Secure Fog Robotics Using the Global Data Plane

John Kubiatowicz (kubitron@cs.berkeley.edu), Ken Lutz (lutz@berkeley.edu),*
Ken Goldberg (goldberg@berkeley.edu),†
Anthony Joseph (adj@eecs.berkeley.edu), Joseph Gonzalaz (jegonzal@berkeley.edu)‡

August 7, 2018

Overview: Today's exciting new world of data-driven technology, AI, and cyber-physical systems rely on information from widely disparate sources, combining processing and storage in the cloud with embedded sensors, actuators, and processing at the edge. This information includes both the *inputs* and *outputs* of computations as well as multi-versioned *executables* that drive such computations. Edge computing is essential, since sensors at the edge of the network (*e.g.*, cameras) generate huge volumes of data that is ideally processed or distilled locally. Further, control loops with sensors, computing, and actuators require local communication for responsiveness and quality of service (QoS).

Unfortunately, the edge of the network consists of islands of trusted hardware interspersed with a vast sea of untrusted hardware. Thus, there is a pressing need for connected systems to uniformly verify the integrity of the information on which they rely and similarly to provide a universal audit-trail for the information that they produce. Further, privacy must be respected by the infrastructure and relinquished only when necessary.

To address these needs, we propose a fundamental refactoring of the network around cryptographically hardened bundles of data, called DataCapsules. DataCapsules are the cyberspace equivalent of shipping containers: uniquely named, secured bundles of information transported over a data-centric "narrow-waist" infrastructure called the Global Data Plane (GDP). The GDP partitions the network into Trust Domains (TDs) to allow clients to reason about the trustworthiness of hardware. When combined with trusted computing enclaves, the GDP enables applications to dynamically partition their functionality between the cloud and network edge, yielding better QoS, lower latency, and enhanced privacy without risking information integrity or obscuring its provenance.

Intellectual Merit: We will pursue research challenges in three distinct areas in this proposal:

- *Secure Networking:* Investigating the architecture of the Global Data Plane (GDP) as a carrier of DataCapsules and study its behavior through deployment in the EdgeNet testbed from US-ignite. The GDP enables data-centric conversations between clients and DataCapsules independent of physical identifiers, allowing DataCapsules to be placed, moved, or replicated based on requirements for availability, durability, and performance. We will investigate how DataCapsules may be used as building blocks for a variety of storage systems with different APIs and consistency semantics.
- *Systems:* Investigating mechanisms for computing at the edge of the network, exploiting untrusted hardware for much of the communication and location infrastructure while exploiting secure enclaves (*e.g.*, using Intel SGX) for computation on DataCapsules at the edge of the network.
- *Intelligent Connected Applications:* Studying new applications, namely Fog Robotics and online system modeling (Data Twins), in the context of the GDP. We will optimize these applications by partitioning them between the cloud and edge, to evaluate how a trust-aware model employing DataCapsules can improve QoS, performance, security, and privacy. We will deploy them in edge environments utilizing a combination of generic EdgeNet testbed nodes and specialized compute nodes supporting secure enclaves for computation.

Broader Impacts: This project marks the genesis of both a new, ubiquitous communication infrastructure and a broader and deeper public understanding of the importance and application of data security measures. We aim to change the level of provenance available to and expected by users of information. By placing our code in the public domain and through collaboration with experimental testbeds such as EdgeNet from US-ignite, we will make the GDP infrastructure available to a wide range of researchers.

*UC Berkeley SwarmLab; See https://swarmlab.berkeley.edu/home/
†Berkeley AUTOLAB; See http://autolab.berkeley.edu/
‡Berkeley RISELab; See https://rise.cs.berkeley.edu/

Project Description

1 Introduction

The need for universal attribution and protection of information has become acute, since modern connected applications rely on information from diverse sources, including ubiquitous sensors, machine-generated models, databases in the cloud, and other human groomed sources of information. The correctness and reliability of this information has a direct impact on the reliability of modern mission-critical applications with life-impacting outcomes. Examples in this category include industrial robots, autonomous vehicles, Google Home, Amazon Echo, home cleaning systems (iRobot), and many IoT-style appliances with personal information. This critical information includes both the *inputs* and *outputs* of computations as well as multi-versioned *executables* that drive such computations. Should incorrect information be introduced either accidentally or maliciously, these modern applications could misbehave with disasterous results—property damage, injury, or even loss of life. Further, the information handled by these applications is extremely sensitive (*e.g.*, health data, personal identities, etc.) and must not be exposed.

