Towards Multi-Stakeholder Clouds

Bohdan Borysei†, Stefan Saroiu*, and Eyal de Lara†
†University of Toronto and *Microsoft

ABSTRACT

Cloud providers are embracing edge computing by developing multi-stakeholder clouds (MClouds), in which the infrastructure is shared among cloud providers, hosting site operators, and, sometimes, system integrators. This form of sharing creates a new set of challenges around infrastructure management. Unfortunately, traditional resource management methods based on partitioning and virtualization do not fit well the needs of infrastructure management in a multi-stakeholder scenario. Our paper describes these new set of challenges, and puts forward a new authorization framework based on two-person control (P^2C).

1 INTRODUCTION

Cloud providers are turning to edge computing in an effort to scale out their services, reduce response times, and comply with data residency requirements. To achieve this, they are starting to build a new form of cloud computing that is subject to *multiple stakeholders*, such as a hosting site operator, a cloud provider, and, sometimes, a system integrator. *Multi-stakeholder clouds (MClouds)*[1] complement traditional datacenter-based infrastructure (known as *public clouds)* and on-premise *private clouds* (Figure 1).

Currently, cloud providers are deploying MClouds for running the workloads of the hosting site operator, whether an individual enterprise [2, 26, 14] or a 5G telco operator [4, 25, 15]. However, the potential of MClouds lies in facilitating multi-tenant scenarios, where tenants share resources in the MCloud like they do with public cloud resources [4, 39].

This paper aims to introduce a new research direction in edge computing: MCloud infrastructure management. We argue that the emergence of the MCloud comes with a new set of systems challenges stemming from its multi-stakeholder nature. In public clouds, the responsibility for infrastructure lies with the cloud provider. The cloud provider owns, manages, and operates all of their servers and switches inside their datacenter. However, in the case of MClouds, the division of responsibilities becomes less clear. Should the cloud provider still handle tasks like powering servers, installing firmware updates, and monitoring server health, or is this now the responsibility of the hosting site's administrator team?

[1]Cloud providers often use the term *hybrid cloud* to refer to the MCloud. However, hybrid cloud sometimes means cloud bursting or cloud composition [36]. We use the term MCloud to avoid such confusion.

HOTMOBILE '24, February 28–29, 2024, San Diego, CA, USA
© 2024 Association for Computing Machinery.
ACM ISBN 979-8-4007-0497-0/24/02…$15.00
https://doi.org/10.1145/3638550.3641135

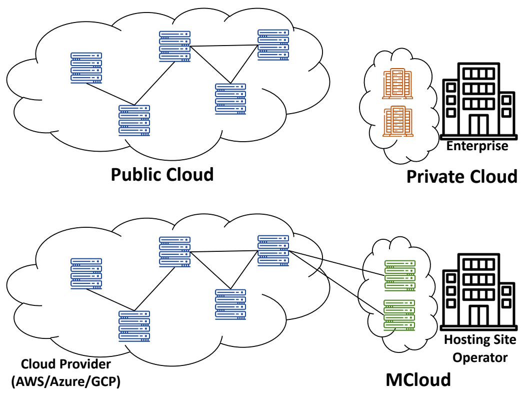

Figure 1: Types of clouds: public and private (top) and multi-stakeholder or MCloud (bottom). With MCloud, the cloud provider and the hosting site operator co-manage infrastructure deployed on site.

This "tussle" arises from the different interests of both stakeholders, which, while not adversarial, can sometimes conflict.

Today's cloud providers have taken radically different approaches to handling infrastructure management. Some cloud providers want almost complete control over the on-premise portion of the MCloud. The hosting site operator is left with rudimentary forms of control over their infrastructure such as plugging the servers into the power outlet or a "panic button" that deletes all their data. Other cloud providers take the opposite approach – they relinquish all control to the hosting site operator. However, when management challenges arise, it remains unclear how these cloud providers would instruct and convince hosting site operators to perform specific infrastructure tasks.

