NORTHWESTERN
UNIVERSITY

Computer Science Department

Technical Report
Number: NU-CS-2023-09

April, 2023

A Case for a User-centered Distributed Privacy Backplane for the Internet of Things

John Lange, Peter Dinda, Robert Dick,
Friedrich Doku, Elena Fabian, Nick Gordon, Peizhi Liu,
Michael Polinski, Madhav Suresh, Carson Surmeier, Nick Wanninger

Abstract

Ubiquitous sensing in human spaces is rapidly becoming a fact of life. Large collections of IoT devices feed a vast sensing infrastructure that records and analyzes personal and private information. Maintaining privacy in the future without hamstringing the utility of IoT-derived data is a key practical and research challenge. In contrast to one-size-fits-all legal privacy frameworks like GDPR and CCPA, we envision a new legal-technical framework called the Privacy Backplane that enables individuals to specify and control the policies governing how information gathered about them is accessed, used, and stored. The legal framework mandates that IoT environments negotiate clear policies with individuals about the how data collected by IoT devices about them can be used, and then enforces those policies. The technical framework implements such negotiation and enforcement, and certifies that the policies are followed. In effect, this can be viewed as the inverse of classic DRM: individuals are the content producers, and have knowledge of and control over the policies applied to their information. This paper makes the case for the Privacy Backplane and argues that it can be implemented on the substrate

of modern, hardware-based trusted computing platforms, using a full-stack approach that spans from sensors to overlay networks. We describe the challenges of designing and implementing such a system. These include defining relevant threat models, near-sensor/on-sensor inference, trust bootstrapping, designing useful and implementable policy models, providing operating systems support, overlay networking, query mapping and scheduling, and supporting real-time applications. We also comment on our first steps in designing and evaluating the Privacy Backplane.

Keywords

privacy, sensors, embedded systems, IoT

A Case for a User-centered Distributed Privacy Backplane for the Internet of Things

John Lange* Peter Dinda† Robert Dick‡
Friedrich Doku§ Elena Fabian† Nick Gordon§ Peizhi Liu†
Michael Polinski† Madhav Suresh† Carson Surmeier† Nick Wanninger†

*University of Pittsburgh and Oak Ridge National Labs
†Northwestern University
‡University of Michigan
§University of Pittsburgh

***Abstract*—Ubiquitous sensing in human spaces is rapidly becoming a fact of life. Large collections of IoT devices feed a vast sensing infrastructure that records and analyzes personal and private information. Maintaining privacy in the future without hamstringing the utility of IoT-derived data is a key practical and research challenge. In contrast to one-size-fits-all legal privacy frameworks like GDPR and CCPA, we envision a new legal-technical framework called the Privacy Backplane that enables individuals to specify and control the policies governing how information gathered about them is accessed, used, and stored. The legal framework mandates that IoT environments negotiate clear policies with individuals about the how data collected by IoT devices about them can be used, and then enforces those policies. The technical framework implements such negotiation and enforcement, and certifies that the policies are followed. In effect, this can be viewed as the inverse of classic DRM: individuals are the content producers, and have knowledge of and control over the policies applied to their information. This paper makes the case for the Privacy Backplane and argues that it can be implemented on the substrate of modern, hardware-based trusted computing platforms, using a full-stack approach that spans from sensors to overlay networks. We describe the challenges of designing and implementing such a system. These include defining relevant threat models, near-sensor/on-sensor inference, trust bootstrapping, designing useful and implementable policy models, providing operating systems support, overlay networking, query mapping and scheduling, and supporting real-time applications. We also comment on our first steps in designing and evaluating the Privacy Backplane.**

***Index Terms*—privacy, sensors, embedded systems, IoT**

I. Introduction

The right to privacy and control over one's personal information is a central issue of our time. Increasingly cheap, powerful, and long-lived sensors are being introduced everywhere, and sensor fusion and machine learning are extracting ever-more actionable information from them. At the same time, legal protections and government-enforced regulations are emerging in the form of the EU's GDPR [1] and California's CCPA [2], along with growing calls for more government intervention. However, the ultimate effectiveness of such laws depends on the existence of infrastructure that makes adhering to them and enforcing them practical. Given current regulatory and technological trends, many are worried that we are heading toward either the death of privacy or a top-down notion of privacy that is disconnected from the realities of the current technical landscape. We instead argue for a future where *individual* control over *personal* data privacy is ensured by a legal framework that is able to leverage technological capabilities inside computing infrastructures to *guarantee* regulatory compliance. More specifically, we claim that it is feasible and practical to enable individual privacy and personalized control over data collected by a distributed sensing infrastructure in a physical environment.

Instrumented environments are becoming increasingly prevalent and will likely be ubiquitous, as our physical infrastructure further integrates computing and sensing capabilities and IoT technologies. In particular, we focus on future environments, such as retail contexts, healthcare facilities, and even public spaces, where the data collection and privacy interests of different actors may come into conflict. In these environments, private companies and public institutions will have vested interests in collecting sensed data inside the physical environments they manage to optimize space use and inform business decisions. On the other side are the individuals whose data will be collected, and who will have personal preferences or requirements regarding *which* data are collected and *how* they are used.

Central to our vision is an intelligent system-level infrastructure that negotiates data collection and access policies on behalf of *users* and the *operators* of the

This project was supported by the United States National Science Foundation via grants CNS-2211507, CNS-2211508, and CNS-2211509.

physical environment. This infrastructure will negotiate, compile, and deploy policies that control data collection and use. It will be pervasive and incorporate infrastructural control capabilities to manage actual sensor nodes, interfaces to enable policy specification, external interfaces for data queries and analysis that honor policy constraints, and finally a data management infrastructure to persistently associate policies with data and derivative information. The infrastructure and the concepts embedded in it have the goal of enforcing user requirements for privacy and data access within a large-scale sensing environment in a secure and trusted way that ensures policy and functionality requirements are met. We claim that such an infrastructure can be implemented on the substrate of modern, hardware-based trusted computing platforms, using a full-stack approach that spans from sensors to overlay networks. We are now working to design and implement a proof-of-concept, the Privacy Backplane.

