Sinfonia: Cross-Tier Orchestration for Edge-Native Applications

Mahadev Satyanarayanan, Jan Harkes, Jim Blakley,
Marc Meunier[†], Govindarajan Mohandoss[†], Kiel Friedt[†],
Arun Thulasi[‡], Pranav Saxena[‡], Brian Barritt[‡]

[†]ARM Inc. [‡]Meta Inc.

July 2022
CMU-CS-22-125

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This research was sponsored by National Science Foundation award number 2106862, by ARM and Meta, and by gifts from many contributors to the Living Edge Lab at Carnegie Mellon University. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring entity or the U.S. government.

Keywords: edge computing, cloud computing, mobile computing, pervasive computing, IoT, offload, cyber foraging, Kubernetes, virtual machines, containers, augmented reality, video analytics, autonomous drones, edge AI, machine learning, wearable cognitive assistance, GPU, flow-aware, load-aware, wireless networks, 5G, low latency, jitter, bandwidth adaptation

Abstract

The convergence of 5G wireless networks and edge computing enables new *edge-native applications* that are simultaneously bandwidth-hungry, latency-sensitive, and compute-intensive. Examples include deeply immersive augmented reality, wearable cognitive assistance, privacy-preserving video analytics, edge-triggered serendipity, and autonomous swarms of featherweight drones. Such edge-native applications require network-aware and load-aware orchestration of resources across the cloud (Tier-1), cloudlets (Tier-2), and device (Tier-3). This paper describes the architecture of *Sinfonia,* an open-source system for such cross-tier orchestration. Key attributes of Sinfonia include:

- support for multiple vendor-specific Tier-1 roots of orchestration, providing end-to-end runtime control that spans technical and non-technical criteria;
- use of third-party Kubernetes clusters as cloudlets, with unified treatment of telco-managed, hyperconverged, and just-in-time variants of cloudlets;
- masking of orchestration complexity from applications, thus lowering the barrier to creation of new edge-native applications.

We describe an initial release of Sinfonia (`https://github.com/cmusatyalab/sinfonia`), and share our thoughts on evolving it in the future.

1 Introduction

The roots of Edge Computing reach back over a decade [8]. It was motivated by the observation that the *consolidation* of cloud computing has negative consequences. Consolidation lengthens network round-trip times (RTT) from mobile users and increases cumulative ingress bandwidth demand from Internet of Things (IoT) devices. These negative consequences stifle the emergence of new real-time, sensor-rich applications such as *deeply-immersive augmented reality (AR)* and *streaming video analytics*. Edge computing creates the 3-tier architecture [7, 9] shown in Figure 1. The *network proximity* of Tier-2 to Tier-3, and the cloud-like computing resources available at Tier-2, together enable new *edge-native applications* that are simultaneously bandwidth-hungry, latency-sensitive, and compute-intensive. By enabling offloading of compute-intensive operations at very low latency to Tier-2, edge computing helps Tier-3 to overcome stringent design constraints such as weight, size, battery life, and heat dissipation. It also improves bandwidth scalability by avoiding excessive bandwidth demand anywhere in the system.

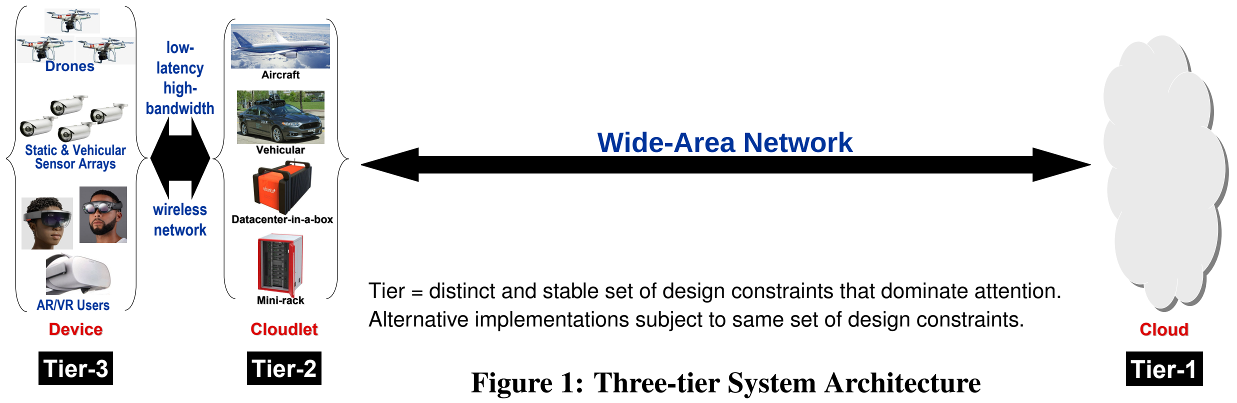

Drones
Static & Vehicular Sensor Arrays
AR/VR Users
Device
Tier-3
low-latency high-bandwidth
wireless network
Aircraft
Vehicular
Datacenter-in-a-box
Mini-rack
Cloudlet
Tier-2
Wide-Area Network
Tier = distinct and stable set of design constraints that dominate attention.
Alternative implementations subject to same set of design constraints.
Cloud
Tier-1

Figure 1: Three-tier System Architecture

Sinfonia is an open source system that enables an app launched on a Tier-3 device to find and dynamically associate with its software back-end on a Tier-2 cloudlet. This association is transient, and it may involve dynamic provisioning and launching of the back-end software on the chosen cloudlet. The association is typically stable for periods ranging from a few minutes to a few hours, and may be broken for many reasons: e.g., the app is terminated, the device moves by a large distance, the cloudlet becomes overloaded, etc. Sinfonia can then be used to find a new cloudlet. It can also be proactively invoked to prepare for a seamless handoff across cloudlets.