While these two approaches are diametrically opposed, they share a common consequence: they will likely lead to a strained relationship between cloud providers and hosting site operators. These two approaches are effectively forms of *winner-take-all* rather than *collaboration* in which all parties have a *seat at the table* and *skin in the game.* Managing large-scale infrastructure is complicated and prone to errors that may go unnoticed until they trigger problems. These mistakes can have serious consequences, such as infrastructure failures, malware compromises, and data breaches. Moreover, in an MCloud even a seemingly correct action taken by one party can adversely affect the other. For example, a decision to reboot a group of servers by a cloud provider can produce a power spike that the site operator is not able to absorb. Similarly, a firmware upgrade carried out by the site operator can lead to degraded performance for the cloud provider. Since the management responsibilities are not equal, this asymmetry in roles will also result in an imbalance when it comes to assigning blame and seeking damages.

In this paper, we argue that the management of MCloud infrastructure requires the collaboration between different parties. We argue that cloud providers and the hosting site operators must come up with novel ways to share control and responsibility for the infrastructure, from racks and servers to network switches and appliances.

Unfortunately, traditional approaches to sharing cloud resources, such as resource partioning [23, 12, 18] and virtualization [35, 5, 6] are not sufficient for handling infrastructure management operations in MCloud. Partitioning and virtualization, effectively *delegates* a resource to a party who is then responsible for its management. This approach works for cloud tenants because their actions are effectively isolated from impacting other tenants. In contrast the hardware management in MCloud involves low-level operations with side-effects that are system-wide.

This brings us to our position statement: *The MCloud needs a new management primitive that enables multiple stakeholders to collaboratively manage their infrastructure.*

1.1 MCloud Two-Person Control

We propose a two-person control (P^{2}C) method for infrastructure management in the MCloud. With P^{2}C, no single party can independently perform an infrastructure operation. Beyond reducing the likelihood of errors and accidents, P^{2}C disperses control, responsibilities, and accountability between both parties. P^{2}C is widely used in a variety of industries that deal with actions that can have significant consequences in case of mistakes, such as the financial [13, 21, 24], military [33, 1, 9, 28], and pharmaceutical sectors [7].

In its purest form, P^{2}C entails two parties with identical roles, and an operation is executed only when both parties perform or approve it. However, there are more nuanced variations of P^{2}C, where one party can grant approval based on specific policies. For instance, a hosting site operator may wish to monitor the CPU load of servers on their premise to optimize power distribution. A cloud provider might have reservations about fine-grained CPU load monitoring especially when third-party workloads are involved, due to privacy and data security concerns. A compromise solution could involve the cloud provider implementing a filter to coarsen the data, such as categorizing CPU load into high, medium, and low buckets. P^{2}C would then require the application of filters and policies by each party, with the operation's results reflecting the combined policies.

1.2 Bringing P^{2}C to the BMC

We take the first steps into designing a P^{2}C-based approach for a common form of sever management infrastructure: the baseboard management controller (BMC). A BMC is a specialized microcontroller embedded on the motherboard of modern servers that enables remote management and system monitoring independently of the main CPU and OS.

We have implemented a P^{2}C prototype on top of OpenBMC, an open-source BMC implementation targeting dozens of different types of motherboard architectures. Our implementation comprises a small proxy running in the BMC responsible for P^{2}C authentication and a corresponding P^{2}C authorization framework. Our framework implements different types of access schema depending on the nature of the operation. Our prototype intercepts Redfish API calls to the BMC and applies the P^{2}C authorization framework before execution. After the operation terminates, the prototype intercepts the result and, depending on the operation's nature, may apply a filter to the results before returning them.

We tested our prototype in two environments. First, we ran it in a QEMU virtual environment emulating the BMC of a Tyan server running a Palmetto-type motherboard. This is similar to how an OEM hardware provider would implement our system in their BMCs. Second, we ran it as a standalone daemon, remote controlling an 8-server rack, and conducted a limited set of experiments with P^{2}C-based management.

The rest of this paper is organized as follows: Section 2 describes the MCloud landscape, the role of the BMC, and P^{2}C. Section 3 presents a mapping of BMC operations into a P^{2}C authorization framework, and Section 4 describes our prototype. Finally, Section 5 explores potential avenues for future research.

2 BACKGROUND

This section presents background on the multi-stakeholder cloud, the BMC, and two-person control.

2.1 Multi-Stakeholder Cloud (MCloud)

Cloud services are expanding to the edge to achieve scalability, lower latencies, and comply with data residency requirements. A prime example is the shift of 5G workloads towards an MCloud approach. Telecommunication companies (telcos) have begun offloading a significant portion of their cellular network functions to cloud-based platforms, specifically those associated with the packet core [22, 27]. Much of the packet core's functionality is performing control plane operations, which are well-suited for the latencies offered by public cloud services.