The regulatory element of our model is the legal requirement that a system like the Privacy Backplane be ubiquitous in systems that sense data about individual users, e.g., IoT devices, and everywhere that data and derived information from such devices flows. That is, the regulations require the Privacy Backplane or something offering similar privacy protections be used, and the Privacy Backplane operates on behalf of both users and operators to negotiate shared privacy policies and then enforce them across all uses of the data collected by the sensors. The regulatory element also imposes liability when devices falsely claim compliance, or when specific pieces of software (like query language operators) falsely claim to have certain privacy properties. In effect, our model can be viewed as the inverse of classic digital rights management (DRM): individuals are the content producers, and have knowledge of and control over the policies applied to their information. On the other hand, operators can use of this information within the limits of user policies. Similar to classic DRM, hardware and software are expected to obey negotiated policies. The most straightforward way to achieve both the users' and the operators' goals is to join the Privacy Backplane, or a system like it.

We are designing and implementing the Privacy Backplane as a distributed trusted execution environment enabling the secure storage of private user data and enforcing policy-based access controls provided by individual or collective stakeholders. The Privacy Backplane will serve as a nexus of control that enables distributed policy enforcement over collected (sensed) data, while also serving as an integration hub for users, the sensing infrastructure, and data accessors. The Privacy Backplane will leverage recent advances in trusted hardware capabilities that allow secure deployment of sensitive data access operations to hardware managed by untrusted 3rd parties. In our emerging system, each hardware component and device used for sensor-based data collection will be required (possibly by a regulatory body) to support some form of TEE capabilities. It will be the responsibility of the Privacy Backplane to configure and coordinate these devices based on collected policy requirements. It must be extensible, allowing dynamically changing queries and query operators. It must also operate in real time, as users navigate around a space in a natural, unencumbered manner, and as operators query data about users.

The overall goal of Privacy Backplane effort is to determine whether the technical model described above is feasible and secure at the kinds of scales needed to be useful. We argue in this paper that this is likely, and describe our initial results in creating a Privacy Backplane system. Our contributions follow.

- We introduce and motivate a joint regulatory and technical model, instantiated in Privacy Backplane, that can provide for individual privacy policies and control over personal data in the IoT and similar distributed environments.
- We give a specification of privacy policy and data processing within the Privacy Backplane.
- We show an architecture for implementing the specification as a distributed system of hardware-based trusted execution environments.
- We lay out challenges and opportunities in such a system that we have encountered thus far in our implementation effort.

Related work: Developing effective mechanisms to protect personal privacy has become a priority in many societies with multiple approaches under exploration.

Regulatory-based approaches The most prominent efforts to provide privacy protections have taken the form of legal and regulatory frameworks that seek to limit both the collection and use of personal data. The best known example is the European GDPR [1] regulation, which places restrictions on how personal data are collected, used, and transferred. Similarly in the U.S., California has begun enforcing the CCPA [2], which seeks to provide transparency and control to individuals over the collection and use of personal data. In addition, Texas and Illinois have imposed regulatory frameworks covering the collection and use of biometric data [3], with Texas having significant restrictions on facial recognition [4]. While these regulations enshrine privacy protections into law, there are questions as to their effectiveness and enforceability [5]. The fundamental limitation is that these approaches do not provide physical, technological protections that make compliance straight-forward and instead rely heavily on self-reported

compliance documentation and the threat of investigatory actions.

Technical approaches The research community has seen a number of proposals that seek to provide privacy protections in IoT environments [6]–[11]. Of note is TIPPERS [12]–[15], which focuses on privacy management in IoT spaces through a wide range of approaches. There has not yet been a holistically designed approach that ensures *trusted* operation at each layer of the system stack, all the way down to data capture by embedded sensors: this is our goal. Creating a fully trusted system that is performant and scales by leveraging recent advances in trusted hardware platforms presents a number of novel and unique architectural challenges that impact the overall design of the system. Our vision is to *use regulation to support a technical system* capable of negotiating and enforcing policies acceptable to operators and users, as opposed to specific policies.

II. Model

To make implementing Privacy Backplane tractable, we propose formal models of policy specification, reconciliation, and data transformation. This section describes our *current* models. The models need to provide representational power for both individual users and venues, and enable processing of sensor data within the Privacy Backplane. These models must be realizable in an online, scalable, distributed system, with responsiveness appropriate for mobile users. Figure 1 shows the big picture of our initial model.

A. Policy representation

Attributes: An attribute is a string chosen by convention (perhaps in a marketplace) that captures a human-understood and human-audited privacy property.

User scope: The starting point of the individual user's privacy interests is modeled as a volume in space. We assume effective and efficient localization (we will later explain why this will be practical for our application). U_i is the scope of the ith user, i.e., the scope within which sensor data about the user is subject to the user's privacy policy. If a sensor can detect anything within the user's scope, its outputs are subject to the user's policy. Scope may or may not be tied to user identity. The specific volume allowed depends on the law, but will generally be the user's immediate vicinity. User scopes move with users in real time.

User policy: A user policy is three sets of attributes: R, the "Require Set", is the set of attributes that must hold for the policy to be in force. F, the "Forbid Set", is the set of attributes that must not hold for the policy to be in force. D, the "Don't Care Set", is the set of attributes that may or may not hold. These may be implicit. A user policy is evaluated in the context of a user scope. $L_i = (R_i, F_i, D_i)$ is the user policy associated with the ith user scope, U_i. User policies will change, e.g., when user preferences change or upon entering new venues.

Sensor scope: A sensor has a special status in our model; it is associated with a volume in space over which it meaningfully collects data. "Meaningfully" is ultimately defined by law, but the typical superlinear dropoff of signal amplitude with distance provides a natural range threshold. The scope of the jth sensor is S_j. Sensor scope can change in real time.

B. Data processing / Query model

Sensor data enters Privacy Backplane immediately after capture. The backplane allows for data processing across all sensors and users, but no result based on information from a given user scope is allowed to leave the backplane unless the processing that produced it complies with the matching user policies. Data processing takes the form of queries that can occur at any time, although our expectation is that these are likely to be rather static, though their results may change.

Directed acyclic graph (DAG): To make this goal tractable at scale and in the presence of change, we restrict the query data processing model. A key research challenge is determining the right balance between model expressiveness and ease of implementation. Tentatively, a query is a DAG in which there are three kinds of nodes: *sensor nodes*, which only produce outputs; *transform nodes*, which accept inputs and produce outputs; and *exit nodes*, which have a single input and represent movement of data out of Privacy Backplane. Edges represent data transfers. Multiple DAGs may exist simultaneously. Conceptually, each DAG represents computation over privacy-sensitive user information, captured by sensors, that an operator would like to perform. However, a DAG may also represent a much larger-scale computation over a wide area.