2 Problem Statement

The discovery and association problem addressed by Sinfonia is framed by Figure 2. Myriad Tier-3 devices such as drones, video cameras, and wearable or hand-held devices are widely dispersed at planet scale. These are typically owned and self-managed by individuals or corporate entities. A large number of cloudlets, deployed by various service providers using diverse business models, are also widely dispersed. For simplicity, Sinfonia assumes that every cloudlet is a Kubernetes cluster. In contrast to serverless functions, such as those supported by AWS GreenGrass (`https://aws.amazon.com/greengrass/`), Sinfonia is meant for settings where the back-end may involve non-trivial state. The debate between stateless and stateful implementations of distributed systems is an old one, going back over 30 years [6]. In Sinfonia, the one-time cost of cloudlet discovery

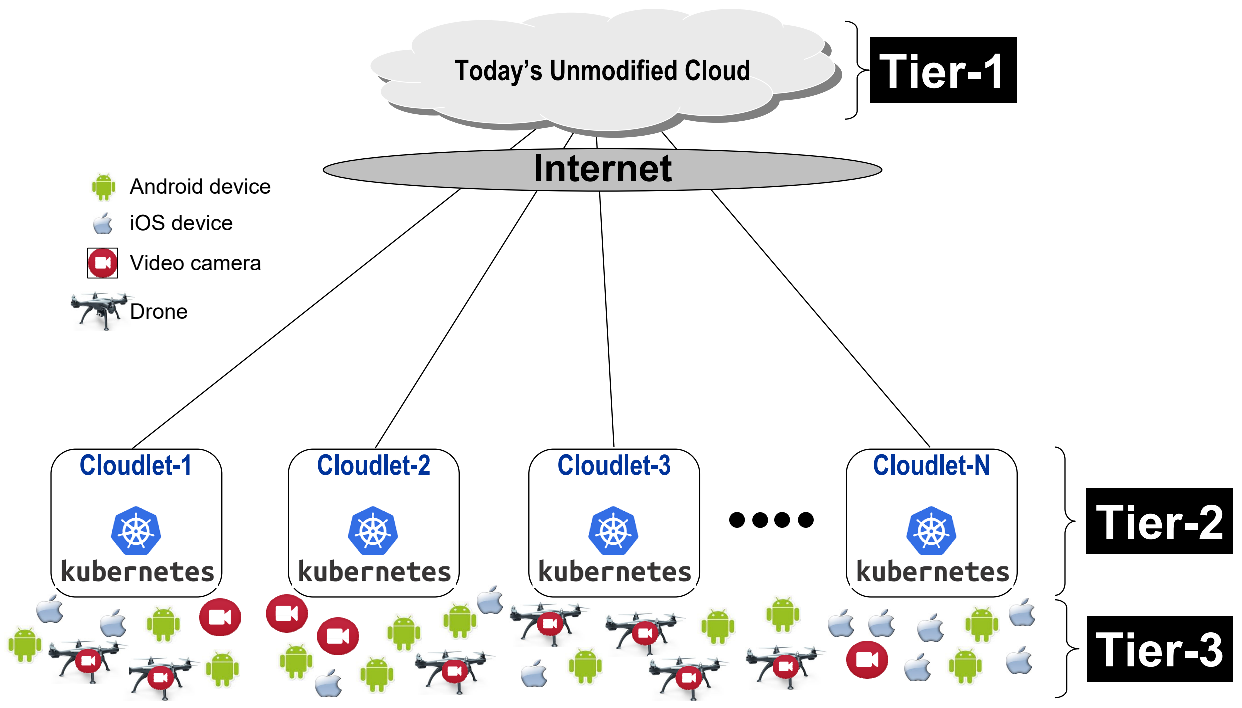


Figure 2: Associating Devices (Tier-3) with Cloudlets (Tier-2)

and provisioning is amortized over many operations that are performed during the lifetime of the association. The cloudlet discovery problem can be simply stated as follows:

> *For an app launched on a Tier-3 device, which is the optimal cloudlet for offloading operations in the near future?*

This simple question has a complex answer because "optimal" is a slippery term. First, it is important to distinguish between *network proximity* and *physical proximity*. As Figure 1 indicates, edge computing requires network proximity (i.e., low latency and high bandwidth) between Tier-3 and Tier-2. Physical proximity is neither necessary nor sufficient to ensure network proximity. At the speed of light in fiber, one millisecond translates to 200 km of physical distance. With a 5G first hop below 5 milliseconds one-way, an end-to-end RTT below 15 milliseconds can be achieved with a cloudlet that is physically quite far away. This is low enough for deeply immersive AR. In fact, this physically distant cloudlet may be a better choice than a nearby cloudlet with a heavily-loaded ingress network. That said, it is typically the case that more distant cloudlets are reached over more network hops. Each network hop is a potential bottleneck, and adds some queueing delay. Hence, there is a weak correlation between physical and network proximity.