At the same time, a significant portion of telco's network functions cannot be offloaded to the public cloud due to latencies or data residency requirements. For example, the radio access network (RAN) software must run close to the radio towers to handle the traffic from the end-user devices [19].

Cloud providers have started developing forms of MCloud tailored to 5G network stacks. However, their approaches to building MCloud infrastructure vary significantly. For example, with AWS Outpost and AWS Wavelength, companies and institutions can purchase servers or even entire compute racks to run at the edge [2]. This hardware is fully built, installed, and operated by Amazon alone. The hosting site operator's responsibilities are limited to meeting specific facility requirements, such as maintaining proper temperature, humidity, airflow, and providing a loading dock [3].

Microsoft's approach is two-pronged. First, they have a turn-key edge server offering known as Azure Stack Edge [26]. A hosting site operator can purchase one or more such servers and install them on premise. Like with AWS Outpost, Azure Stack Edge's hardware is built and operated by the cloud vendor. However, Microsoft also offers a model in which a hosting site operator purchases the entire edge server hardware. This offering is known as Azure Operator Nexus and is aimed at 5G telcos [25]. The server hardware specifications are outlined in a bill-of-materials (BOM) document, which

	Cloud Provider	Hosting Site Operator	System Integrator
AWS Outpost/Wavelength	Builds/Installs/Operates Server/Rack	Provides Site Facility	N/A
Azure Stack Edge	Builds/Operates Server	Installs Server + Provides Site Facility	N/A
Azure Operator Nexus	Provides BoM Spec	Purchases/Installs/Operates Whole Rack	N/A
Google Anthos	Provides Reference Specification	Purchases/Installs/Operates Whole Rack	Builds Whole Rack

Table 1: Multiple stakeholders and their roles in four hybrid cloud examples.

provides a limited set of choices for the types of racks, servers, and networking equipment that can be used. In this case, the hosting site operator purchases, installs and manages all the edge hardware equipment. Once the hardware is set up and operational, the operator proceeds to download and install Microsoft's Linux-based OS, which enables the deployment of cloud services.

The Anthos platform is the MCloud architecture developed by Google. Their approach is to work with several system integrators (SI), such as Dell [31], HPE [16], Lenovo [34], and VMware [38] to develop reference specifications. A hosting site operator purchases turn-key racks from their system integrator partner and takes on the responsibility of installing and operating this hardware on their premises.

These MCloud offerings (shown in Table 1) assign most of the responsibility for handling and managing infrastructure to either the cloud provider or the hosting site operator, sometimes with assistance from a system integrator. This setup makes it challenging to handle low-level infrastructure operations that affect the entire system.

The MCloud market is relatively new and emerging. For instance, AWS Outpost, which is the oldest MCloud, was launched in 2018. Because this market is still evolving, cloud providers are quite eager to work closely with hosting site operators to meet their requirements. However, it is less clear whether these multi-stakeholder relationships will remain harmonious as the market matures.

2.2 Baseboard Management Controller

Modern servers are equipped with a server management processor alongside the main CPUs. This processor functions as a BMC controller, implementing a standard set of management operations that allow administrators to remotely manage and monitor their servers without the need to be physically present near the server.

Generally speaking, BMCs perform three types of operations: server management, monitoring, and logging. Server management includes server reboots and reimages, firmware upgrades, and, in specific server SKUs, console access. Server monitoring encompasses activities like checking the status of the server's CPU, memory, peripherals, as well as monitoring power and thermal aspects. Finally, server logging refers to retrieving event logs from the system.

The set of management interface specifications that a BMC controls is called the *Intelligent Platform Management Interface (IPMI)* [20]. The remote administrator makes IPMI calls to perform BMC operations in an RPC-like fashion. More recently, a new session-based protocol has emerged called Redfish [11]. With Redfish, the remote administrator authenticates and creates a session that comprises of REST API calls corresponding to BMC operations [10]. Redfish supports a simple role-based access control scheme with three roles: administrator, operator, and read-only user. Table 2 summarizes the roles and their capabilities.