The graphs are acyclic: user information captured by sensors flows from sensor nodes to exit nodes. Computation on the DAG itself is not Turing-complete, although transform nodes may be (but time-limited). This facilitates reasoning about the privacy aspects of the computation. User scope information flows through the DAG alongside raw and transformed sensor data. Each sensor coveys its user visibility set into the system. We can thus determine which user scopes contributed to each output.

Transform properties: A transform node may have several privacy-relevant properties: A *segregating* transform partitions data by user scope. For example, detecting users loitering around an exit. An *aggregating* transform fuses data by user scope. For example, combining video and audio data for each user. The output of an aggregating transform may also compute higher-level prop-

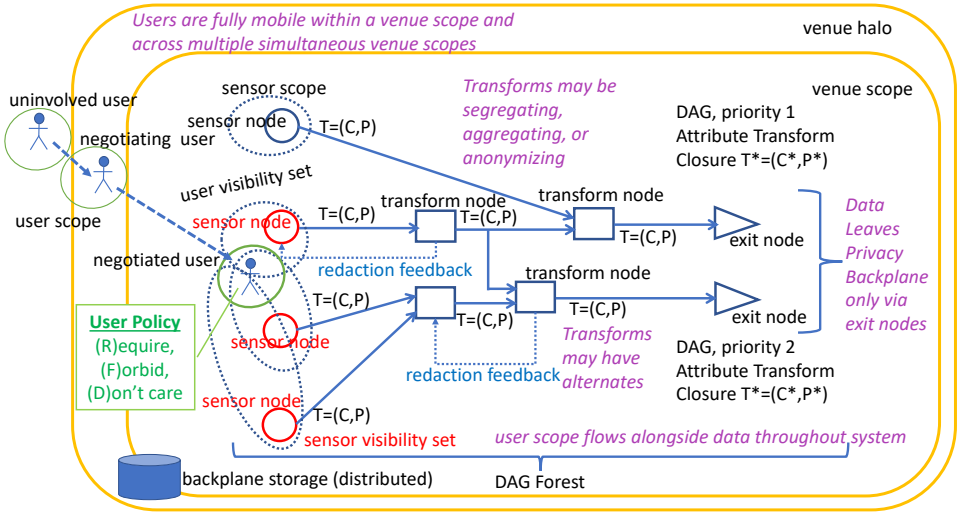

Fig. 1: Privacy Backplane's current policy representation, reconciliation, and data processing model.

erties, such as emotional state or anomalous behavior. An *anonymizing* transform removes user scopes. For example, counting the number of individuals near the exit with anomalous behavior.

Redaction feedback: A node in the graph may also provide feedback to an earlier node with the primary purpose of assisting resource-limited nodes in determining which information should be locally redacted. This capability must be used with care if used to inform sensors with such limited hardware capabilities they cannot fully join the privacy backplane; it risks revealing information about the contents of a particular privacy policy to the sensor, although that information would only be associated with a volume, not a particular user.

Venue scope and halo: Each DAG has an associated volume in space. This scope is the union of the sensor scopes. The venue halo extends beyond the venue scope and represents the liminal space in which policy reconciliation occurs before the venue is entered.

Query scope: Conceptually, a DAG is a streaming query with agreement among all participants. While all data meeting policy constraints may flow from the exit node, venues will generally be interested in only a tiny subset. Query scope captures this concept. When a query asserts a condition, e.g., "crime occurred near redacted image", to enable unredaction, the signature will entered in a global log that can be maintained using one of several existing technologies such as Blockchain.

Attribute transform: Node outputs have the following sets of attributes: C is the "Consume Set": the attributes that will no longer be true at the output. P is the "Produce Set": the attributes that will be true at the output. Node k's lth output has attribute transform $T_{k,l} = (C_{k,i}, P_{k,l})$, which captures how the data transformation of the node manipulates privacy attributes.

Attribute transform closure: Note that closures can be computed by starting at any exit node k and working recursively through the DAG until reaching the sensor nodes. $T\star_k = (C\star_k, P\star_k)$ is such a closure.

DAG forest: To facilitate policy negotiation, multiple DAGs may be associated with a venue scope and volume, with each DAG rooted in the same sensor nodes. Each DAG may have a different attribute transform closure, and thus a different balance between privacy policy compatibility for users and information extraction value for operators. The DAGs may be given priorities, so that user negotiations can progressively move from more to less revealing versions of a query.

Node variants: A DAG node might have several variants, each with different C and P sets. Node variants have a priority order, so a venue can negotiate starting from the most revealing version of the node.

Backplane storage: Data may persist in the backplane. Storage facilities are provided, but all data have time-to-live values determined by policy reconciliation.

C. Policy negotiation and processing

The user and the venue both advertise their scopes. Since these are spatial volumes, no use of identity is needed. Negotiation begins upon entering a venue halo.

Fast check: Negotiation compares the attribute transform closure $T\star_k = (C\star_k, P\star_k)$ of an exit node with the user's policy $L_i = (R_i, F_i, D_i)$. The negotiation succeeds if $R_i \cap C\star_k = \emptyset$ and $F_i \cap P\star_k = \emptyset$ and $D_i \supseteq (P \star_k - R_i)$. The first clause assures that no attribute required by the policy is consumed by the DAG. The second assures that the DAG produces no forbidden attribute. The third assures that all attributes produced by the DAG are explicitly allowed by user policies.

Back-off model: The fast check is applied to the DAGs in descending order of priority, resulting in an agreement on the highest priority DAG for which the check succeeds. This balances the needs of the user (encoded

in the policy) and the venue (encoded in the priorities).
DAG transformation: If no DAG succeeds, we re-iterate through DAGs in priority order. This time, we traverse the DAG from the exit node to find the nodes that result in fast check failure. For example, if $C\star_k$ contains some attribute from R_i, we isolate the nodes that consume that attribute. We then consider node alternates, in descending order of priority, seeking one without conflict.
Negotiation failure: If negotiation does not succeed by the time the user scope intersects the venue scope, the user is warned. This implies the user agrees to the default policy or is trespassing.
Negotiation success, normal operation, teardown: If negotiation succeeds by the time the user scope intersects the venue scope, Privacy Backplane assigns a one-time key to the user scope. This represents a successful connection to the backplane, and will be used to associate information about the user scope throughout the system. When the user scope leaves the venue scope, the key is destroyed. In any case, if the user scope reenters the venue scope, from the user's perspective, the negotiation process occurs again.