Second, network proximity alone cannot ensure choice of an optimal cloudlet. What matters is total end-to-end offloading performance. In AR jargon, this is referred to as *motion-to-photon latency (MTPL)*. It is a metric consisting of both transmission and processing components, and is measured at a Tier-3 device by benchmarks such as OpenRTiST [4]. In this benchmark, MTPL consists of (a) pre-processing and encoding of a video frame captured at a Tier-3 device; (b) uplink transmission of the frame from Tier-3 to Tier-2, including any delays due to contention on a shared wireless network; (c) processing at Tier-2, including queuing delays and cache/TLB interference on shared cloudlet resources such as CPUs, GPUs and memory; (d) downlink transmission of the transformed video frame, including possible network contention delay; (e) decoding and rendering at the Tier-3 device. Items (a) and (e) are fixed for a given Tier-3 device. Network proximity only affects items (b) and (d); it does not impact item (c). The relative importance of (c) versus (b)+(d) depends on how much processing needs to be done, versus the amount of data that needs to be transferred. This is clearly application-dependent. In some cases (e.g., recognition of all faces in an image), it is also highly data dependent; in other cases (e.g., inference for scene classification by a deep neural network (DNN)), it is not. The available hardware resources at Tier-2 also matter. For certain algorithms (e.g., DNN inferencing), a GPU makes a huge difference; for others (e.g., a Viola-Jones face detector [13]), a GPU is irrelevant. In summary, MTPL is a very complex dynamic metric with no simple mapping to static metrics of processing or networking. This can lead to counter-intuitive outcomes. As an example, for DNN inferencing on the Android device at the bottom extreme left of Figure 2, Cloudlet-2 rather than the more powerful and closer Cloudlet-1 may be the better choice at the moment because Cloudlet-1's GPU is heavily overloaded. However, for a different app that does not use a GPU, Cloudlet-1 may indeed be the better choice.

Third, non-technical factors may influence the definition of "optimal." For example, app creator A may have a business relationship with cloudlet provider B. As a result, it may be more profitable for A to offload its app to a cloudlet from B rather than to one from C or D. As long as B's cloudlet can provide an adequate QoE for A's user (e.g., meet the acceptable MTPL bound for the app), technical optimality of the alternative cloudlets is irrelevant. In general, non-technical factors may span a wide range of business, legal and societal considerations. Consider, for example, the futuristic scenario presented in Figure 3. In this scenario, Ron sees no reduction in QoE but the AR gamers suffer poorer QoE. This resource allocation policy is the moral equivalent of reserved parking spaces for the disabled in everyday life. It is an example of *differentiated QoE* across users. More generally, the ability to offer differentiated QoE enables new business models. Higher-paying customers can be offered a better definition of "optimal cloudlet" than lower-paying customers.

3 Solution Requirements

Based on the discussion in Section 2, as well as current trends in edge and cloud computing, we list below the requirements of a good cloudlet discovery mechanism.

First and foremost, such a mechanism must allow end-to-end control. Edge-native applications are diverse, complex, and stateful, with behavior and resource needs that defy simple characterization. New releases of an app may change its behavior and resource needs. Attempting to characterize such complexity in declarative form is unlikely to be successful. The logic to choose from available options is best expressed as code provided by the app developer. Sinfonia should center the discovery process around the invocation of such app-specific code.

Second, emerging edge computing trends should be leveraged. Clusters based on Kubernetes orchestration are emerging as a *de facto* standard for cloudlets. While alternatives such as OpenStack are available, there is value in focusing an initial implementation on a single underlying abstraction. In the Kubernetes ecosystem, the Prometheus resource monitoring subsystem (`https://github.com/prometheus-operator/kube-prometheus`) has gained traction. Monitoring by Sinfonia for within-cloudlet orchestration should leverage Prometheus, if possible.

Third, a diversity of Tier-2 business models is emerging, with no clear winners yet in terms of specific models or companies. This may continue for a number of years, before there is clarity

Ron is a young veteran with combat-related brain injury. He is extremely forgetful, often unable to recognize friends and relatives, and frequently neglects daily tasks. He is helped by "Clarence," a wearable edge-native assistive system. Ron takes the bus this morning to meet his friends. Clarence recognizes his bus number, and alerts him as it approaches. On the bus, Ron is able to smile and greet acquaintances because Clarence promptly whispers their names. As his stop nears, Clarence reminds him to get off, and guides him to the coffee shop where he is meeting his friends. A large group of teenagers are there, playing a deeply immersive multi-user AR game. Ron's arrival overloads the coffee shop's cloudlet. Although Clarence continues to work well, the gamers receive an alert message that they will be receiving lower-quality video and lowered crispness of game controls. Neither the gamers nor Ron are aware of how much happens under the covers to effect these changes. Ron's friends arrive soon, and one of them (Pam) is also assisted by Clarence. With Pam's arrival, the cloudlet in the coffee shop is so oversubscribed that its only recourse is to shed load. All the gamers receive a message that their game will have to be briefly interrupted. Within one minute, their shared game state has been seamlessly migrated to the cloud. Execution is resumed there with game immersiveness settings changed to reflect the much higher end-to-end latency, and the teenagers resume their AR game. After visiting for an hour, Ron and his friends leave. The gamers' back-end state is seamlessly migrated back to the coffee shop cloudlet, and they receive an alert about the improvement in immersiveness that they will shortly see. Soon they are back to enjoying their AR game in full fidelity ...