Unfortunately, many sophisticated applications make use of middleware that assumes by default that it operates in a trusted, non-malicious environment (*i.e.*, relying on border security from firewalls) – an assumption that is clearly at odds with both the widely distributed nature of these applications and the multi-tenant usage of the edge infrastructure[1]. The "next best thing" solution employed by many other applications is to utilize *secure channels* [57, 56] to encrypt information between communicating devices; while better than no security at all, this solution relies on the endpoints to be physically secure and unbreachable. Such a solution reflects an implicit *trust* in the endpoints – a trust that *may* be justified when communicating with cloud providers but is likely misplaced with devices at the edge of the network [22, 52, 25, 38, 23, 68]. In fact, privacy and security weaknesses in intelligent, connected systems seem to emerge daily [27, 26].

Further, even when applications ostensibly secure their information, they typically rely on ad-hoc or proprietary storage solutions utilizing the cloud (including situations in which producers and consumers are at the same edge location). Not only does this ubiquitous interaction with the cloud increase latency *and* timing variability, it introduces vulnerability to observation by unknown parties since communication traverses many networks on its way to and from the cloud. Risks of such traversal exist even when information is encrypted because of a vast array of information leakage through timing channels [6, 63].

Since each new application has hand-crafted storage and communication technology, designing these applications results in duplicated work and inevitable bugs or design flaws. The result is fragile, stove-piped applications that exhibit high-latency, high variability, and an inability to easily share with other stove-piped applications; *this lack of interoperability is a prominent feature of modern IoT infrastructures*. Lower latency and more predictable behavior (better QoS) could be achievable if data could be moved close to where it is produced and used, but this simple optimization is often avoided because it requires customized caching, security, and synchronization solutions *in addition to* remote storage mechanisms in the cloud. Even worse, such single-use infrastructures reinvent the same security mechanisms over and over, leading to unvetted and possibly flawed security mechanisms. Such flaws pose a unique threat to the widespread adoption of robots and smart technology especially for home security and surveillance applications.

At the same time, the shear volume of information involved in modern applications continues to increase at a rapid pace, leading to a increasing need for computation at the edge of the network – close to the sources and ultimate sinks of this information (in addition to secure interaction with the cloud for model building and access to global data and computation). Designing custom, secure and adaptive protocols for moving information between the cloud and secured computation at the edge is complex and fraught with challenges for the same reasons as mentioned above, including an inability to judge which edge components are trustworthy and which are not. In fact, the edge of the network consists of islands of trusted hardware adrift in a vast sea of untrusted hardware (*i.e.*, poorly maintained, easily compromised, or owned by an untrusted party). What is needed are new, uniform mechanisms for securely and privately distributing both code *and* data to the edge while protecting both the providers of computations and owners of the information.

[1] For instance, the first version of the Robot Operating System (ROS) [53, 54] assumed by default that it was operating in a trusted, non-malicious environment. The subsequent version, called ROS2, addresses security by using the DDS middleware for transport [44].