III. Architecture

The architecture of Privacy Backplane is that of a *distributed trusted execution environment (Distributed TEE, or DTEE)* that extends from IoT sensors and the edge to processing and storage infrastructure in the cloud. Each DTEE component includes hardware and software TEE features to handle information and code locally in a secure manner. The DTEE components then interoperate as a distributed system that ensures that data and information privacy, security, and integrity are maintained from the time information is captured to when it either leaves the DTEE with all negotiated policies having been applied, or until deletion from the DTEE. Conceptually, the DTEE includes a distributed policy engine that performs policy negotiation between users and venues as described in §II, as well as a distributed query engine responsible for executing the DAGs described in that section. The distributed query engine controls the flow of sensed information, and its derivatives, based on the negotiated policy. The DTEE includes a join/leave protocol for hardware (sensors, servers, etc), users, and queries. Finally, the DTEE runs additional distributed processes that continuously measure and verify each system component in order to maintain trust of the entire system and ensure that policies are honored as long as the relevant data and information persist.

Figure 2 illustrates the architecture at a high-level. A fundamental building block of our system is an element, the Privacy Backplane Daemon (PBD), that is engineered to be portable to a broad range of heterogeneous CPU

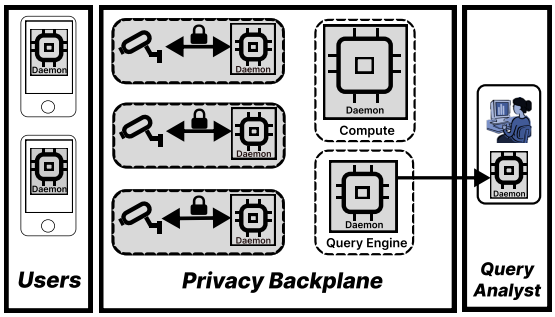

Fig. 2: Privacy Backplane architecture

and platform architectures, from the sensors on up. Every node runs a PBD. The PBDs integrate through a common protocol that routes across a dynamically constructed trusted overlay network (the PBTON). To the greatest extent possible there is a separation of concerns in the design of the PBD and PBTON, the various engines they host, and the specific mechanisms used to establish trustworthiness, but these are generally built on top of hardware-based trusted execution and remote attestation. In order to maintain the trust of the full PB distributed system, any PBD must be able to verify the trustworthiness of any other PBD through a distributed attestation mechanism that provides both direct and transitive attestation as well as preemptive invalidation.[1]

To ensure end–to–end privacy guarantees, each sensor is ideally paired directly to a PBD running on a trusted platform. The goal is to ensure that all collected data are directly routed into the DTEE and are inaccessible from untrusted code, even if it is running on the sensor platform itself. Each sensor executes a local policy engine enabling direct control over sensor operation by the PBD. This allows the Privacy Backplane to exploit compute capabilities at the point of data collection to honor policies with minimal opportunities for data leakage.

A. Node-level

The Privacy Backplane is designed around the pervasive use of hardware TEEs on each node in the infrastructure. TEEs provide secure computational and data access capabilities on untrusted platforms through the use of special instruction set architecture extensions that enable verifiably secure execution environments. TEEs allow the deployment of trusted software components on systems managed by untrusted 3rd parties and allow the execution of that software to be verified through attestation mechanisms. TEE capabilities have become commonplace and are now available on a wide range of

[1]While the user's device runs a PBD, as shown, this is configured to only provide basic, user-centric operation: user volume/location/policy association, halo determination, policy negotiation, etc. No other code runs in the user's device, and no user identity leaks from it.

system platforms from server-based systems to edge devices [16] and sensor platforms [17], [18]. TEEs provide three features required to provide a secure distributed architecture: (1) *Isolated Execution* of the runtime from untrusted components including the OS and external devices, (2) *Sealed Storage* to securely persist secrets onto local disk in a manner that is not accessible outside the protected runtime executing on the particular CPU, and (3) *Remote Attestation* to validate the identity of a remote trusted runtime and its underlying platform. These components provide a solid foundation for developing tamper-resistant, non-bypassable, and verifiable trusted monitors.

We intend to leverage the full range of hardware-specific TEE capabilities across different platforms and architectures, to provide the PBD with a common abstraction it can build on in a portable way. Note that the PBD itself needs to be able to run specialized code in order to implement DAG operators (the transform nodes of §II). Here we plan to use a lightweight Web Assembly (WASM) [19] runtime environment in the PBD to provide such code with a common service-based framework that can in turn be used by the DTEE's distributed query engine's planner to map a query DAG onto PBDs running on disparate hardware.[2]

Supporting a constantly available PBD on each node is a requirement. While some TEE architectures are amenable to this use case, others will require either modifications or workarounds. Modern TEE architectures cover a broad range of execution models from self-contained application libraries (e.g., Intel SGX) to full-system software stacks containing both an OS and userspace environment (e.g., ARM TrustZone). While we separate concerns in our design to support multiple architectures, we expect that each will provide specialized features that map better to certain uses. For instance, we expect SGX server nodes to handle much of the backend processing while ARM TrustZone based platforms will make up the edge nodes that interface directly with the sensors due to their ability to support trusted partitioning of I/O devices as well as full trusted OS environments complete with device drivers. Therefore, while PBDs will be deployable throughout the infrastructure with a shared abstraction, each specific node will offer additional capabilities for the PBD to leverage. In the case of ARM, this will entail a fully custom system software stack complete with trusted device drivers and I/O channels that enable the backplane to extend the trust boundary directly into the sensors themselves. This system software stack will be based on a lightweight kernel [20] architecture due to its ability to provide a mostly compatible Linux ABI with a small TCB.

[2]It is important to understand that WASM is a fully-supported target architecture for the widely-used LLVM compiler framework. Consequently, it is thus possible to incorporate arbitrary code, which a PBD will nonetheless be able to run in a sandboxed environment. In our model, a transform node (DAG operator) must provide an attribute transform. This is bound with the code by the code provider, and thus if the asserted attribute transform is incorrect, there is a legal recourse against the code provider.

B. Distributed

The DTEE is based on the trusted overlay network (the PBTON) dynamically established between individual node-local TEEs to manage the system. The PBTON supports encrypted and trusted communication channels between local TEEs via TEE attestation. Because each node executes a common runtime environment (the PBD), links in the overlay communicate using a common protocol. This allows any node to establish a trusted overlay connection to any other node, be it a local sensor node or a cloud-based server. The PBTON topology thus encodes the current state of the web of trust and provides persistent trust across the distributed system. The fundamental responsibility of the DTEE is managing the PBTON topology to ensure the trustworthiness of the environment while also supporting the processing and communication requirements of the distributed policy engine, distributed query engine, and other DTEE elements. This coordination requires both the ability to dynamically establish trust between any two nodes (PBDs) as well as a mechanism for quickly invalidating node trust across the overlay network. Furthermore, it must accomplish this efficiently enough to meet real-time constraints imposed by sensor data rates, policy negotiations, and query execution.