Figure 3: An Everyday AR Scenario from 2030

in this space. Major Tier-1 players themselves embrace this diversity regarding Tier-2. AWS Wavelength (https://aws.amazon.com/wavelength/), for example, extends Tier-1 abstractions to Tier-2 sites that are physically operated by telco entities such as Vodafone and Verizon [14, 1]. In contrast, AWS Snowball Edge (https://docs.aws.amazon.com/snowball/index.html) and Microsoft Azure Stack Hub (https://azure.microsoft.com/en-us/products/azure-stack/hub/) are physically located on customer premises. Sinfonia should embrace this diversity of deployment models, and minimally constrain current and future business models in the open-source space. In particular, it should enable participation by any entity that wishes to stand up a Tier-2 presence, even if it has no Tier-1 or Tier-3 presence. Exactly how their business models might work remains an open question. Blockchain-based micropayment solutions may play a role here. To accelerate the roll-out of edge computing and thereby catalyze the creation of new edge-native applications, Sinfonia should aim to simplify participation by any Tier-2 entity. It should support the case of a Tier-3 device owner installing a standalone cloudlet on her private wireless network, and seamlessly enabling its discovery as the optimal cloudlet. Canonical's "Ubuntu Orange Box" was an early example of such a hyperconverged cloudlet [12]. Although this product is now discontinued, we posit that similar just-in-time cloudlets will be widely used in the future (Section 4.4).

Finally, independent of Sinfonia, many proprietary ecosystems for edge computing are emerging. Examples include AlefEdge (https://www.wearealef.com/), Equinix (https://www.equinix.com/), Nodeweaver (https://nodeweaver.eu/), and StackPath (https://www.stackpath.com/). These *walled gardens* fragment the edge ecosystem. At the same time, their presence benefits edge computing as a whole. Especially in the early days of edge computing, when coverage by major players is poor, it is valuable to have many service providers with different coverage areas and service niches. For the largest market and easiest deployability, edge-native applications should be usable with any of these walled gardens. Unfortunately, this complicates the apps, making their initial release and maintenance harder. Sinfonia has the opportunity to simplify life for the app developer. By subsuming the complexity of these diverse walled gardens and providing a single unified interface, Sinfonia can lower the barrier to entry of new edge-native applications. This is analogous to printer drivers in an operating system that allow diverse printers, each with its unique proprietary technology, to coexist; a document can be easily printed on any of them. If a developer writes code using the Sinfonia API, the resulting app should be able to use any supported walled garden with no code modifications. In this situation, Sinfonia can treat the entire orchestration process of the walled garden as a black box. It need only act as an authorized agent of the app.

4 Sinfonia Design and Implementation

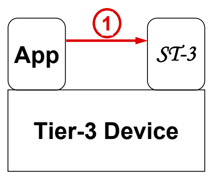
Befitting its role as a cross-tier mechanism, Sinfonia has code components that reside at Tier-1, Tier-2 and Tier-3. We refer to these as *ST-1*, *ST-2* and *ST-3* respectively. The workflow begins when an app at Tier-3 is launched, and requests *ST-3* on its device for a backend on a cloudlet. *ST-3* returns a short list of plausible targets, much like hostname lookup in DNS. We interpret "short" to mean "3 or less;" in many cases, it may just be one. Each target is a VPN endpoint to access the private IP address space where the backend is deploying. Sinfonia views all choices in the short list as "good enough" to meet the app's stated requirements.

The narrowing of the short list to one element may be done in many ways. A simple-minded app may always pick the first element. A slightly more sophisticated app may randomize its choice, with the goal of load balancing. An app that is very particular about the end-to-end properties of offloading can conduct runtime measurements (e.g., pings, test data transfers, and test operations) on each target and use that information to make the final choice. Providing this level of flexibility provides true end-to-end control. It recognizes the fact that cross-tier orchestration at Internet scale is necessarily best-effort rather than truly optimal. The constraints of latency and scalability induce uncertainty in *ST-1*'s knowledge of current system-wide conditions. In addition, current or past conditions may not be indicative of future conditions. Thus, Sinfonia's decision making is inherently fraught with uncertainty. In the face of this uncertainty, the next three sections describe how Sinfonia components work together to generate a short list of targets. Figure 4 illustrates the steps involved in this workflow.

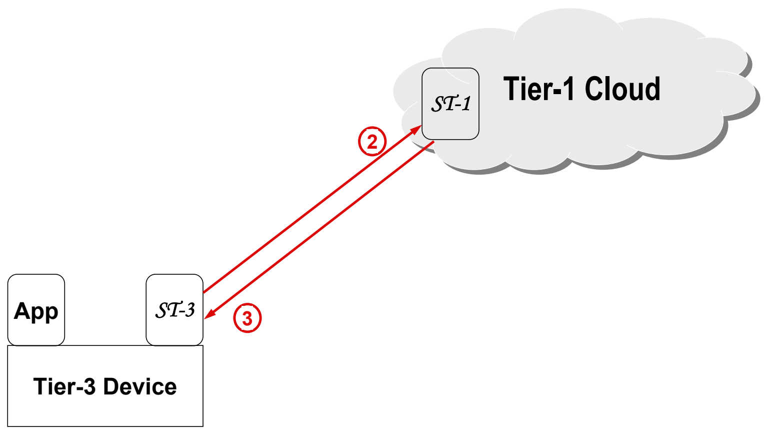
4.1 Workflow at Tier-3

Linux and Android implementations of *ST-3* are provided. The Sinfonia workflow begins when an app on a Tier-3 device invokes the `deploy_backend` operation on that device's *ST-3* (Figure 4(a)). This operation could equally well be named `find_cloudlet,` because it returns a short list of "optimal" cloudlets with already-launched backends that are ready for use by this app. We focus on the discovery aspect in this section, deferring to Section 5 the discussion of alternative ways of provisioning the chosen cloudlets with the exact backend needed by this app.