Role	Capabilities
Administrator	Server management (w/ console access if available) Server monitoring (w/ debugging if available) Server logging (w/ clearing logs) BMC configuration and administration
Operator	Server management (no console access) Server monitoring (no debugging)
Read-only User	Server monitoring (no debugging)

Table 2: Redfish roles and their capabilities.

2.3 Two-Person Control (P^2C)

Two-person control (P^2C) originates from the financial industry and refers to the concept that no one person is able to unilaterally access a physical asset. Over time, P^2C has been adopted by more modern, emerging industries to protect physical assets [13, 21, 24, 33, 1, 9, 28, 7]. This form of access control is often used in scenarios that deal with actions with significant consequences in case of mistakes.

In computer systems, although P^2C has been occasionally considered as a potential scenario for role-based access control schemes [29], P^2C remains relatively rarely supported in practice. The most relevant earlier work is ISE-T [30], a system designed for configuring standalone machines. With ISE-T, performing a sensitive operation requires two administrators. Each administrator independently issues a set of commands to perform the operation. ISE-T then compares these two sets of commands to determine if they are equivalent. If equivalent, ISE-T automatically executes them. Otherwise, a notification is raised to both administrators who must reconcile the commands through an out-of-band mechanism.

Our use of P^2C of MCloud is much simpler than ISE-T – both individuals must approve a BMC operation before it is carried out. These operations consist of single Redfish calls, not complex sequences of steps. If an individual does not approve an operation, the operation fails with no need of any out-of-band reconciliation. This simplicity should make P^2C for MCloud easier to deploy and adopt in practice.

3 P^2C AUTHORIZATION FRAMEWORK

Our P^2C authorization framework provides three categories:

1. Either-party: Either party can execute an operation without the need of the other's party approval. This is reserved for operations deemed by either party to have no important consequences. Consider the creation of a Redfish session. For this operation, each

Consumed Power	**hso-ml**	**hso-compute**	**cp-5G**
Steady-state	180W	140.7W	249W
During bootup	240W (+33%)	180W (+28%)	317W (+27%)

Table 3: Power consumption spikes during server bootup.

party should be allowed to use their credentials to create a Redfish session independently.

2. Both-parties: Both parties must approve an operation before it can be executed. This is reserved for operations with important consequences and system-wide side-effects.

3. Both-parties-with-policy: Similar to *both-parties but* the operation's inputs and outputs are subject to a policy defined by either party. Consider a Redfish call that reads the server load. The cloud provider may not want to disclose detailed server load data to the hosting site operator, but be willing to provide coarse-grained information, categorizing the load as *low*, *medium*, or *high*. In this case, the cloud provider specifies a policy that maps the operation's output to one of these three server load classes.

We assign BMC Redfish calls [11] to one of these three authorization categories. We perform this assignment for a hypothetical rack server deployed by a major cloud provider on the premises of an academic institution. The academic institution and their team of administrators act as the host site operator. This categorization should not be viewed as fixed and universally acceptable. Instead, the MCloud's multi-stakeholders have the flexibility to determine which P^2C categories are most suitable for their BMCs.

When performing this assignment, we developed policies that depend on factors external to a single server. For example, our hosting site operator (the academic institution sysadmin) had reservations about simultaneously rebooting all nodes in the rack. Their concern was that, at bootup, servers' power usage spikes due to performing device initialization and testing. To validate this concern, we measured the bootup power consumption of three server SKUs:

(1) *hso-ml*: a single-socket 16-core AMD server located at the academic institution used for ML experiments equipped with two NVidia Quadro PRO4000 GPUs.
(2) *hso-compute*: similar to *hso-ml* but lacking any GPUs.
(3) *cp-5G*: a single-socket 32-core Intel server located at a cloud provider used in a 5G testbed.

Table 3 shows that a server consumes 27-33% additional power during bootup confirming the sysadmin's concerns. As a result, our bootup policy lets a server reboot only if the overall rack power consumption is below a fixed threshold. Such a policy depends on factors external to a server.

Table 4 presents three BMC operations labeled as *both-parties-with-policy*. The first operation reads system event logs and is subject to a policy listed as a post-condition. If the cloud provider (CP) initiates this operation, the hosting site operator (the academic institution) requires removing all OEM-specific information (i.e., the results of the operation that start with the prefix "Oem*" are all assigned to null). If, instead, the host site provider initiates this operation, the cloud provider requires filtering out all non-critical log entries (i.e., the returned log entries must have "Severity" field set to "Critical").