In addition to providing real-time privacy controls, the Privacy Backplane must also maintain policy restrictions for the full duration of the data and derivative information lifecycle. After collection, data may persist inside the DTEE for future use, with its lifetime dependent on negotiated policies and system requirements. As a result, policies must be permanently associated with data as long as they persist. Persistent data will be stored inside a sealed environment provided by a TEE, either at the edge or in the cloud. Whenever it returns to motion, the policy will once again apply.

Accessing and exporting data can only occur via a query DAG initiated on a trusted gateway interface and executed by the distributed query engine, which will apply the negotiated policy as described in §II. On entering a venue, the user's device will negotiate a policy given the current query DAGs. Any new query DAGs during the user's dwell time in the venue, or executed later on stored information, must conform to the same policy.

C. Threat model

Our primary security objective is to ensure that all data collection and access operations take place within the boundaries set by stakeholders' specified and deployed policies. We assume an adversary that has complete access to the sensing and data processing/storage infrastructure, including OS and hypervisor components. The attacker may also tamper with, delete, reorder, or replay network traffic that flows between components of the system. In our case, this attacker may be the space (venue) owner, cloud service providers used by the space owner, or individuals within the space. To achieve our goals, we focus on ensuring the integrity of individuals' policies, the correctness of bindings between policies and individuals' sensed data, and the confidentiality of sensed data. Attacks on availability are not considered.

We assume that TEE hardware and support code is correctly implemented, and free of any security-relevant vulnerabilities. In addition, we assume that the TEE attestation, sealing, and memory protection features of all hardware components function according to their specifications (i.e., Intel SGX or ARM TrustZone). We do not explicitly consider side-channel attacks or implementation flaws [21]–[23]; addressing these issues is orthogonal to our core problem of enforcing policy-based controls throughout the data lifecycle, and is an active area of research within the community [24], [25].

IV. Challenges and early results

A. Transducer leakage

Sensors are ultimately based on transducers, and transducers will generally not support TEEs, but rather present an electrical interface that, after digitization, ties to a device driver. Hijacking the electrical interface would impose a similar cost on attackers to deploying their own sensors. Therefore, attackers would benefit only when the electrical signals or derived information escape the physical vicinities of sensors. As a result, the only requirement for near-edge sensing and analysis systems integrated into Privacy Backplane is that they do not leak sensed or derived information from the local environment, although it may be transmitted to a TEE's device driver.

This requirement allows for electrical connections between sensors and TEE-enabled processors, given precautions against side channel attacks, e.g., monitoring electromagnetic radiation from communication lines. It also admits (transitive) connections to hardware accelerators and processors that do not support TEEs, provided that they are capable of communicating only among themselves and TEE-enabled processors. This implies that near-sensor analysis hardware can be easily integrated, perhaps with some communication channels disabled. For example, a non-TEE chip with built-in wireless BTLE connectivity might have that connectivity disabled in favor of a wired SPI interface to the processor.

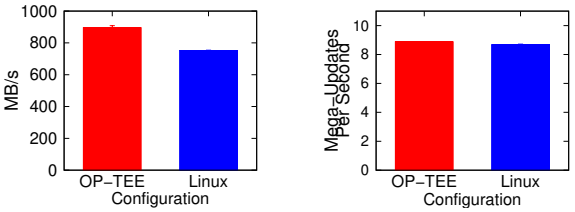

Fig. 3: Memory-bound performance of ARM TrustZone.

B. TEE overheads and early redaction

If hardware TEEs impose a large computational overhead, this may limit their use. We measured the performance of ARM TrustZone using the OP-TEE [26] environment on the RPI 3 and the Pine64 platforms. Figure 3 shows the overhead compared to native performance of two microbenchmarks (STREAM, RandomAccess) on Pine64. As the figure shows, by this measure the overhead is negligible. These are memory-focused benchmarks and TrustZone does not include memory encryption, unlike SGX. We have also measured the overhead, on RPI 3, of training and inference by fully connected neural networks of different sizes running in TrustZone under OP-TEE. This is a computationally intensive benchmark, and we found an overhead of 10%-15% provided the model fits into the OP-TEE trusted environment. We further on define "fits" in the next section.

These results are particularly important for sensor/edge nodes, which may well be less computationally capable to begin with, and to which we would like to push policy-based redaction (redaction feedback in §II), if possible. More generally, a good system design will handle policy enforcement as close to the sensors as possible to minimize the effects of downstream problems on privacy.

The overhead of inference required to associate sensed data with user policies might potentially be high if a naïve implementation were used. However, our model's use of spatial volumes to indicate policy domains, helps make it tractable. Users will generally use personal devices such as smartphones to interact with Privacy Backplane, e.g., as their representative, to learn about policy conflicts, and/or change their policies. These devices will be capable of providing locations associated with policies to Privacy Backplane. Consider video analysis: an application with potentially high overhead for associating data with policies. For this application, a static (and computationally inexpensive) mapping from

spatial regions to camera pixels is sufficient for redaction. Even mobile cameras can use a similar, geometric, approach provided that they measure changes to camera orientation and position. In our experience, inertial measurement units can do this at high speed, low cost, and low computational overhead. Accumulated error can be remedied with (infrequent) automated camera-based calibration. Similarly, in audio sensing applications, location (distance) and noise floor aware methods may be used to associate data with policies. Given policy volume locations, efficient methods of associating volumes with gathered data exist for most sensing modes.

These above methods of associating data with policies may be somewhat imprecise, e.g., due to imperfect localization. There may therefore be benefits to using inference on the captured data for redaction. This could either be done near the sensor, requiring for example near-sensor face recognition, or deeper into the cloud, with communication of redaction regions back to sensing devices. Even if initial associations between visible objects and user policies require cloud analysis, resource-constrained edge devices might subsequently use (hardware-accelerated) object tracking to continue local tracking, policy assignment, and redaction. Although such analysis-intensive methods would not generally be required to support Privacy Backplane, they might improve precision and their computational costs would be similar to, and likely able to reuse computation from, the applications already running on system, e.g., video analysis. Recent innovations in algorithmic efficiency [27] and hardware efficiency [28] that can often reduce inference overhead by an order of magnitude with little impact on accuracy may support near-edge inference for policy assignment and data redaction.

