the Magma Project in the Linux Foundation. For the JIT Cloudlet, Magma's architecture and open-source implementation allows flexibility and low-cost in a small-scale deployment.

  - *Open vSwitch* – OVS [31] is an open-source implementation of a distributed virtual multi-layer switch. OVS is maintained by the Open vSwitch Project in the Linux Foundation. In the JIT Cloudlet, OVS is important to assure high performance and throughput for user plane traffic traversing the AGW.
  - *Application(s)* – In the reference architecture, specific applications are not called out – in practice, they are determined by the use cases that the JIT Cloudlet solution implements. For example, a bridge inspection solution would implement a computer vision application to detect rust and other degradation on metal bridges. In general, though, cloudlet applications will be containerized, service oriented, and Helm-deployable. The prototype in Section 2.4 calls out specific applications tested.

- The **Management System** hosts the operator consoles for all of the primary components in the solution. When backhaul is available, remote access to these consoles is also possible but, in disconnected mode, a local console with keyboard, video, and mouse (KVM) is required. In addition, the management system runs the Magma Orchestrator (Orc8r) component in its own Kubernetes cluster.

  - The *Kubernetes Cluster* exists predominately to host the Magma Orchestrator. However, this cluster can also support deployment of other Docker-based management components as needed (e.g., a custom Grafana dashboard).
  - The *Magma Orchestrator (Orc8r)* provides a simple and consistent way to configure and monitor the Magma wireless network securely. Its primary management console is the *Network Management System (NMS)*. Running the Orc8r on the management console system was necessary in our prototype for reasons discussed in Section 2.4. However, there is general value in offloading the Orc8r from the Cloudlet to assure adequate resources for the time sensitive AGW and applications on the Cloudlet. There is no functional reason why the Orc8r could not be deployed on the Cloudlet.
  - *Other management consoles, dashboards, and tools*. The management system enables user access to the components of the JIT Cloudlet solution via http, VNC, and command line. The primary managed components are the Cloudlet (http/VNC/command line), eNodeB (http), router (http). Cross component monitoring (e.g., using Wireshark, tcpdump, iPerf) is also available via the management system.

- The **eNodeB(s)** are the Radio Access Network (RAN) for the JIT Cloudlet solution. In 4G LTE, available CBRS eNodeBs are primarily COTS hardware solutions. The eNodeB includes the radio and antenna for the solution and the interface to bring user and control plane traffic to the EPC. In general, the JIT Cloudlet solution allows multiple eNodeBs, different types of eNodeBs, and both indoor and outdoor eNodeBs. While the JIT Cloudlet solution is designed to operate with limited backhaul, compliance with CBRS GAA requirements mandates a remote SAS service to reduce power on interfering eNodeB radios. See the discussion on backhaul below.

- **User Equipment (UE)** are the myriad of devices required to deliver the JIT Cloudlet application(s). They can include user phones and computers, drones, cameras, sensors, or other IoT devices. All UE must be CBRS-capable and able to connect to 4G LTE networks (or connect through a dongle or hotspot that enables this connectivity). Other UE functionality depends on the needs of the specific use case requirements.

- The **Switch and Router** provide basic ethernet network connectivity and security between the individual solution components and, when backhaul is available, between the solution and the outside world. The size of the switch is driven primarily by the number of eNodeBs connected to the network.
- **Backhaul (optional)** allows the solution to communicate with the outside world to enable operations, data transfer, and communications. Many JIT cloudlet use cases assume that backhaul may not be available so the applications are designed to be self-contained. However, when backhaul is available, it can be useful (See CBRS SAS discussion above). Given the three hour on-site deployment KPI, backhaul requires careful pre-deployment planning. See Table 2 for more information on backhaul considerations.

Capacity		
Backhaul Type	**Use Cases**	**Considerations**
Disconnected	Completely offline operation • Remote unserved area • Backhaul not pre-provisioned	Enables operation at any arbitrary site; Requires self-contained edge-native applications apps with no external dependencies
Narrowband (less than 1Mbps)	Keep alive • CBRS SAS Service • Low bandwidth communication • Remote system management	Enables critical lifeline services for systems and operators; Satellite provides a ubiquitous, pre-provisionable option
Broadband	Cloud service access • Applications with remote cloud service dependencies	Enables edge-native applications in areas with no existing edge computing infrastructure has been deployed and/or where existing access networks are poor

Provisioning		
Backhaul Type	**Use Cases**	**Considerations**
Pre-provisioned	Pre-planned deployment • Area with existing infrastructure • Advance service order possible	Likely when new telco provisioned backhaul is required
Just-in-Time provisioned	Existing infrastructure but no pre-provisioned service • Operator available • Physical connection point available • Network capacity available	Possible in existing private infrastructure when the provisioning process can support