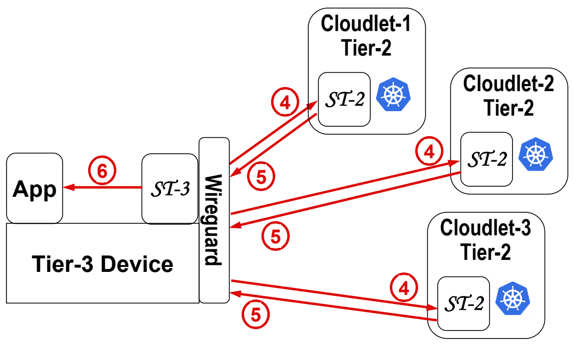
An input parameter provided by the app to the `deploy_backend` call is the URL of the *ST-1* to use as the root of orchestration. The developer of the app embeds this URL in its source code. There is thus true end-to-end control of the orchestration process — it is the app developer who



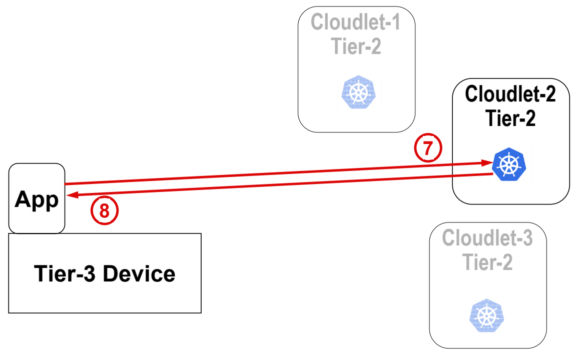
(a) App invokes `deploy_backend`



(b) $\mathcal{ST}$-3 requests short list from $\mathcal{ST}$-1



(c) Private networks set up to cloudlets on short list



(d) App offloads work to chosen cloudlet

Figure 4: Cross-Tier Workflow of Cloudlet Discovery and Association

has ultimate control over which code will be used for the orchestration process. We expect that each major software vendor (e.g. Electronic Arts, Microsoft, Adobe, Meta, etc.) will have its own $\mathcal{ST}$-1 at Tier-1. However, the mechanism is open-ended and flexible enough to support cross-organization sharing of a single $\mathcal{ST}$-1, if that is desired. This may be a future business opportunity for Internet registrars such as NameCheap and GoDaddy.

Also provided by the app in the `deploy_backend` call is a UUID that uniquely identifies this specific version of the app. $\mathcal{ST}$-3 forwards the app's request to $\mathcal{ST}$-1 after adding device-specific details such as its current geo-location, its current networking environment (e.g., IP address), and device hardware attributes. Thus, $\mathcal{ST}$-1 receives all the contextual Tier-3 knowledge needed for making a good decision (Figure 4(b)). Section 4.2 describes how $\mathcal{ST}$-1 obtains the Tier-2 knowledge needed. The output returned by $\mathcal{ST}$-1 to $\mathcal{ST}$-3 is a short list of public IP addresses of cloudlets. $\mathcal{ST}$-3 uses the Wireguard VPN mechanism [3] to bind a private network to a target cloudlet (Figure 4(c)), and associates the VPN with the application. The application can perform any runtime performance tests it desires. It can then try another target until it finds the optimal one. From that point onwards, the app directs offloading requests to the chosen backend (Figure 4(d)). The unused backends, if any, are asynchronously garbage collected.

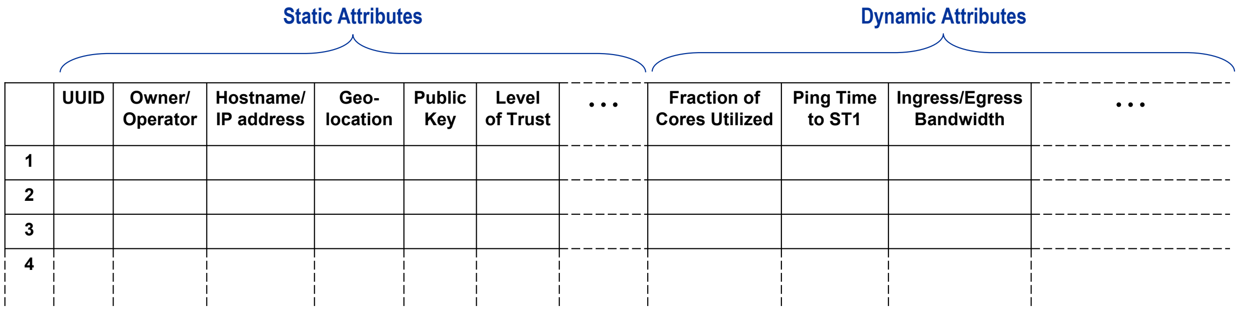
4.2 Workflow at Tier-1

There is considerable flexibility in how *ST-1* is implemented. For simplicity, our description here assumes a monolithic implementation in the cloud, using well-known techniques for scale-out, fault-tolerance and high availability. We expect that many hyperscalers will provide such implementations as part of their standard cloud services. Not discussed here is a possible implementation of *ST-1* as a collection of software-defined networking (SDN) applications that are located on the northbound interface of SDN controllers to benefit from a graph-based workflow of a Tier-2 topology. Such an SDN-based implementation would allow *ST-1* to estimate network proximity from live measurements rather than relying on the geolocation of *ST-3* and the weak correlation between physical and network proximity.