C. Limitations of TEE OSes

Memory: Current TEE OSes appear to place significant limits on the amount of memory accessible within the trusted environment, even when there is no hardware limitation. For example, in OP-TEE on TrustZone, the default available memory is 32 MB with only 30 MB available for use by trusted applications (TAs) [29], likely due to the TLB-level implementation. This imposes design constraints on memory intensive applications, such as neural networks. Using object detection as an example, YOLO, YOLOv2, and TinyYOLO require 753 MB, 193 MB, 60.5 MB. Existing responses include incrementally loading neural network parameters from encrypted store on the untrusted side [30], restructuring the neural network to match the constraints [31], or making the neural network more parsimonious for specific domains [27].

RPC: Extant TEE OSes heavily revolve around an RPC model between the trusted and untrusted "sides". The trusted side itself has minimal services. Most designers assume that only a small amount of functionality is in the trusted side, and thus there is minimal support for long-running, resource rich, and service rich applications there. But that is precisely what a PBD requires.

Devices: Current TEE OS designs delegate interaction with devices, such as sensors, to the untrusted side, and make it difficult to write device drivers on the trusted side, especially for device-mapped memory and MMIO. However, TrustZone relies on the system MMU to enforce the boundary between secure and insecure memory, meaning it is possible to directly and exclusively assign devices to trusted applications.

Our response: Given these limitations, a natural question to ask is whether we can simply design and implement a TEE OS that is geared specifically to Privacy Backplane and has no artificial memory limitations, supports long running code, such as a PBD, on the trusted side without RPC, and facilitates the development of device drivers for the trusted side alone. That is what we are now doing, building on top of lightweight kernels, as described earlier. One advantage we have is that for many nodes, the Privacy Backplane needs *only* the trusted side.

D. Attestation and overlay communication

Ensuring trust in a large-scale distributed system presents a number of challenges beyond traditional distributed systems. Because all communication is bootstrapped using attestation mechanisms, a common root of trust must provide attestation between any pair of nodes, and these processes must support a large-scale, dynamic, heterogeneous distributed environment. This requires that attestation processes be scalable, lightweight enough to support computationally and energy constrained devices, and responsive to invalidation events when trust in nodes is lost. The attestation mechanisms must also be resilient to node failures that result in overlay network partitions, because all inter-node communication relies on attestation.

Addressing these challenges requires an intelligent attestation framework that dynamically modifies the web of trust in the backplane's DTEE. We envision a collection of attestation mechanisms that is selectively deployed based on the state of the backplane's overlay and the profiles of nodes. For instance, mutual trust between nodes enables a transitive trust model: if two nodes have established mutual trust with a 3rd node, then the 3rd node may provide indirect attestation for the pair. We will employ internal centralized and external attestation mechanisms to support a variety of environments.

The overlay network should also provide mechanisms for fault containment and isolation. If a node's trust is lost (e.g., a TEE attack vector is identified), the

node must be isolated from the rest of the backplane. This requires rapid dissemination of trust invalidation signals and poisoned overlay links, and requires rapid establishment of new overlay routes around the untrusted node.

E. Policy negotiation and enforcement

We are co-designing our policy negotiation and enforcement model alongside our architecture (§III), starting with the model described in §II. The overall goal is to develop a formal model that is capable of capturing real user privacy concepts, avoids the notion of identity, and yet is realizable in a scalable system providing acceptable responsiveness for the users. A number of challenges specific to policy arise.

Balancing expressiveness and tractability: Our current model intentionally simplifies user policies to three sets of attributes. This design makes policy negotiation tractable on any machine in the Privacy Backplane that has access to the DAGs and their attribute transform closures. In effect, this is a distributed directory service, but with the need to perform lookup on the timescale of human mobility. In other words, the current model errs on the side of being tractable in a system.

Is this model sufficient to capture the kinds of privacy policies that users may want to express? We have found that there is surprisingly little literature in which the language for privacy policy expression is formal, and hence amenable to analysis. Focusing on expressiveness makes it extremely easy to produce a Turing complete language, likely making implementation in a distributed system with time constraints infeasible. We argue for using the simplest adequately expressive, computationally tractable language. We are in early stages of designing and implementing a user study to capture policy concepts from participants via free association as they shop in a venue. We hope that this study will provide a data pool of privacy terms, as understood by ordinary users, as well as logical constructs derived from the informal vernacular they will use. The privacy terms will form the attribute lexicon of our system, while the logical constructs will be used to determine what specific forms of logic, hopefully a limited set, must be added to our initial design.

Localization: A key aspect of our current model is the idea of a user, sensor, and venue being defined as a scope within the physical world, namely a spatial volume with a location. The ability to determine the locations associated with visual, audio, and other sensor data provides a foundation for the volume-based specification of privacy policies. Recall that all policies are (perhaps indirectly) associated with precise spatial volumes. As a result, reporting the (time-varying) location associated with a particular privacy policy is a necessary precondition for honoring a policy or being warned when it comes into irreconcilable conflict with another (spatial) policy. This produces a natural incentive for participants to accurately specify the spatial volumes to which their policies apply. Falsifying one's location might cause one's policy to be violated. Keep in mind here that the locations are associated with policies, not actual personal identity, so they are mostly privacy neutral.

Malicious users: Malicious sensed users might attempt to specify unreasonably large volumes for their policies in an attempt to make it more difficult to resolve policy conflicts. However, the worst-case scenario is irreconcilable policies, in which case the malicious user would be informed that the policy will not be honored. This is a natural implication of sensors being associated with physical locations owned by the same organizations as the sensors. Sensed users cannot specify policies requiring data to be shared. They can only forbid sharing. Therefore, it is impossible for user policies to come into conflict with each other. Finally, were malicious venue owners to apply their policy requirements to physical volumes exceeding their property, this would be easily detectable and addressable through legal means.

Incentivization: Defining policy application in reference to spatial volumes solves a number of problems, e.g., it makes practical both policy enforcement and anonymity (or more precisely, pseudonymity). However, it introduces reliance on localization. Making users (instead of venues) responsible for determining the volumes associated with user policies has several benefits. First, it keeps this crucial policy component under user control. Second, it provides users with a strong incentive to accurately report volumes to venues: inaccurate reports might result in user policies being violated. Although user policy volumes might in practice be legally constrained, even without legal constraints, users would have incentives to limit these volumes because using too large a volume would frequently produce irresolvable conflicts with venue policies while providing no benefit to the user.