Table 2: Backhaul Considerations

Figure 5: Prototype

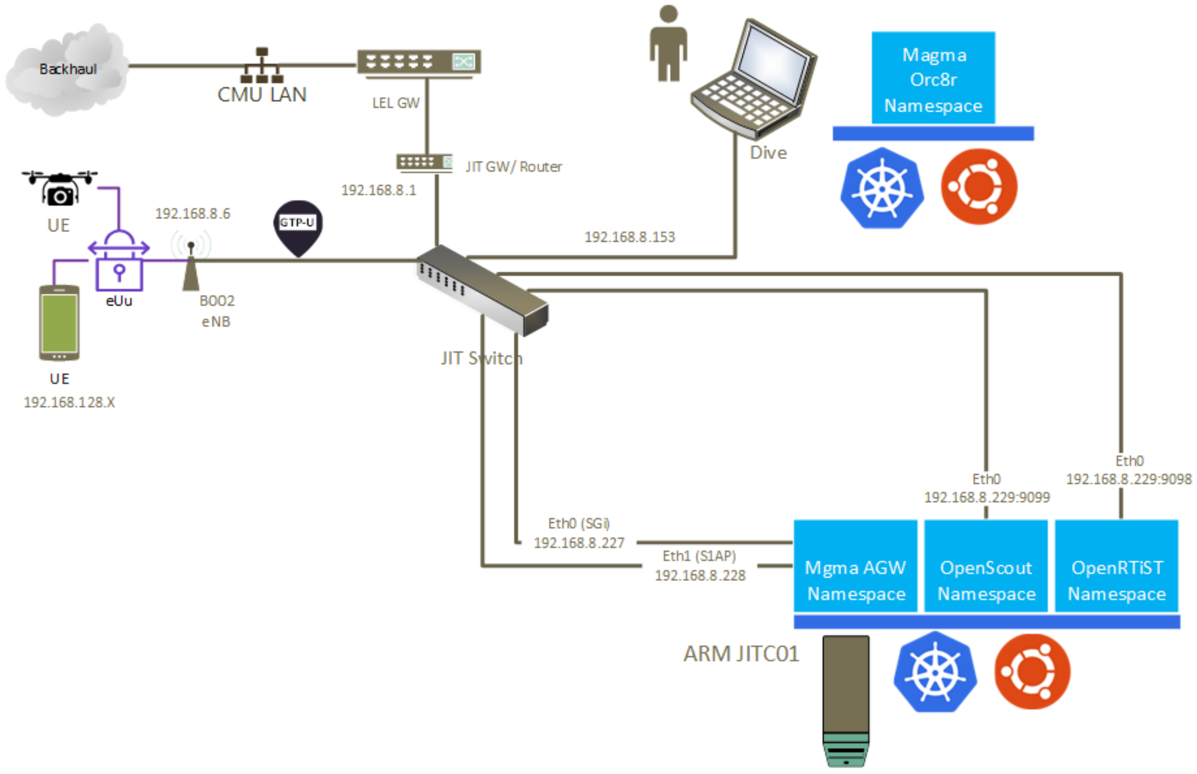


Figure 6: Prototype Architecture

2.4 The Prototype

The Just-in-Time Cloudlet solution prototype is a physical, working instantiation of the reference architecture described in Section 2.3 and a representative example that the design criteria in Section 2.1 can mostly be met by today's building block technologies in Section 2.2. The detailed solution component configurations can be found in the appendix. The physical footprint of the prototype can be seen in Figure 5 and the design in Figure 6.

Very little new software was developed to create the prototype. Instead, the primary effort was iteration through the CI/CD steps in Section 1.4. The building block enablers made this possible but, because of the strict footprint requirements, a standardized platform architecture was mandated. More details on this platform architecture are provided in Section 2.3

In the prototype, the cloudlet was built on the ADLINK AVA ARM64 Developer Platform with an added NVIDIA GeForce GTX 1080ti GPU Card. We chose an ARM platform to better meet the power constraints imposed by the JIT Cloudlet design criteria (See Section 2.1). Ubuntu, Docker, and Kubernetes were deployed on the platform and a Magma AGW and two Living Edge Lab developed edge-native applications were deployed in the Kubernetes cluster.

The management system was built on a Dell Latitude 5420 Rugged Laptop. Ubuntu, Docker, and Kubernetes were deployed on the platform and a Magma Orchestrator was deployed in the Kubernetes cluster.

Using the gateway/router and the switch, the cloudlet, management system, and eNodeB were physically connected and provisioned with IP addresses. The UE were provisioned with network subscriber identity module (SIM) cards. All information for the network was provisioned into the Orc8r and the AGW, eNodeB, Orc8r, and UE were connected.