To handle incoming cloudlet discovery requests, *ST-1* maintains a large data structure called "CloudletTable" (Figure 5). As its name implies, this is a two-dimensional array in which each row corresponds to a cloudlet known to *ST-1* and identified by a UUID. The columns correspond to attributes of that cloudlet. Some of these attributes may be static. Examples include owner/operator of the cloudlet, its hostname or public IP address, its public key, its hardware configuration (including acceleration hardware such as GPUs), and (usually) its geolocation. In the case of a cloudlet on a moving platform (e.g., truck, ship or aircraft), the geolocation will be a dynamic rather than static attribute. Other examples of dynamic attributes include current load, utilization of CPU cores and GPUs, memory utilization, and ingress and egress bandwidth utilization. Software resources (e.g., presence or absence of a large ML model) may also be viewed as attributes; in a system that caches such state, these would be dynamic rather than static attributes.

We expect the actual algorithm that produces a short list of cloudlets from CloudletTable to be a topic of intense experimentation and empirical exploration. The literature is full of theoretical results on optimal node selection for offloading in a distributed system. Few of these algorithms have real-world relevance. To simplify experimentation and easy customization of this algorithm in live deployments, it is invoked as an upcall from *ST-1* to external code. The rest of *ST-1* views this algorithm as a black box; all that matters is that it produce a short list of cloudlets that would be good offloading sites for the app that initiated this request. Parameters of the upcall include the CloudletTable, as well as the app and device information provided by *ST-3*. The default selection algorithm provided by Sinfonia uses CPU/GPU utilization and ingress/egress bandwidth as the primary variables of interest. We expect this default algorithm to evolve significantly in the light of actual usage experience with new edge-native applications.

Given the importance of CloudletTable, maintaining it becomes one of the primary responsibilities of *ST-1*. This maintenance is based on a push model: i.e., *ST-1* relies on *ST-2*s to announce



	Static Attributes							Dynamic Attributes			
	UUID	Owner/ Operator	Hostname/ IP address	Geo-location	Public Key	Level of Trust	• • •	Fraction of Cores Utilized	Ping Time to ST1	Ingress/Egress Bandwidth	• • •
1											
2											
3											
4											

Figure 5: Conceptual View of CloudletTable at *ST-1*

the availability of their cloudlets for service, and to periodically provide dynamic attributes of the cloudlets. These periodic updates effectively serve as keepalives. If *ST-1* does not hear from an *ST-2* for longer than a pre-determined timeout period, it assumes that the cloudlet is no longer available and deletes that row from CloudletTable. When *ST-1* hears from a new cloudlet (possibly one that was dropped earlier because of a timeout), it extends CloudletTable with this new entry. This onboarding of a brand new cloudlet may involve additional steps to insure integrity and security, as discussed in Section 4.3.

4.3 Workflow at Tier-2

After a cold start of *ST-1*, its CloudletTable is empty. The filling of this table occurs incrementally, as *ST-2*s contact this *ST-1* and each indicates that it is able and willing to serve as a cloudlet. An *ST-1* is free to accept or decline the services of an *ST-2*. Conversely, an *ST-2* is free to concurrently offer its services to many *ST-1*s. It is not obligated to offer its services to any specific *ST-1*. The relationship between an *ST-1* and an *ST-2* thus resembles a free market. We expect that all *ST-1*s will be widely known (e.g., possibly listed in a global web page), but cloudlets may be more regional in ownership, management, and coverage.

If an *ST-1* welcomes an *ST-2*, it creates an entry in its CloudletTable for that cloudlet and fills its static attributes. It also fills initial values of the cloudlet's dynamic attributes. Relevant billing-related initialization is also performed at this point. This is the point at which mutual trust is established. The procedure for establishing mutual trust may evolve as Sinfonia is developed, and may include use of mechanisms such as signed certificates and trusted enclaves. The level of trust that an *ST-1* has in a cloudlet may be a static attribute in its CloudletTable entry for that cloudlet.

This dynamic, decentralized, and bottom-up binding of cloudlets to roots of orchestration draws inspiration from the early evolution of the World Wide Web. A critical catalyst to that explosive early growth was the fact that anyone, anywhere could set up a web server and publish pages on it. No centralized permission or authorization was necessary. In like vein, we wish to enable any entrepreneurial entity to easily set up a cloudlet if it sees a business opportunity. The push model of cloudlets marketing their services to *ST-1*s is consistent with this vision. Various *ST-1*s may or may not choose to do business with a specific cloudlet. Such policy decisions are best left to each *ST-1*, and do not have to be centralized.

We believe that Sinfonia's approach will scale well. We expect relatively few *ST-1*s worldwide, perhaps on the order of $10^2 - 10^3$. This figure is consistent with the published estimate of roughly 10^5 independent software vendors (ISVs) in existence worldwide today. A few are large ISVs (e.g., Microsoft, Adobe, Amazon, Meta, Apple, Electronic Arts, etc.) that may each run their own *ST-1*. But many more are very small, and are likely to outsource their *ST-1*s to shared registrars. Since we are only in the early stages of edge computing rollout, it is hard to predict how many cloudlets there will be worldwide. Our current best guess is that there will eventually be on the order of $10^5 - 10^6$ cloudlets worldwide. Since the inhabited area of the earth is estimated to be about 25×10^6 square miles, this suggests a coverage area between 25–250 square miles per cloudlet. This estimate may be easily off by an order of magnitude, since there may be competing cloudlets covering the same area and their distribution will almost certainly be clumpy rather than uniform. In spite of this uncertainty, the estimate of $10^5 - 10^6$ cloudlets seems reasonable from two other viewpoints. First, there are on the order of 10^4 cities worldwide, and we expect at least a few cloudlets per city eventually. Second, there are estimated to be 6×10^9 smartphones worldwide today. If 10% are used as Tier-3 devices for edge-native applications, a fan-in of 10^2 devices per cloudlet is plausible.