In this framework, for user policies to be applied, users must accurately specify the appropriate application volumes to venues. This requires localization services associated with users, the primary provider of which will be user-carried smartphones. Instead of rebuilding localization services, we will allow users to report policy locations/volumes using the location services provided by their smartphones, which today are generally based on fused WiFi and GPS data. However, finer resolution may be necessary, which opens questions about localization approaches, e.g., WiFi access point signal strengths, audio environments, inertial measurement unit dead reckoning, and passively inferred crowd-sourced spatial mapping.

Venues will also be responsible for accurate localization, and will have legal incentives to determine the volumes relevant to their sensing infrastructure. Note that, for stationary (but perhaps gimballed) sensors such as cameras, this can be a simple, one-time field-of-view based calibration during sensor deployment. Mobile sensors will require on-line, possibly inertial measurement unit assisted recalibration when locations change. Sensors without clear sensed volumes, e.g., microphones, will introduce some challenges because difficult-to-predict environmental conditions (such as background noise floor) will influence their coverage. This is a continuum problem, i.e., the problem cannot be perfectly solved and applies to all systems using sensors with difficult-to-precisely-determine coverage volumes. However, it is not a fundamental practical problem in our system because nearly all such sensors have superlinear drop-offs in sensitivity with distance. As a result, being slightly conservative about volumes for policy application will produce the intended results with high probability.

Responsive negotiation and enforcement: Policy negotiation in our model is the analog of call/connection setup, while enforcement is the analog of guaranteeing the QoS of the call/connection given the negotiated policy. While negotiation can clearly be done in either a centralized (per-venue) or distributed manner, enforcement requires continuous interaction with the distributed query DAGs. An important challenge here is how to flow information about the negotiated policy and the user scope involved through the DAGs. The obvious approach of simply including the negotiated policy and the user scope along with any sensor data captured from a sensor scope that intersects the user scope is not likely to scale. We envision using commonalities among negotiated policies of multiple users to identify a policy with a content hash. We also envision using simple compact volume representations (e.g., octrees) to capture unions of user scopes. In this manner, a common case may be a relatively large, but simple volume (the composite of several users) tagged with a hash that identifies their common policy.

F. Scheduling, data processing, and storage

Policy enforcement should be as close to sensors as practical to minimize information leakage. To this end, Privacy Backplane will distribute abstract policy operations that the sensors execute locally, e.g., redaction of data captured at a particular location or changes to sensing resolution. These operations will be composed into a set of policy operations that are executed directly at the sensor or distributed across a set of nearby computational resources that can be assembled into a query DAG. This opens questions about which mechanisms should be used to distribute policies across sensor/edge resources to meet performance/energy constraints and which should be supported for policy migration when available resources change.

Data storage and management: Maintaining data privacy requires that any operation accessing data or derivative information must execute inside Privacy Backplane's trusted environment. Therefore, the backplane must manage data storage inside the trusted environment. One of the primary challenges facing data management in Privacy Backplane is the limited resources available on trusted systems, and the need to distribute the processing and data storage requirements across a diverse set of systems. Data cannot always be in motion, so it must be secure while at rest, before all policies have been applied to it. To address this challenge in our prototype, we are building a trusted TEE enabled cloud-based storage service. This opens questions about managing and placing data to support the data rate presented by the sensors and query DAGs, and how the adaptation mechanisms inside the backplane overlay network can ensure that the data movement capacity is sufficient for the processing input requirements.

Distributed adaptive query planning and execution, with policy enforcement: Within Privacy Backplane, query DAGs must be executed with high throughput. Each node in the DAG computes data transformations, forwards a representation of user scopes and their negotiated policies, and may produce redaction feedback. Mapping DAG nodes onto the resources of Privacy Backplane provides an opportunity to optimize both query execution and policy enforcement. First, because numerous DAGs are likely to be simultaneously in effect, especially in a popular venue, it is possible to use common subexpression elimination to reduce the total work done by the distributed system. Second, because DAG nodes are generally pure functions, the mapping can be adapted to workloads. For example, in normal operation, we might push DAG evaluation as close to a venue's sensors as possible. But if a crowd develops, this would overwhelm them, and we would move DAG nodes into the rest of the backplane.

Adaptation will generally be triggered by Privacy Backplane itself. However, the closer we get to the sensors, the more external load must be considered. While Privacy Backplane will run on some sensor platform, its priority will be limited and thus the amount of CPU time available for it will depend on the platform's other activities. To address this, sensor platforms will be able to initiate dynamic load balancing themselves, forcing the backplane to redistribute load to other nodes in Privacy Backplane.

Global real-time model: Policy negotiation has responsiveness requirements because the user is in the loop and

jitter degrades user experience [32] Here, the situation is even worse because without a responsive outcome, the user could easily enter a venue and then only later, after their privacy has been compromised, discover that their policy cannot be reconciled. An additional constraint is that on some machines (e.g., sensor platforms), there must be a mechanism for Privacy Backplane to share very limited resources in a controlled manner with the main task of the machine.

We plan here to use concepts from real-time systems to provide (1) guaranteed responsiveness to the user. Policy negotiation will be of highest priority, and will be treated as a form of admission control for policy enforcement. We will also use real-time concepts to provide (2) guaranteed responsiveness to critical, admission-controlled queries (e.g., criminal investigations), and provide (3) non-real-time, admission-control-free prioritization of non-critical queries (e.g., marketing). The ensemble of admitted work (users and their policies, the venues critical queries) will then execute with global timing constraints respected throughout the system.

Privacy Backplane will be a distributed real-time system with the following design elements. The system software will implement hard real-time properties within devices and other nodes of the distributed system, and these entities will coordinate across nodes to support synchronized soft real-time behavior across the distributed system using approximate global time with error below the human perception threshold. Users and their policies can rely on and leverage this highly predictable behavior. It is important to point out that traditional adaptation or load balancing attempts to minimize latency or maximize throughput, making the set of possible solutions small. In contrast, honoring a deadline allows for many more possible solutions, and thus greater flexibility. We anticipate that this real-time approach to responsiveness will exhibit benefits seen in other applications of real-time systems technology to contexts that do not obviously have a real-time elements [33]–[36].