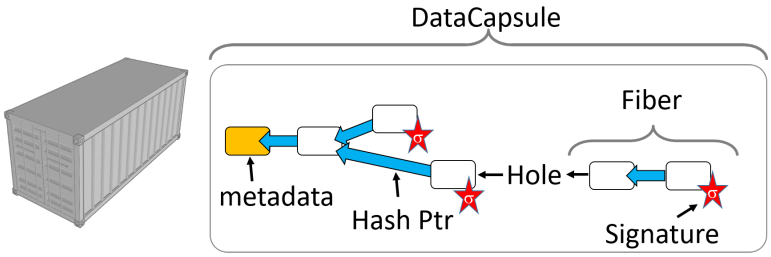

Figure 1: A *DataCapsule* is like a shipping container for information. It contains standardized metadata with a cryptographically linked transaction history, secured by signatures. It has a unique name which is a 256-bit hash over the metadata (including public keys of one or more *owners*). Individual transactions may be formatted in any way and may be encrypted for privacy. Verifiable chunks of information, called Fibers, are returned as a result of reading a DataCapsule. Only authorized *owners* of a DataCapsule can add new Fibers to the DataCapsule. Multi-writer storage systems with custom semantics are built with *services* that export common storage APIs (*e.g.*, Filesystems, Databases) to clients.

1.1 Our Approach: Refactor the Network Around Information and Trust

To address interoperability, performance, and security in intelligent connected applications, we propose a fundamental refactoring of the information and communication infrastructure into a new data-centric "narrow waist" focused on *information* rather than *physical, destination-based packet communication*. Unlike other data-centric proposals (such as NDN [74]), our proposal provides data update and attribution semantics by introducing a new entity, called a *DataCapsule*, that operates on top of a data-centric infrastructure that we call the Global Data Plane (GDP). As shown in Figure 1, DataCapsules may be thought of as the information equivalent of the ubiquitous shipping containers seen at every port and on ships, trains, and trucks. DataCapsules are standardized metadata containers holding cryptographically linked information histories which are directed graphs of transactions with provenance (*e.g.*, signatures). They have unique names derived from hashes over the metadata (shown in yellow in Figure 1). Further, they are lightweight enough that every sensor could have an associated DataCapsule while at the same time powerful enough that a DataCapsule could hold the complete "ground truth" of a large database. Application writers view DataCapsules through the lens of "Common Access APIs" (CAAPIs) which make the corresponding DataCapsules appear as file systems, data bases, key-value stores, or other common storage entities.

The GDP enables location-independent conversations between clients (or application components) and DataCapsules based only on DataCasules' unique names. Further, the GDP is organized as a graph of Trust Domains (TDs), enabling communication to be restricted to portions of the network trusted by DataCapsule owners. Since conversations with DataCapsules do not involve physical identifiers, such as IP addresses, they may be placed, moved, or replicated based entirely on requirements for availability, durability, performance, or privacy without compromising the integrity or global shareability of enclosed information.

Conversations with DataCapsules involve *Fibers*, which are self-authenticated fragments of DataCapsules (see Figure 1). Fibers returned from read operations on DataCapsules allow readers to validate returned information. A DataCapsule writer generates a Fiber for inclusion into the DataCapsule; such Fibers are admitted only when they meet the security constraints enforced by the enclosing metadata[2]. Membership of a transaction in a given DataCapsule may be verified by anyone with accesss to that DataCapsule.

1.2 New optimizations through agile secure edge computing

To protect information while enabling on-demand edge computing, the GDP infrastructure is ideally coupled with secure, lightweight, virtual machine containers that can be entrusted with secret keys and attested code. Such containers should be able to execute in an edge environment, decrypt information from and make transformations to DataCapsules without risking physical attacks, namely those in which an adversary stops a running machine, dumps memory, and recovers either keys, executables, or decrypted information. We propose to study the requirements for such secure virtual containers by developing an initial version of these containers with SGX enclaves [4], with particular attention to multi-tenancy and efficiency.

[2]The simplest inclusion criteria is that the Fiber is properly linked to existing transactions and is signed by the owner of the DataCapsule. A more interesting criteria is that the Fiber meets the needs of an admission script resembling transaction scripts in blockchain applications such as BitCoin [5].

An important aspect of edge computing (to complement the mobility of DataCapsules) is *agility*, namely the ability to instantiate a secure computation on demand in an edge environment–complete with code, cryptographic keys, and data inputs and outputs. Section 4 discusses our approach in more detail. Such computations can act as *services* for other computations or as localized components of a distributed application–thereby opening up a broad new class of optimizations through locality-based refactoring.