After initial registration, an *ST-2* is expected to periodically send *ST-1* a keepalive with current values of dynamic attributes. The frequency of these keepalives is negotiated at registration. As mentioned earlier, if an *ST-1* does not hear from a registered *ST-2* beyond some timeout period, it deletes that *ST-2* from its CloudletTable.

4.4 Just-in-Time Cloudlets

We envision many use cases in which a cloudlet is set up for temporary use at a work site. Examples include military settings, disaster recovery settings, outdoor construction sites, and drone operations for infrastructure maintenance. In this case, the owner of Tier-3 devices (e.g., drones) is also providing the cloudlet to be used by them. Sinfonia supports such just-in-time (JIT) cloudlets by having them configured to be on the same private wireless network as the Tier-3 devices. Using a Zeroconf discovery protocol analogous to Bonjour, *ST-3*s on these devices can discover the JIT cloudlet. This discovered JIT cloudlet can be added by an *ST-3* to the short list returned by *ST-1*, most likely as the very first element of the list. Simple apps that just use the first entry in the list will always select the JIT cloudlet. More sophisticated apps can treat it as just one candidate in their final selection process. It is also possible to configure *ST-3* so that it completely avoids contacting *ST-1* if a JIT cloudlet is discovered. This leads to a form of static scoping: a JIT cloudlet is deemed to always be a better choice than any infrastructure cloudlet.

From the viewpoint of an app, use of a JIT cloudlet is identical to its use of any other cloudlet. No configuration or software changes to the app are necessary. It may be helpful in some use cases that the JIT cloudlet is in the same trust domain as the Tier-3 device. Especially during the early evolution of edge computing, when coverage by service providers is patchy and limited, the ability to easily setup a JIT cloudlet will be a valuable enabler for edge-native applications. For example, a JIT cloudlet in the home may enable an elderly person to use an assistive AR app on a wearable device. That elderly person can benefit from the app long before 5G coverage or telco-provided edge computing appears in his or her neighborhood.

5 The Road Ahead

At the time of writing this document, an initial version of Sinfonia has been released on GitHub. This is a bare-bones implementation, sufficient for the workflow described here but without many optimizations that we envision adding later. The goal of this release is to "fail early" — i.e., to help us gain hands-on experience with the cross-tier orchestration approach of Sinfonia, and to expose flaws in our thinking as soon as possible. Some of the key assumptions and mechanisms that we hope to validate through hands-on usage include:

- the value of end-to-end control and multiple roots of orchestration.
- the value of making the cloudlet selection algorithm a field-replaceable external module.
- the ease with which new cloudlets can be added, and the practical issues involved in sustaining a market-oriented approach to Tier-2.

- the simplification provided by Sinfonia to developers of new edge-native applications, and the QoE that it helps them achieve.
- the ability to integrate multiple walled gardens into Sinfonia, and the unique opportunities and challenges of using diverse types of cloudlets.
- implementation of cloudlet selection algorithms published in the literature, and experimental validation of their claims to optimality.

Evolution beyond the bare-bones implementation will depend on our learnings from hands-on usage, as well as the broader evolution of the ecosystem for edge-native applications. We see a number of long-term research thrusts and development initiatives arising from this work. We list these below in no particular order, mainly as placeholders to capture these thoughts for the future.

Provisioning: Section 4.1 is silent on the exact details of how provisioning of the backend for a specific app occurs on a chosen cloudlet. In practice, there are many approaches that could work well. Only usage experience will tell which of these approaches strikes the optimal balance between reliability, speed, flexibility and ubiquity. The simplest approach is static provisioning, in which the presence of the precise backend for an application is one of the cloudlet's static attributes. A slightly more general approach is for provisioning details to be part of the app-specific knowledge possessed by *ST-1*, and for these details to be conveyed to the *ST-2*s of the chosen cloudlets. The latter may fetch the relevant bits if they are not already present. This cache state would be a dynamic attribute of the cloudlet. The most general approach is *Just-in-Time Provisioning* as described by Ha et al [5]. In this approach, the app provides (by value) a bag of bits called an *overlay* that can be applied to a *base image* to rapidly synthesize a bit-exact backend at runtime. It pushes the end-to-end aspect of Sinfonia to its extreme, and may have particular value for a JIT cloudlet that has no connectivity to the Internet. Such a disconnected JIT cloudlet may have been discovered via a Zeroconf protocol.

Privacy: Implicit in the offloading of processing from Tier-3 to Tier-2 is a widening of the privacy perimeter of data captured at a device. Since Tier-1 (via *ST-1*) only participates in the control plane of offloading and not in its data plane, the privacy perimeter does not expand to Tier-1; it only expands to Tier-2. But even this limited off-device expansion of the privacy perimeter may be unsettling for some users, or illegal in some jurisdictions. One possible solution is to carefully choose the cloudlets permissible for sensitive applications, by using the static attributes of CloudletTable. A different approach is to use the concept of *denaturing* [2, 11, 15], in which real-time policy-guided transformation of raw data is performed to preserve privacy. For example, the wearable devices of the gamers in the scenario of Figure 3 could black out all human faces before transmitting video to their cloudlets. ASIC hardware for face detection is cheap and fast enough for this today. By doing this, bystanders in the coffee shop are not exposed by the gamers to Tier-2. Unfortunately, this approach does not help Ron in Figure 3, because he needs help to recognize the faces of people who greet him. Fortunately, there is empirical evidence that this is indeed an acceptable tradeoff in the context of assistive technologies. Siewiorek et al [10], for example, report that "people will trade privacy for enhanced capability and the ability to maintain independence." The bottom line is that privacy is a huge and sensitive topic in the context of edge-native applications, and will need careful thought and attention to be paid to the policies and mechanisms that are developed.