V. Conclusion

We have made the case for a joint regulatory and technical model for achieving individualized personal privacy within IoT and similar sensing environments. Our initial logical design and architecture was described, along with challenges that result from these, as well as from the overall concept. We are currently in the process of implementing the Privacy Backplane prototype.

References Cited

[1] "European General Data Protection Regulation (GDPR)," https://gdpr-info.eu/.

[2] "California Consumer Privacy Act (CCPA)," https://oag.ca.gov/privacy/ccpa.

[3] Illinois, "(740 ilcs 14/) biometric information privacy act."

[4] Texas, "Business and commerce code, section 503.001."

[5] C. for Data Innovation, "What the Evidence Shows About the Impact of the GDPR After One Year," June 2019.

[6] B. Chen, K. Nahrstedt, and C. Gunter, "ReSPonSe: Real-Time, Secure, and Privacy-Aware Video Redaction System," in *Proc. Int. Conf. on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2018.

[7] J. Wang, et al., "A scalable and privacy-aware IoT service for live video analytics," in *Proc. Multimedia Systems Conf.*, 2017.

[8] J. Wickramasuriya, et al., "Privacy protecting data collection in media spaces," in *Proc. Int. Conf. on Multimedia*, 2004.

[9] N. Davies, et al., "Privacy mediators: Helping IoT cross the chasm," in *Proc. Int. Wkshp. on Mobile Computing Systems and Applications*, 2016.

[10] F. Z. Berrehili and A. Belmekki, "Privacy preservation in the internet of things," in *Advances in Ubiquitous Networking 2*, 2016.

[11] A. Das, et al., "Assisting users in a world full of cameras: A privacy-aware infrastructure for computer vision applications," in *Proc. Conf. on Computer Vision and Pattern Recognition Wkshps.*, 2017.

[12] S. Mehrotra, et al., "Tippers: A privacy cognizant IoT environment," in *Proc. Int. Conf. on Pervasive Computing and Communication Wkshps.*, 2016.

[13] P. Pappachan, et al., "Towards privacy-aware smart buildings: Capturing, communicating, and enforcing privacy policies and preferences," in *Proc. Int. Conf. on Distributed Computing Systems Wkshps.*, 2017.

[14] N. Panwar, et al., "IoT notary: Attestable sensor data capture in IoT environments," *ACM Transactions on the Internet of Things*, vol. 3, no. 1, October 2021.

[15] H. Jin, et al., "Peekaboo: A hub-based approach to enable transparency in data processing within smart homes," in *Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP)*, 2022.

[16] "ARM TrustZone," https://developer.arm.com/technologies/trustzone.

[17] D. Lee, et al., "Keystone: An open framework for architecting trusted execution environments," in *Proc. European Conf. on Computer Systems*, Apr. 2000.

[18] C. Shepherd, R. N. Akram, and K. Markantonakis, "Establishing mutually trusted channels for remote sensing devices with trusted execution environments," in *Proc. Int. Conf. on Availability, Reliability, and Security*, Aug. 2017.

[19] "WebAssembly," https://webassembly.org/.

[20] R. Riesen, et al., "What is a Lightweight Kernel?" in *Proc. Int. Wkshp. on Runtime and Operating Systems for Supercomputers*, 2015.

[21] W. Wang, et al., "Leaky cauldron on the dark land: Understanding memory side-channel hazards in SGX," in *Proc. Conf. on Computer and Communications Security*, B. M. Thuraisingham, et al., Eds. ACM, 2017, pp. 2421–2434.

[22] J. Götzfried, et al., "Cache attacks on Intel SGX," in *Proc. European Wkshp. on Systems Security*, C. Giuffrida and A. Stavrou, Eds. ACM, 2017, pp. 2:1–2:6.

[23] G. Chen, et al., "SgxPectre: Stealing Intel secrets from SGX enclaves via speculative execution," in *Proc. European Symp. on Security and Privacy*. IEEE, 2019, pp. 142–157.

[24] F. Brasser, et al., "DR.SGX: automated and adjustable side-channel protection for SGX using data location randomization," in *Proc. Annual Computer Security Applications Conf.*, D. Balenson, Ed. ACM, 2019, pp. 788–800. [Online]. Available: https://doi.org/10.1145/3359789.3359809

[25] O. Oleksenko, et al., "Varys: Protecting SGX enclaves from practical side-channel attacks," in *USENIX Annual Technical Conf.*, H. S. Gunawi and B. Reed, Eds. USENIX Association, 2018, pp. 227–240.

[26] "Open Portable Trusted Execution Environment," https://www.op-tee.org/.

[27] E. S. Lubana and R. P. Dick, "Digital Foveation: an energy-aware machine vision framework," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, pp. 2371–2380, Nov. 2018.

[28] Y. Zhu, et al., "Euphrates: Algorithm-SoC co-design for low-power mobile continuous vision," arXiv, Tech. Rep., Apr. 2018.
[29] J. Amacher and V. Schiavoni, "On the performance of ARM TrustZone," in *Distributed Applications and Interoperable Systems*. Springer International Publishing, 2019, pp. 133–151.
[30] P. M. VanNostrand, et al., "Confidential deep learning: executing proprietary models on untrusted devices," *arXiv preprint arXiv:1908.10730*, 2019.
[31] Z. Liu, et al., "Trusted-DNN: A TrustZone-based adaptive isolation strategy for deep neural networks," in *ACM Turing Award Celebration Conf.*, 2021, pp. 67–71.
[32] A. Komatsubara, "Psychological upper and lower limits of system response time and user's preferance on skill level," in *Proc. Int. Conf. on Human Computer Interaction*, G. Salvendy, M. J. Smith, and R. J. Koubek, Eds., vol. 1. IEE, Aug. 1997, pp. 829–832.
[33] C. Lu, J. A. Stankovic, and G. Son, Sang H.and Tao, "Feedback control real-time scheduling: Framework, modeling, and algorithms," *Real-Time Systems*, vol. 23, no. 1, pp. 85–126, July 2002.
[34] B. Lin, A. Sundararaj, and P. Dinda, "Time-sharing parallel applications with performance isolation and control," in *Proc. Int. Conf. on Autonomic Computing*, June 2007.
[35] R. Liu, et al., "Tessellation: Space-time partitioning in a many-core client OS," in *Proc. USENIX Conf. on Hot Topics in Parallelism*, Mar. 2009, pp. 10:1–10:6.
[36] S. Peter, et al., "Design principles for end-to-end multicore schedulers," in *Proc. USENIX Conf. on Hot Topics in Parallelism*, June 2010.