For instance, DataCapsules may be moved and replicated to optimize performance, availability, durability, or privacy without risking integrity or obscuring the information provenance. Computations may then be instantiated in an edge domain to take advantage of the movement of information. Moving active information onboard robots becomes an obvious optimization choice, as does moving one of multiple DataCapsule replicas into a stub network domain to guarantee that local control loops will operate efficiently and information from vital sensors is not lost during network partitions. We propose to investigate several such optimizations for a distributed network of robots under various trade-offs such as latency, bandwidth, privacy and security.

1.3 Why do we need the Global Data Plane?

Many storage and communication systems have been developed over the last decade, harnessing cloud infrastructures and information-centric networks (ICNs) [18, 24, 3]. However, none of these solutions quite address the rapid growth of information devices and computation at the edge of the network. Despite numerous advantages, the cloud is ill-suited as an *exclusive infrastructure* for information applications and cyberphysical systems: communication between edge domains and the cloud can have high latency, high variability, and too public visibility [73, 47]. Further, while many ICN solutions have arisen over the last decade, they are oriented around packets of *data* rather than bundles of self-authenticating *information*. This distinction is subtle, but important: DataCapsules provide *standardized* semantics for *when* information may be admitted, *how* this information is related to other related information, and *who* produced this information. Yet, these semantics are not heavyweight or restrictive; instead, they provide a universal primitive on which to build a cornucopia of storage models, consistency mechanisms, and caching protocols.

Thus, DataCapsules really *do* resemble cargo containers: they may be embedded into any networks or subnetworks; they can live in the cloud or at the edge or be transported on demand. While many of the ideas and mechanisms behind the GDP and DataCapsules have been known for over 20 years, it is the *standardization* and *generality* that are the most powerful features of the DataCapsule value proposition: DataCapsules are lightweight enough that one could envision a whole new class of network hardware (*e.g.*, routers, storage servers, switches, edge computation devices) that are targeted at DataCapsules.

2 Application Case Studies

We propose to investigate the advantages of programming to an information-centric API via realistic application studies. Two of the co-PIs of this proposal are domain experts in robotics and machine learning. Further the proposing faculty have close ties to the Berkeley RISE Laboratory and US-Ignite, yielding many opportunities for technology transfer and further investigation.

Our proposed application domains have potential benefit to the US economy: *Fog Robotics*, specifically applied to distributed mobile robots using deep learning for “surface decluttering” and *Digital Twins*, which are continuously updated digital models of physical systems. These case studies are data and computation-demanding examples of the Internet of Things (IoT), a broader umbrella of applications based on distributed sensors, actuators, devices, smart home appliances, embedded electronics, and other devices.

2.1 Fog Robotics

Fog Computing [10] extends Cloud Computing by introducing the opportunity for partitioning applications between edge devices and the cloud to better address privacy, security, scalability, latency, bandwidth, availability and durability control [71, 19, 69]. Analogously, “Fog Robotics” [1, 2] extends Cloud Robotics, defined as “a robot or automation system that relies on either data or code from a network to support its operation, *i.e.*, where not all sensing, computation, and memory is integrated into a single standalone system” [31]. Fog Robots may be deployed in environments where privacy and security are concerns, such as households or factories. In a household, individual privacy and household security are concerns, while for a factory, preservation of corporate trade secrets is a concern.

This project will investigate ways in which a Fog Robot system can exploit the GDP to safely and efficiently refine and learn sensing and control policies for surface decluttering from high-dimensional multi-modal signals in a multi-agent distributed setting, while preserving the privacy and security of the data[3]. Specifically, we will consider the Dex-Net approach to robust robot grasping of a diverse variety of objects [42, 43]. This approach incorporates deep convolutional neural networks (CNNs) that take as input 3D point clouds from structured lighting sensors and rapidly compute parallel-jaw grasps of objects. Dex-Net offers a promising approach for surface decluttering but currently does not continue to learn after initial deployment.