With the network now serving UEs with connectivity, the UE were able to connect to the two edge-native applications. OpenScout [2] is a computer vision/object classification application analogous to what might be used in drone-based search and rescue. OpenRTiST [1] is an augmented reality application analogous to what might be used to deliver multimedia entertainment at a backcountry ski event.

Both applications ran unmodified in Kubernetes (after creating appropriate Helm charts for deployment). The applications, which both use image and AI technologies, shared the GPU. These applications embody the JIT Cloudlet design principles, using Docker, http-based services, and open AI technologies like OpenCV, Pytorch, YOLO, and TensorFlow to implement their functionality.

The performance results for the prototype are presented in Section 2.5.

2.5 Results and Performance

The prototype described in Section 2.4 was evaluated for two primary attributes:

1. Does the prototype meet the design criteria defined in Section 2.1?
2. Does the prototype perform as well or better than a non-integrated LTE network and Cloudlet solution?

The design KPIs results are shown in Table 3.

Criteria	**Nominal**	**Result**	**Met/ Not Met?**
Weight and Space	*“Fit in the weight and space of a single normal checked bag on a typical US airline”*	Footprint: • Weight: 80lb • Space (Cloudlet): 18.5in x 16.5in x 7.5in	Partially Met
Power Consumption	*“Run on household power”*	250W JIT Solution w/GPU Active	Met
On-site Deployment Time	*“Up and running quickly”*	Less than three hours	Met
Accelerated Computing and Graphics	*“Needs a GPU for AI and graphics rendering”*	Shared GPU for applications	Met
Backhaul	*“Work with no or narrow-band backhaul”*	Fully disconnected except low-bandwidth SAS connection	Met (narrowband)
Software Licensing	*“Everything open-source”*	Cloudlet and Management System all open-source; eNodeB, Switch, Router closed-source	Met within state-of-art
Hardware Technologies	*“Everything COTS”*	Cloudlet, Management System, Switch, Router, eNodeB, COTS; GPUs not COTS	Met within state-of-art
Security and Privacy	*“Meets all security and data protection requirements given the JIT Cloudlet deployment model”*	Deferred	Unknown

Table 3: Protoype Design KPI Results

As shown in Table 4, the complete prototype including a hardened case to contain it weighs approximately 83 lbs (38 kg). This weight exceeds the typical US airline checked baggage standard (50 lb). However, it is within the oversized baggage limit (100 lb) and, therefore, can be checked with an additional fee. The largest component in the JITC solution, the Cloudlet, measures 18.5in x 16.5in x 7.5in. This component plus all other solution components has been shown to fit in a 33.5in x 21.9in x 13.0in hardened case [34] with sufficient padding to protect the equipment.

Solution Component	Weight (lb)
Router	0.4
eNodeB	3.5
Switch	1.4
Management System	7.5
Cloudlet (Without GPU)	29.0
GPU	1.4
Cables/Power Adapters	4.0 (est)
Hardened Case	36.0
Total	83.2

Table 4: Solution Weight

As shown in Table 3, the JIT Cloudlet solution meets the power consumption KPI with a total power consumption (excluding the UE) of 250W or less with the GPU actively running the openscout application for object detection. The power consumption by solution component is shown in Table 5. Power varies with the Cloudlet GPU and CPU load but does not exceed 250W. With a definition of household power as requiring only a 20A/120V power line *(2500 running watts)*, the solution easily meets the power requirements. A typical 2500W portable generator will run for over 10 hours on one gallon of gas with this load.

Solution Component	Power Consumption (w)
Router	3
eNodeB	5.5
Switch	9.9
Management System	5.5
Cloudlet (GPU Inactive)	133
Active GPU	76
Total	233

Table 5: Power Consumption

The solution meets the accelerated computing KPI by integrating a GPU PCIe card to support OpenScout and OpenRTiST simultaneous use. However, the prototype was not tested under a scaled application load and therefore might require additional accelerated compute capacity in a commercial solution. Adding a larger GPU or additional GPU cards to accomplish this would, of course, increase solution power consumption and weight.

As discussed in Section 2.3, the prototype meets the backhaul KPI with narrowband backhaul required for SAS connection.

While there are exceptions to the open-source KPI within the "closed" parts of the prototype (e.g., eNodeB, Switch, Router and the BIOS, drivers, etc., in the Cloudlet and Management System), the spirit of the KPI is met with fully open-source platform software from the operating system up on the Cloudlet and Management System software stack.

The COTS KPI is met in spirit with the Cloudet and the Management System based on COTS computing technologies. The Switch and Router are COTS as they provide standard ethernet switching and routing functions with similar systems available from a wide variety of suppliers. The eNodeB is COTS, however, there are only a few available substitute CBRS-compatible eNodeBs from a limited number of suppliers.