The second operation reads information about the server's chassis. In this case, the hosting site operator requires filtering out all security certificates before returning this information to the cloud provider.

The third operation resets the server system and is accompanied by a policy that has pre-conditions involving factors *external to a server*. The host site operator's pre-condition requires that the overall power consumed by the rack is less than 1 Kilowatt before the reset takes place. The cloud provider's pre-condition is around availability concerns – a server can reboot only if sufficient servers are up and available. In our example, a server can reboot if the rack has at least 6 healthy servers running.

4 PRELIMINARY SYSTEM PROTOTYPE

We have implemented the P^2C authorization framework as a standalone *proxy*. To make Redfish API calls to the BMC, both parties must authenticate themselves to the proxy. Neither party possesses BMC credentials; their BMC access always goes through the proxy. The proxy can operate in the cloud or on a on-premise server.

The proxy maintains the parties' credentials, the Redfish Administrator credentials, and the BMC operations it supports. In our current implementation each party is responsible for checking its pre- and post-conditions, if any. Checking these conditions is done using callbacks to interfaces registered by each party and might result in additional (nested) calls to the proxy. The proxy labels calls using distinct operation IDs so that all parties can track different ongoing operations.

The proxy's implementation differentiates the two parties based on who initiates the BMC call. The initiator is referred to as the *executor* whereas the other party is referred to as the *approver*. Note that the cloud provider or the host site operator can play either role.

Figure 2 shows the timeline of a server reboot initiated by the cloud provider. As Table 4 showed, the host site operator allows reboots only if the sum of all server's power consumption is less than 1KW. Performing this pre-condition requires the hosting site operation to perform additional BMC operations that, in turn, need to be approved. If the pre-condition holds, the host site operator approve the reboot and the proxy reboots the server.

Certain combinations of nested operations can lead to an infinite call loop, ultimately causing a *livelock* situation for the system. In such scenarios, the framework implementation will spawn operation calls until it runs out of resources. Figure 3 illustrates a simple example between the cloud provider and the hosting site operator.

Our current prototype is susceptible to such forms of livelock. One straightforward approach to mitigate this problem is by imposing a constraint on the depth of nested operations, limiting them to a fixed threshold (e.g., 10 levels deep). When this threshold is exceeded, the authorization framework can respond by rejecting the operation and asking the parties to review their policies that caused the livelock.

Our proxy is written in Go, we conducted tests with two different setups. First, we deployed the proxy inside a BMC environment emulated in a QEMU virtual environment. This scenario corresponds

Type	Operation	Description	P^2C Category	Policy
Logging	Get EventLog entries	Return System Event Log's LogEntry collection	(HSO and cp) or (CP and hso)	hso.post: LogEntry.Oem* := null cp.post: LogEntry.Severity == "Critical"
Monitoring	Get chassis info	Returns schema of server's chassis	HSO or (CP and hso)	hso.post: Chassis.Certificates := null
Operation	Place system in {ResetState}	Power on, reboot, or shutdown system	(HSO and cp) or (CP and hso)	hso.pre: SUM(External.PDU.Outlets.Power) < 1000 cp.pre: LEN(External.Servers.State == "Healthy") ≥ 6

Table 4: Three examples of BMC operations and their corresponding P^2C authorizations. HSO stands for hosting site operator and CP for cloud provider. When shown in lower-case, this indicates that their P^2C approval is subject to a policy.

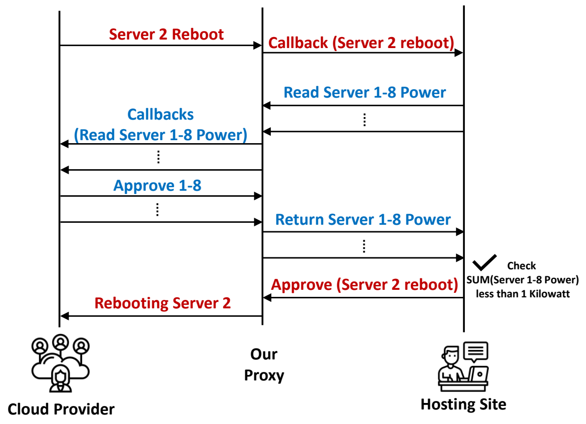

Figure 2: A timeline of nested operations required for a server reboot initiated by the cloud provider.