In effect, the Deep learning is performed in the Cloud—at remote data centers with big datasets. The resulting weights—the deep model consisting of millions of parameters—can be downloaded to robots that can apply the model to sensory input to compute appropriate grasping and manipulation actions.

To enable continuous learning, this project will consider a Fog Robotics scenario where positive and negative physical "episodes" are collected by every robot. Each episode includes: a sensor image, computed grasp with associated confidence value, and a performance report (*e.g.*, whether the object was successfully grasped or not). As the sensor images are from the interiors of homes or offices, the sensor images and performance reports are private; hence, we propose to place each such episode into a Data-Capsule and to periodically or continuously debug and retrain the deep model weights using Fog Computing.

We further propose to investigate a fog-robotics approach which leverages the GDP to securely democratize and scale robot learning over a distributed network of robots: exploiting DataCapsules with encryption for privacy, security through provenance, and low latency/high bandwidth access to resources at the edge.

2.2 Distributed Digital Twins

Rich sensing coupled with fast and secure networks have unlocked the potential for virtual models of physicals systems that can be used to better manage and actuate those systems. Recently, [11, 62] proposed the concept of *Digital Twins* referring to virtual models of industrial systems that can be used to predict failures and manage maintenance. A Digital Twin of a physical system combines all the sensory data collected about that system with sophisticated models of the system to enable users to predict how the system will perform in the future. For example, a Digital Twin of a jet engine would collect all the telemetry and service history of a particular jet engine into a single physical model. This model could then be used to digitally simulate the future failure modes of that engine and predict when maintenance will be needed.

Much of the focus and excitement around Digital Twins has been in the context of aggregating data in large cloud based models. However, by capturing and modeling data from many related systems at a much larger scale we have the opportunity to compose the Digital Twin models to better capture context and make more complete decisions. Furthermore, in many cases aggregating the entire model to a single location will be impractical due to latency and security constraints. As a consequence we need ways to distribute the models that enable Digital Twins across a complex fabric of compute and storage devices.

When deployed in settings with actuation (*e.g.*, Fog Robotics) these *Distributed Digital Twin* models will allow more intelligent planning. For example, Digital Twins of robots in a factory can share information about errors in parts to accommodate those errors during their assembly. This planning and interaction can happen quickly on a single assembly line requiring low latency planning and model interaction. However, it may also happen across a supply chain over extended periods of time requiring longer term storage and model movement. Thus, we must be able to move the execution of models across contexts in space and time as well as between trust zones while preserving the security and integrity of models and queries.

Fundamentally, models that enable intelligent planning and execution of complex physical systems must balance computation and latency requirements with security and privacy. To address these challenges we will study new ways to support model inference. On the edge, we may employ smaller, more lightweight models, perhaps with lower floating point precision and decreased accuracy, to meet predictions within latency requirements. In the cloud, we may want to combine models to improve accuracy at the expense of increased latency and higher compute demand. We must also be aware of privacy and security concerns; models developed in one context may need to be applied to sensitive data streams in another.

[3]2017-2020 NSF NRI Project, "Scalable Collaborative Human-Robot Learning". The proposed project will investigate data and systems issues that can greatly enhance applications such as robot surface decluttering but are far outside the scope of the ongoing NRI project which focuses on learning algorithms that assume reliable and secure networks.