After an initial assessment of the security and privacy requirements and potential vulnerabilities in a JIT Cloudlet environment, we deferred a detailed definition and assessment of these requirements to a later stage of the prototype effort. For more information on our plans, see Section 3.

In addition to the KPIs, we assessed the overall solution performance relative to a similar non-integrated solution. The comparative non-integrated solution was our existing LEL environment where a private mobile network (EPC and RAN) runs on separate physical systems from the application-serving cloudlet. The LEL environment provides better performance than that achieved when using a commercial mobile network and, so, represents a higher bar for the JIT Cloudlet solution to meet.

As shown in Table 6, ping round trip times (RTT) were measured at 21ms on the JIT Cloudlet solution. This value is at the low (good) end of what is typical in LTE networks [44] and is 10ms lower than typically measured in the LEL private LTE network [43]. As expected from previous measurements, most of this improvement comes from reduction in the uplink latency between the UE and the RAN. The JIT Cloudlet solution has a median uplink and downlink latency of 13ms and 9ms respectively as shown in Figure 7. Comparatively, the LEL private LTE network achieves typical uplink and downlink median latencies of 25ms and 9ms. See [43] for more detail about previous LEL private LTE network measurements.

Application	**Measurement**	**Result**	**Met/ Not Met?**
Ping	Round Trip Time – median (stdev) in ms	21 (4.2)	Met
Iperf	Download Throughput – median (stdev) in Mbps	51.7 (10.5)	Met
	Upload Throughput – median (stdev) in Mbps	13.4 (0.2)	Met
OpenRTiST	Round Trip Time median (stdev) in ms	150 (20)	Met
	Frames Per Second median (stdev) in fps	11.5 (1.5)	Met

Table 6: Performance Results

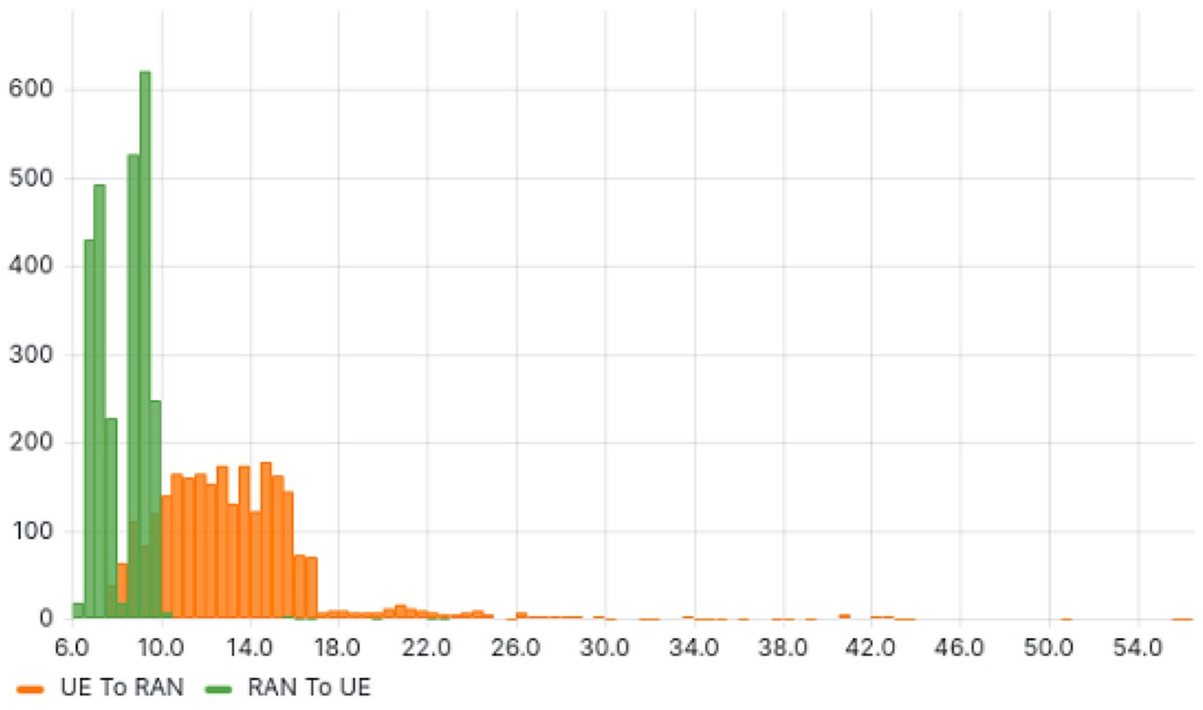


Figure 7: Latency between UE and RAN

Network throughput, measured by upload and download speeds, is a common measurement for mobile networks. In these networks, the limiting factor on throughput is usually the realized wireless channel bandwidth between the server and client. Download throughput was measured using an iperf3 [26] server on the management server and an iperf3 client on the UE. The UE periodically launched an iperf session and recorded the resulting bit rate. Upload throughput was measured with the iperf reverse option enabled. The measured download and upload throughput was 52 Mbps and 13 Mbps respectively. Both download and upload throughput are comparable to LEL network measurements.