Micropayments and Trust: The "free market" relationship between Tier-1 and Tier-2 sketched in Section 4.3 suggests many opportunities for developing blockchain-based micropayment business models for edge offload. In the networking world, companies like Helium ((`https://www.helium.com/`)) have pioneered such business models. However, safeguarding the integrity of offloading is a challenging problem. Freedom and open-endedness bring with them questions of trust. How does an $\mathcal{ST}$*-1* gain confidence that a particular cloudlet has adequate integrity to return as a possible target? How does it guard against Byzantine cloudlets? One approach could possibly be for $\mathcal{ST}$*-1* to conduct attestation-based validation of a target cloudlet before it returns control to $\mathcal{ST}$*-3*. Mechanisms such as secure enclaves become especially valuable in this context. The problem is even more challenging if a JIT cloudlet with no Internet connectivity is discovered via a Zeroconf protocol. $\mathcal{ST}$*-3* is then on its own; there is no $\mathcal{ST}$*-1* to help. Such situations may prevail in disaster recovery and military settings. Clearly, there are many research questions to be addressed in this space before the vision of a safe micropayment-based ecosystem for offloading becomes viable.

Network Protocols for Edge Offload: The application-level protocols between Tier-3 and Tier-2 represent a unique opportunity. Today's application protocols have been developed for the Internet, where worst-case bandwidth may be low and latency high. However, the whole point of edge computing is to ensure network proximity. New edge-native applications thus have an opportunity to rethink from first principles how the assumptions of low latency and high bandwidth can benefit the protocols that they use for offloading.

In closing, the convergence of 5G wireless networks and edge computing enables many new edge-native applications that are simultaneously bandwidth-hungry, latency-sensitive, and compute-intensive. Such applications require network-aware and load-aware orchestration of resources across tiers. We have created Sinfonia, an open-source system for such cross-tier orchestration. This paper has described its driving vision, high-level assumptions and initial implementation. Sinfonia opens the door to many opportunities for experimental research and business innovation.

References

[1] ASHRAF, C. Verizon and AWS expand edge computing to 19 U.S. metro areas. (https://www.verizon.com/about/news/verizon-aws-expand-edge-computing-19-us-metro-areas), May 2022.

[2] DAVIES, N., TAFT, N., SATYANARAYANAN, M., CLINCH, S., AND AMOS, B. Privacy Mediators: Helping IoT Cross the Chasm. In *Proc. of ACM HotMobile 2016* (St. Augustine, FL, February 2016).

[3] DONENFELD, J. A. WireGuard: Next Generation Kernel Network Tunnel. In *Proceedings of the 24th Annual Network and Distributed System Security Symposium, (NDSS)* (February 2017).

[4] GEORGE, S., EISZLER, T., IYENGAR, R., TURKI, H., FENG, Z., WANG, J., PILLAI, P., AND SATYANARAYANAN, M. OpenRTiST: End-to-End Benchmarking for Edge Computing. *IEEE Pervasive Computing 19*, 4 (2020), 10–18.

[5] HA, K., PILLAI, P., RICHTER, W., ABE, Y., AND SATYANARAYANAN, M. Just-in-Time Provisioning for Cyber Foraging. In *Proceedings of MobSys 2013* (Taipei, Taiwan, June 2013).

[6] OUSTERHOUT, J. K. The Role of Distributed State. In *In CMU Computer Science: a 25th Anniversary Commemorative* (1991).

[7] SATYANARAYANAN, M. The Emergence of Edge Computing. *IEEE Computer 50*, 1 (January 2017).

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

[9] SATYANARAYANAN, M., GAO, W., AND LUCIA, B. The Computing Landscape of the 21st Century. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications (HotMobile '19)* (Santa Cruz, CA, 2019).

[10] SIEWIOREK, D., SMAILAGIC, A., AND DEY, A. Architecture and Applications of Virtual Coaches. *Proceedings of the IEEE 100*, 8 (2012).

[11] SIMOENS, P., XIAO, Y., PILLAI, P., CHEN, Z., HA, K., SATYANARAYANAN, M. Scalable Crowd-Sourcing of Video from Mobile Devices. In *Proceedings of the 11th International Conference on Mobile Systems, Applications, and Services (MobiSys 2013)* (June 2013).

[12] VAUGHAN-NICHOLS, S. Canonical's cloud-in-a-box: The Ubuntu Orange Box. (https://www.zdnet.com/article/canonicals-cloud-in-a-box-the-ubuntu-orange-box/), May 2014.

[13] VIOLA, P., AND JONES, M. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)* (2001).

[14] VODAFONE PRESS RELEASE. Vodafone uses AWS Wavelength to launch first Multi-access Edge Computing services in European region. (https://newscentre.vodafone.co.uk/press-release/partnership-aws-wavelength-launch-first-multi-access-edge-computing-services-in-europe/), June 2021.

[15] WANG, J., AMOS, B., DAS, A., PILLAI, P., SADEH, N., AND SATYANARAYANAN, M. A Scalable and Privacy-Aware IoT Service for Live Video Analytics. In *Proceedings of ACM Multimedia Systems* (Taipei, Taiwan, June 2017).