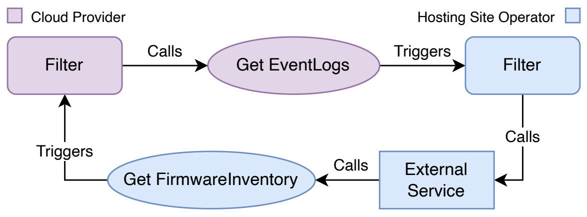

Figure 3: An example of an infinite call loop.

to the management of one single server SKU. Second, we deployed the proxy as a daemon on a management node that controls an 8-server rack.

5 DISCUSSION

In this paper, we argue that the management of MCloud infrastructure requires a collaborative approach between all parties involved. We described a P^2C architecture that enables collaborative control over a server's BMC functionality. This is only a first step in this direction, and additional research is required including:

Proxy Trust Model: Both parties must have confidence that the proxy operates in good faith and enforces each party's policies. We envision two models to establish this trust. One relies on a trusted third-party, while the other relies on an open-source implementation of the proxy coupled with trusted hardware. The trusted hardware can provide hardware-backed software attestations demonstrating that the correct proxy version is in use. The open-source nature of the software enables each party to verify its correct behavior [8].

Proxy Provisioning: The proxy setup must involve a one-time provisioning step during which it is configured with the credentials of the involved parties as well as the Redfish Administrator credentials. We envision a model in which the proxy immediately logs on all BMCs under its control and changes the Redfish Administrator credentials to revoke access to any existing party. This approach ensures that all BMC operations must go through the proxy, preventing any single party from bypassing it unilaterally.

Proxy Recovery: A mature proxy implementation requires the ability to recover from a proxy failure. Our envisioned recovery scheme involves the proxy periodically checkpointing its state, encrypting it, and distributing the encryption key to all stakeholders through a secret sharing scheme [37]. Upon a failure, all stakeholders must cooperate to recover the proxy and restore its state from the checkpoint.

Transactional Support and Scheduling for Atomicity: Some tasks could encompass multiple operations, such as a firmware update followed by system reboot. Parties should be able to use transactions ensuring that either all or none of the operations are executed. One simple approach would pre-approve all operations in the series prior to their execution. Another approach would offer genuine transactional support ensuring operations can be reverted in case a an execution step fails to be approved.

Networking Considerations: This paper focused on server and rack management. However, an equivalent P^2C authorization framework could be applied to managing the MCloud's underlying network fabric.

Multi-Party Control: While our focus in this paper is on two-party infrastructure management, we predict the emergence of more intricate scenarios in the future. For example, hosting site operators might require integration with multiple cloud providers. As control shifts from one-party (P^1) to two-party (P^2) today, we foresee a progression to multi-party (P^n) control in the near future.

Crowdsourcing: Some of the world's largest edge platforms are based on crowdsourcing [32]. A range of third parties, from enterprises to individual owners, provide their spare servers to these edge platforms. These third parties retain control of their servers and have the freedom to restart or withdraw them from the platform at any time. As a result, these platforms are faced with much more churn that traditional cloud platforms. A server's status can frequently change requiring edge services to quickly adapt to these fluctuations. A multi-party control mechanism, such as P^2C, could let the edge provider gain a degree of control over the underlying infrastructure, leading to enhanced stability of the platform.