OpenRTiST performance is typically measured by its framerate (FPS) and round-trip-time (RTT) for an image to be sent from the UE to the Cloudlet and back. Both measures are impacted by network transmission times and image processing at the Cloudlet. Because OpenRTiST is a more realistic application than ping or iperf (i.e., it transmits larger packets of data and executes a compute-intensive workload at the cloudlet), its performance is a useful supplement to those benchmarks. The OpenRTiST FPS and RTT measurements were 12 fps and 150 ms. These are typical measurements for OpenRTiST over LTE networks. It should be noted that the selected artist style has a significant impact on the per frame Cloudlet compute time and accordingly on the FPS and RTT measurements.

JIT Cloudlet use cases will typically support small numbers of user applications and active users. For this reason, we did not subject the prototype to substantial load and interaction testing. However, simple testing with two active users and combinations of OpenRTiST and OpenScout usage yielded some interesting anecdotal results. Both OpenRTiST and OpenScout were configure to use the single GPU within the prototype. With two simultaneous users, GPU utilization never exceeded 35% (two OpenScout users). There was no perceptual impact on

OpenScout performance. With two OpenRTiST users, GPU utilization never exceeded 20% however framerates dropped from $\sim$14 fps for a single user to $\sim$11 fps each for two users – suggesting a bottleneck somewhere other than in the GPU. With a single user for each application, GPU utilization remained below 35%, OpenRTiST framerate dropped to $\sim$11 fps and there was no perceptual impact to OpenScout performance. These results suggest load testing for production deployment use cases should be a priority.

2.6 Learnings

The JIT Cloudlet Prototype is a proof-of-concept for the idea of a portable, open, integrated network and application solution for the use cases that this report focuses on. It demonstrates that such a solution can be built with open, off-the-shelf hardware and software. In the process of building and using the prototype, we discovered:

- The open technologies in the prototype are relatively mature and complete for JIT Cloudlet uses. We did not do significant load and scaling testing and it is likely that performance and reliability issues will arise when such testing is done.
- Integration of the component technologies was relatively straightforward given their common design approaches. The choice of Kubernetes as a common framework created a focus for our integration requirements.
- As is often the case, IP networking was the most complex part of the integration process. In particular, both Magma and Kubernetes introduce a number of unique networking practices that required workarounds. Diagnosing and debugging networking issues required substantial time with container logs, Wireshark, and tcpdump.
- Our Cloudlet prototype is an ARM64 server. Given the large open-source content of the prototype, we initially feared that we would find many components without ARM64 support. In the end, as discussed above, only the Magma Orchestrator was not available on ARM64. We did however have to adapt some Helm charts and Docker makefiles to deploy ARM64 containers.
- Creation of the prototype required a highly motivated developer and a "do-it-yourself" (DIY) approach to integration. Broader adoption of our approach would require creating a "developer-ready" platform that pre-packages large portions of the stack and provides support to the developer users.

2.7 Work In Progress

As of the writing of this tech report, several enhancements of the JIT Cloudlet solution are underway.

- *5G Network* – Our 4G LTE prototype implementation was driven by factors discussed in Section 2.3 however we are currently testing 5G for deployment in both the LEL private network and the JIT Cloudlet prototype. We expect the 5G New Radio to give us significant reduction in end-to-end latency for edge-native applications in both environments.

- *Realtime Collaboration* – Many JIT Cloudlet use cases depend on realtime collaboration between the users on-site. Many common realtime collaboration tools (e.g., Zoom, MS Teams) require broadband backhaul to the cloud. We are implementing a WebRTC [12] JIT Cloudlet service to support this on-site collaboration requirement.
- *Integrated Edge-Native Application* – The edge-native applications used in the prototype demonstrate the JIT Cloudlet design goals but they do a poor job of illustrating the use cases. We are currently developing two demonstration edge-native applications showcasing the search and rescue and the remote concert use cases.
- *Deployment Recipe* – To mitigate the amount of DIY work necessary to bring up a new JIT Cloud instantiation, we are working on an Ansible and Helm based deployment recipe.
- *Form Factor* – As discussed above, our prototype did not meet the weight and space KPI. We are beginning to look at other system form factors that may bring us closer to these goals.