Fine-Grained Hardware Configuration and Multiple Cloud Providers: Finer-grained control over hardware settings can give a hosting site operator additional flexibility when allocating hardware resources among multiple cloud providers. For example, the operator could host two cloud providers on a single server, and dedicate certain hardware resources and peripherals (e.g., GPUs) to one of the cloud providers only. Also, upcoming CPUs incorporate a chiplet-based architecture [17] that would naturally extend to resource partitioning and allocation among multiple cloud vendors.
Nested P^2C Authorization Frameworks: Hosting site operators often involve multiple independent teams, such as an on-site team and an off-site, more centralized team. There are instances when these teams lack precise coordination in managing their infrastructure. For instance, the off-site team might be comfortable with rebooting a rack of servers, while the on-site team may require additional policies for such an operation. To address this, a hosting site operator can employ its own P^2C layer to coordinate among various internal stakeholders.
Development of a High-Level Policy Language: Currently, we manually defined our policies, and such an approach is error-prone and unscalable. Instead, the development of a policy definition language would allow system administrators to reason about high-level goals and develop policy analysis and verification tools.
Addressing Scalability: We developed our proxy and policy management without having scalability as an explicit goal. For instance, the scenario depicted Figure 2 requires performing eight explicit approvals for each of our eight servers. Such an approach is impractical in large deployments.
Auditability: A mature P^2C authorization framework must include the capability to audit its behavior effectively. Audit logs provide a detailed record of system activities and multi-stakeholder actions. This information is crucial for security monitoring and compliance with regulatory requirements. Auditors can review these logs to ensure that security policies and procedures are being followed.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive feedback. We also thank Mark Dalton Gray for his experienced insights.