3 Future Work and Opportunities

This section discusses future opportunities in three areas: extensions to the current prototype platform, technologies that have potential value to future JIT Cloudlets, and business and ecosystem issues relevant to adoption of JIT Cloudets.

3.1 Prototype Extensions

We did not validate the prototype in a number of important areas for JIT Cloudlet solutions. These are opportunities for future work by us or others.

- *Realistic RAN deployment* – our prototype includes only a single indoor eNodeB. The defined use cases are, for the most part, outdoors and distributed. There are sure to be complexities in deploying multiple geographically dispersed eNodeBs in a typical deployment environment. Three significant complexities will be:
  - Deploying fronthaul from a distributed eNodeB to the Cloudlet. A fixed wireless approach is likely to be the only practical method for any eNodeB distant from Cloudlet but engineering both wireless access and backhaul on the fly will be challenging.
  - Physical deployment of the radios at an elevation sufficient to achieve sufficient area coverage. Radios mounted on telescoping poles are the likely solution however testing in a realistic environment is necessary to validate practicality.
  - If radios are temporary and portable, then maintaining the correct radio location information with the CBRS SAS service provider may be complex. Spectrum licensing approaches beyond CBRS were not examined.
- *Operations and management* – the prototype is managed using the discrete management capabilities of each component (e.g., a separate management console for Magma, eNodeB, router, latency monitoring, etc.). As a prototype, this approach was sufficient. However, a

production solution would call for a more integrated and planned approach to operations and management.

- *Real applications* – our test applications are "toy" applications that exercise key prototype capabilities. A full use case specific solution would require a more complete and production-ready set of applications tested for load, performance, reliability, and security.

- *ORAN* – The lack of a 5G JIT Cloudlet meant that we could not explore using ORAN in our prototype. Conceptually at least, running the Centralized Unit (CU) and possibly the Distributed Unit (DU) in the Cloudlet's Kubernetes cluster would further the architectural consistency and integration of the solution. Understanding the performance trade-offs of this approach would also be informative. For example, what is the impact of running the DU on the same system as, say, a graphics intensive augmented reality application?

3.2 Technology Evolution

- *Network slicing* – given the small scale of JIT Cloudlet deployments and the diversity of workloads in a given deployment, we hypothesize that there may be value for 5G network slicing to provide differential quality of service for different workloads. For example, we could envision a network slice for voice communications, another for real-time video analytics, and another for drone flight control. On the other hand, the small scale and low capacity requirements for applications may make specialized QoS functionality unncessary. Validating these ideas requires a 5G network enabled with network slicing. We expect to follow up in this area when we have upgraded our prototype to 5G.

- *SD-RAN* – Software Defined Radio Access Networks can extend the JIT Cloudlet architecture even further into the network. Coupled with ORAN, the RAN can become yet another building block of an open solution. As SD-RAN projects mature, integration into a JIT Cloudlet will become a natural next step.

- *Device offload provisioning* – When a JIT Cloudlet provides compute offload resources to connected devices, the configuration of those resources can be done statically during initial service provisioning or dynamically as the connected device requires them. We explored this type of dynamic provisioning in 2013 [25]. We believe this approach could be valuable in JIT Cloudlets where it is not possible to provision from a remote cloud.

- *Cross-tier orchestration* – In multi-JIT Cloudlet use cases, the orchestration of application resources across devices, users, cloudlets, and cloud-based services becomes more complicated and dynamic. Integration of a cross-tier orchestration system like Sinfonia [42] will likely be needed.

3.3 Business and Ecosystem

- *Security and privacy* – the unique nature of the JIT Cloudlet supply chain and deployment environments warrants a better understanding of the possible vulnerabilities and threats in the JIT Cloudlet solution. In particular, establishing a chain of trust through a disaggregated supply chain, while not unique to the JIT Cloudlet, it is a salient concern especially given the high open-source content. Also, since the JIT Cloudlet will often be deployed in locations without secure physical access, direct physical attacks are a greater threat than in, say, a data center deployment. Time and expertise has so far limited our ability to explore these areas.

- *Mobile operator managed JIT Cloudlets* – Mobile network operators could offer JIT Cloudlets-as-a-Service to customers for intermediate or long term deployment in areas where edge computing has not already been deployed. These services can enable operators to deploy edge computing gradually without the need for large infrastructure capital projects. It can also provide an agile approach to responding to evolving customer needs for edge-computing. Their scale, expertise, customer support infrastructure, and access to spectrum beyond the CBRS GAA range may give mobile operators greater reach than that of a private network operator.

4 Conclusion

The Just-in-Time Cloudlet is a new approach to addressing longstanding needs to rapidly and temporarily deploy applications in places without mobile access networks and backhaul to the internet. This approach is enabled by academic and industry innovations that can be broadly labeled "Cloud Native at the Edge". Our open JIT Cloudlet reference architecture and prototype show the feasibility and possibilities of the approach. But, much work remains for application and solution developers to create production solutions for JIT Cloudlet use cases.