REFERENCES

[1] Giles David Arceneaux. *Beyond the Rubicon: Command and Control in Regional Nuclear Powers*. PhD thesis, Syracuse University, 2019.
[2] AWS. AWS Outposts Family. https://aws.amazon.com/outposts/, 2023.
[3] AWS. AWS Outposts rack hardware specs. https://aws.amazon.com/outposts/rack/hardware-specs/, 2023.
[4] AWS. AWS Wavelength. https://aws.amazon.com/wavelength/, 2023.
[5] Paul Barham, Boris Dragovic, Keir Fraser, Steve Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the Art of Virtualization. In *SOSP*, 2003.
[6] Muli Ben-Yehuda, Michael D. Day, Zvi Dubitzky, Michael Factor, Nadav Har'El, Abel Gordon, Athony Luiguori, Orit Wasserman, and Ben-Ami Yassour. The Turtles Project: Design and Implementation of Nested Virtualization. In *OSDI*, 2010.
[7] Matthew Bunn. Preventing Insider Theft: Lessons from the Casino and Pharmaceutical Industries. *Journal of Nuclear Materials Management*, 2013.
[8] Antoine Delignat-Lavaud, Cédric Fournet, Kapil Vaswani, Sylvan Clebsch, Maik Riechert, Manuel Costa, and Mark Russinovich. Why Should I Trust Your Code? *Communications of the ACM (CACM)*, 67, 2014.
[9] Department of Defense. DoD Instruction 5210.65 – Security Standards for Safeguarding DoD Chemical Agents. https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/521065p.pdf, 2020.
[10] Distributed Management Task Force. Redfish Sessions. https://www.dmtf.org/sites/default/files/Redfish_School-Sessions.pdf, 2018.
[11] Distributed Management Task Force. Redfish. https://www.dmtf.org/standards/redfish, 2023.
[12] Yaozu Dong, Xiaowei Yang, Jianhui Li, Guangdeng Liao, Kun Tian, and Haibing Guan. High Performance Network Virtualization with SR-IOV. *Journal of Parallel and Distributed Computing*, 2012.
[13] Federal Financial Institutions Examination Council. Authentication and Access to Financial Institution Services and Systems. https://www.ffiec.gov/guidance/Authentication-and-Access-to-Financial-Institution-Services-and-Systems.pdf, 2021.
[14] Google Cloud. Anthos. https://cloud.google.com/anthos, 2023.
[15] Google Cloud. Bringing partner applications to the edge with Google Clouda. https://cloud.google.com/blog/topics/anthos/anthos-for-telecom-puts-google-cloud-partners-apps-at-the-edge, 2023.
[16] Hewlett Packard Enterprise. HPE Reference Configuration for hybrid cloud mobility with Google Cloud's Anthos on HPE Nimble Storage dHCI. https://www.hpe.com/psnow/doc/a50001096enw.pdf, 2023.
[17] Intel. Fostering a Chiplet Ecosystem for the Future of Moore's Law. https://www.intel.com/content/www/us/en/newsroom/opinion/fostering-chiplet-ecosystem-future-moores-law.html, 2022.
[18] Intel. Introduction to Memory Bandwidth Allocation. intel.com/content/www/us/en/developer/articles/technical/introduction-to-memory-bandwidth-allocation.html, 2022.
[19] Intel. Network and Edge Reference System Architectures - vRAN Setup with FlexRAN Software Quick Start Guide. https://www.intel.com/content/www/us/en/content-details/737687/network-and-edge-reference-system-architectures-vran-setup-with-flexran-software-quick-start-guide.html, 2023.
[20] Intel, Hewlett-Packard, NEC, and Dell. Intelligent Platform Management Interface Specification. https://www.intel.la/content/dam/www/public/us/en/documents/specification-updates/ipmi-intelligent-platform-mgt-interface-spec-2nd-gen-v2-0-spec-update.pdf, 2015.
[21] Panna Kemenes. What is dual control banking & should you use it? https://wise.com/us/blog/dual-control-banking, 2022.
[22] Ammar Latif, Ash Khamas, Sundeep Goswami, Vara Prasad Talari, and Young Jung. Telco Meets AWS Cloud: Deploying DISH's 5G Network in AWS Cloud. https://aws.amazon.com/blogs/industries/telco-meets-aws-cloud-deploying-dishs-5g-network-in-aws-cloud/, 2020.
[23] Robert Love. Kernel Korner: CPU Affinity. *Linux Journal*, 2003.
[24] Cory Mann. Using Dual Control to Help Prevent Payment Fraud. https://www.fnbo.com/insights/commercial-business/2023/using-dual-control-to-help-prevent-payment-fraud, 2023.
[25] Microsoft. Azure Operator Nexus. https://azure.microsoft.com/en-us/products/operator-nexus, 2023.
[26] Microsoft. Azure Stack Edge. https://azure.microsoft.com/en-us/products/azure-stack/edge, 2023.
[27] Microsoft News Center. AT&T to run its mobility network on Microsoft's Azure for Operators cloud, delivering cost-efficient 5G services at scale. https://news.microsoft.com/2021/06/30/att-to-run-its-mobility-network-on-microsofts-azure-for-operators-cloud-delivering-cost-efficient-5g-services-at-scale/, 2021.
[28] Vipin Narang. *Nuclear Strategy in the Modern Era: Regional Powers and International Conflict*. Princeton University Press, 2014.
[29] Robert D. Pedersen. Two-Person Control – A Brief History and Modern Industry Practices. Technical Report SAND2017-7995, Sandia National Laboratories, 2017.
[30] Shaya Potter, Steven M. Bellovin, and Jason Nieh. Two-Person Control Administration: Preventing Administration Faults through Duplication. In *LISA*, 2009.
[31] Michael Richtberg. Google Cloud Anthos Bare Metal Support from Dell Technologies. https://www.dell.com/en-us/blog/google-cloud-anthos-bare-metal-support-from-dell-technologies/, 2023.
[32] Shihao Shen, Yicheng Feng, Megwei Xu, Cheng Zhang, Xiaofei Wang, Wenyu Wang, and Victor C. M. Leung. A Holisting QoS View of Crowdsourced Edge Cloud Platform. In *IWQoS*, 2023.
[33] US Air Force. Nuclear Surety Tamper Control and Detection Programs. https://irp.fas.org/doddir/usaf/afi91-104.pdf, 2013.
[34] David West, Chandrakandh Mouleeswaran, Xiaotong Jiang, and Markesha Parker. Reference Architecture for Google Anthos with Lenovo ThinkAgile VX. https://lenovopress.lenovo.com/lp1215.pdf, 2023.
[35] Andrew Whitaker, Marianne Shaw, and Steven D. Gribble. Scale and Performance in the Denali Isolation Kernel. In *OSDI*, 2002.
[36] Wikipedia. Cloud Computing. https://en.wikipedia.org/wiki/Cloud_computing, 2024.
[37] Wikipedia. Secret Sharing. https://en.wikipedia.org/wiki/Secret_sharing, 2024.
[38] Ka Kit Wong. Running Google Anthos on VMware Cloud Foundation. https://core.vmware.com/resource/running-google-anthos-vmware-cloud-foundation, 2023.
[39] Mengwei Xu, Zhe Fu, Xiao Ma, Li Zhang, Yanan Li, Feng Qian, Shangguang Wang, Ke Li, Jingyu Yang, and Xuanzhe Liu. From Cloud to Edge: A First Look at Public Edge Platforms. In *IMC*, 2021.