Our work also demonstrates the critical importance of two specific recent innovations – the CBRS GAA licensing model and the Magma open-source mobile wireless core. Without the former, the private network approach that the JIT Cloudlet requires would not be possible. Without the latter, the integration of the mobile network and applications would not be possible using only open-source components.

.

5 Appendix A. Prototype Hardware and Software Configurations

Cloudlet Hardware

Adlink AVA Developer Platform

- Ampere® Altra® 32-core SoC (Arm Neoverse N1 architecture)
- 4x 10GbE and 1x GbE LAN ports
- 32GB DDR4 memory, 2x128GB NVMe M.2 storage
- 64 PCIe Gen4 lanes (3 x16, 2 x4, and 2 M.2 slots)
- Open source firmware (EDKII bootloads with TianoCore/UEFI)
- Arm SystemReady SR V2.1 certified, SOAFEE-enabled

Accelerated Compute

- NVIDIA GeForce GTX 1080ti

Cloudlet Software

- Operating System: Ubuntu 20.04
- Docker v20.10.21
- Kubernetes v1.19.1
- Helm v3.3.4
- Magma v1.9.0
- OpenScout
  - github https://github.com/cmusatyalab/openscout arm64 branch (commit: e89a898)
  - dockerhub cmusatyalab/openscout:arm64 (digest: 811f408c815a)
- OpenRTiST
  - github https://github.com/cmusatyalab/openrtist, arm64 branch (commit: b87a3bc)
  - dockerhub cmusatyalab/openrtist:arm64 (digest: 09d2904e64d1)

Table 7: Cloudlet Platform Specifications

Management System Hardware
Dell Latitude 5420 Rugged Laptop • Intel® Core™ i7-8650U CPU @ 1.90GHz • 16GB DRAM • 512GB NVMe SSD
Management System Software
• Operating System: Ubuntu 20.04 • Docker v20.10.21 • Kubernetes v1.24.10 • Helm v3.11.2 • Magma v1.9.0

Table 8: Management System Specifications

Other Hardware
• **eNodeB** Baicells Neutrino 430 • **Router** GLiNet GL-MV1000 • **Switch** Belkin 5-port ethernet switch • **User Equipment**   – Google Pixel4   – Multitech MultiConnect® microCell Cellular Modem (MTCM2) USB Dongle

Table 9: Other Hardware

References

[1] CARNEGIE MELLON UNIVERSITY. Openrtist: Real-time style transfer. https://github.com/cmusatyalab/openrtist.

[2] CARNEGIE MELLON UNIVERSITY LIVING EDGE LAB. Openscout: Distributed automated situational awareness. https://github.com/cmusatyalab/openscout.

[3] CARNEGIE MELLON UNIVERSITY LIVING EDGE LAB. Edge Computing @ CMU Living Edge Lab, 2023. https://www.cmu.edu/scs/edgecomputing/.

[4] CHIARAVIGLIO, L., BLEFARI-MELAZZI, N., LIU, W., GUTIERREZ, J. A., VAN DE BEEK, J., BIRKE, R., CHEN, L., IDZIKOWSKI, F., KILPER, D., MONTI, P., BAGULA, A., AND WU, J. Bringing 5G into Rural and Low-Income Areas: Is It Feasible? *IEEE Communications Standards Magazine 1*, 3 (2017), 50–57.

[5] CNCF. Cloud Native Computing Foundation. https://www.cncf.io.

[6] CNCF. CNCF Cloud Native Definition v1.0. https://github.com/cncf/toc/blob/main/DEFINITION.md.

[7] CNCF. gRPC. https://grpc.io/.

[8] CNCF. Helm. https://helm.sh/.

[9] CNCF. Kubernetes. https://kubernetes.io/.

[10] CNCF. Prometheus. https://prometheus.io/".

[11] COMMUNITY PROJECT. OpenCV. https://opencv.org/.

[12] COMMUNITY PROJECT. WebRTC. https://webrtc.org/.

[13] CRADLEPOINT. Pop-Up Networks. https://cradlepoint.com/solutions/use-cases/pop-up-networks/.

[14] CUSTOM MARKET INSIGHTS. Global Private 5G Network Market 2023–2032. https://www.custommarketinsights.com/report/private-5g-network-market/.

[15] DOCKER, INC. Docker. https://www.docker.com/.

[16] ELASTIC. Elasticsearch. https://www.elastic.co/elasticsearch/.

[17] ELASTIC. Kibana. https://www.elastic.co/kibana.

[18] ELASTIC. Logstash. https://www.elastic.co/logstash/.

[19] ETSI. Multi-access Edge Computing (MEC). https://www.etsi.org/technologies/multi-access-edge-computing.

[20] FAN, W., LIU, Y., TANG, B., WU, F., AND WANG, Z. Computation offloading based on cooperations of mobile edge computing-enabled base stations. *IEEE Access 6* (2018), 22622–22633.

[21] FIERCE WIRELESS. What is CBRS?, 2020. https://www.fiercewireless.com/private-wireless/what-cbrs.

[22] GOOGLE. TensorFlow. https://www.tensorflow.org/.

[23] GRAFANA LABS. Grafana. https://grafana.com.

[24] GSMA. Operator Platform Telco Edge Requirements. https://www.gsma.com/futurenetworks/resources/operator-platform-telco-edge-requirements/.

[25] HA, K., PILLAI, P., RICHTER, W., ABE, Y., AND SATYANARAYANAN, M. Just-in-time provisioning for cyber foraging. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services* (New York, NY, USA, 2013), MobiSys '13, Association for Computing Machinery, p. 153–166.

[26] IPERF.FR. iPerf - The ultimate speed test tool for TCP, UDP and SCTP. https://iperf.fr/.

[27] JA'AFREH, M. A., ADHAMI, H., ALCHALABI, A. E., HODA, M., AND EL SADDIK, A. Toward integrating software defined networks with the internet of things: a review. *Cluster Computing* (2022), 1–18.

[28] JUST TOTAL TECH. Future Of Edge Computing: Top 6 Trends 2023. https://justtotaltech.com/edge-computing-trends-in-2023/.

[29] KUMAR, S. K. A., STEWART, R., CRAWFORD, D., AND CHAUDHARI, S. Business model for rural connectivity using multi-tenancy 5G network slicing. In *2020 IEEE 17th International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET)* (2020), pp. 182–188.

[30] LINUX FOUNDATION. Magma. https://www.magmacore.org.

[31] LINUX FOUNDATION. Open vSwitch. https://www.openvswitch.org/.

[32] LINUX FOUNDATION. PyTorch. https://pytorch.org/.

[33] MIRANTIS. Mirantis to Democratize Connectivity with Magma, a Converged Access Gateway Developed by Facebook. https://www.mirantis.com/company/press-center/company-news/mirantis-to-democratize-connectivity-with-magma-a-converged-access-gateway-dev

[34] MONOPRICE. Hardened Case Model 12277. https://www.monoprice.com/product?p_id=12277.

[35] OPEN NETWORKING FOUNDATION. https://opennetworking.org/.

[36] QURESHI, K. N., AHMAD, E., ANWAR, M., GHAFOOR, K. Z., AND JEON, G. Network functions virtualization for mobile core and heterogeneous cellular networks. *Wireless Personal Communications* (2022), 1–17.

[37] RCR WIRELESS NEWS. Canada announces new licensing framework for local 5G access. https://www.rcrwireless.com/20230504/5g/canada-announces-new-licensing-framework-local-5g-access.

[38] RCR WIRELESS NEWS. German regulator has already approved 123 private 5G networks. https://www.rcrwireless.com/20210607/5g/german-regulator-already-awarded-123-private-5g-networks.

[39] RCR WIRELESS NEWS. NodeEngine, a unique 5G base station-embedded edge computing solution. `https://www.rcrwireless.com/20220810/5g/nodeengine-a-unique-5g-base-station-embedded-edge-computing-solution`.

[40] RED HAT, INC. AND COMMUNITY PROJECT. Ansible. `https://www.ansible.com/overview/how-ansible-works/`.

[41] SATYANARAYANAN, M., BAHL, P., CACERES, R., AND DAVIES, N. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing 8*, 4 (2009), 14–23.

[42] SATYANARAYANAN, M., HARKES, J., BLAKLEY, J., MEUNIER, M., MOHANDOSS, G., FRIEDT, K., THULASI, A., SAXENA, P., AND BARRITT, B. Sinfonia: Cross-tier orchestration for edge-native applications. *Frontiers in the Internet of Things 1* (2022), 1025247.

[43] SMITH, S., DARWHEKAR, I., BLAKLEY, J., EISZLER, T., AND HARKES, J. Segmenting Latency in a Private 4G LTE Network. Tech. Rep. Technical Report CMU-CS-22-115, School of Computer Science Carnegie Mellon University, 2022.

[44] TECHTARGET. 5G vs. 4G: Learn the key differences between them. `https://www.techtarget.com/searchnetworking/feature/A-deep-dive-into-the-differences-between-4G-and-5G-networks`.

[45] WEAVIX. The Benefits of Deploying a Temporary Private LTE Network. `https://blog.weavix.com/temporary-lte-network-benefits`.