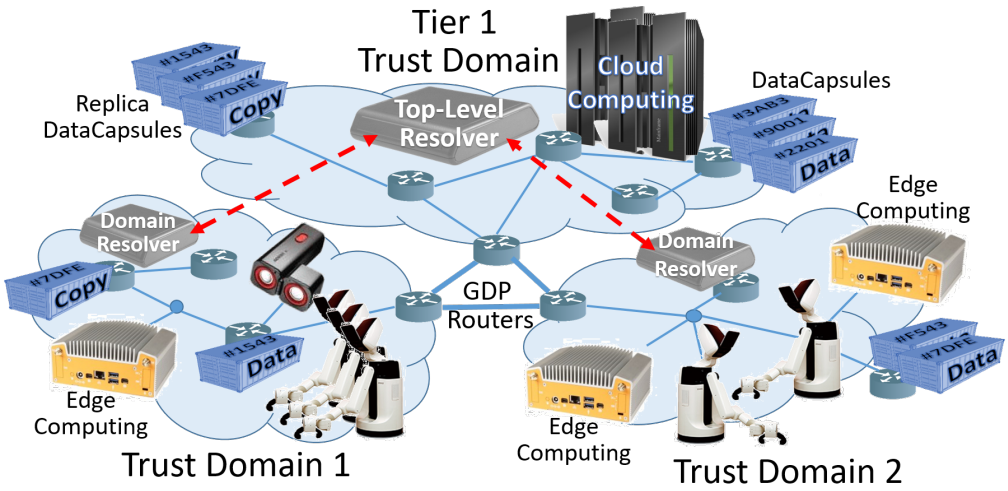

Figure 2: Our Proposal in a Single Figure: *DataCapsules*, shown as cargo-containers, are *advertised* by Capsule Servers (not shown) embedded in the network. The *GDP Routing Layer* carries GDPinUDP datagram packets; this overlay mesh is structured according to trust domains (TDs) and physical locality. The *GDP Location Resolution Layer* is a graph of connected resolvers, one per TD. Resolvers can reside on a single physical node or be widely distributed. "Tier-1" resolvers must track all globally-visible DataCapsules, while other resolvers track DataCapsules for their local domain and peer TDs. Also shown: multi-tenant cloud or edge computing nodes supporting agile secure containers.

3 The Global Data Plane

DataCapsules and the Global Data Plane (GDP) provide a universal storage mechanism on top of which a variety of storage semantics and access models can be constructed. Here, we sketch our proposed infrastructure, reflecting several important criteria: First, we seek a highly scalable architecture, supporting an arbitrary number of DataCapsules and simultaneous conversations[4]. Second, we seek a system that resembles the current topology of the Internet sufficiently well to support efficient communication and a realistic mapping to natural boundaries of trust. Third, we seek to meet the needs of modern information applications; one example is incorporating the need for efficient, secure multicast to subscribers.

3.1 DataCapsules: A Universal Primitive for Secure Mobile Storage

As discussed in Section 1.1, DataCapsules are metadata containers holding an interlinked history of transactions and anchored by signatures. Transactions added to a DataCapsule are *immutable*. Consequently, DataCapsules can be viewed as secure, append-only, versioned logs[5]. Consequently, the result of synchronizing two DataCapsules with the same name is a new DataCapsule containing all the transactions held by either (*i.e.*, the union of the two DataCapsules). Note that the holes shown in Figure 1 might have resulted from a network partition or server crash during write.

DataCapsules are *building blocks* on which to build secure, globally-sharable information repositories. By themselves, they do not provide consistency semantics or multi-writer access control. Instead, DataCapsules provide support for explicitly sequencing related transactions and restricting additions (append operations) to a single writer (*e.g.*, cryptographic principal of the owner). On top of DataCapsules one can build arbitrarily storage semantics through use of services running in secure containers.

Before discussing storage semantics, however, we briefly mention the Storage Service Provider (*i.e.*, SSP or "storage provider") which is an entity or organization responsible for storing DataCapsules on behalf of owners. Storage providers are an explicit aspect of the GDP model [36]. DataCapsules owners choose their storage providers based on a variety of criteria and sign their identity into DataCapsule metadata. We briefly mention two ways in which the storage provider assists in providing data semantics:

[4]One important target is handling *one trillion* devices [41], each of which should be able to have an associated DataCapsule. Another target might be to support an infrastructure in which each file or database in the world could be housed in a DataCapsule.

[5]DataCapsules are superficially similar to **git** repositories [40, 13]. However, unlike **git**, the official history in a DataCapsule can only be truncated, never changed after the fact. History can be accessed in many ways including through CAAPIs, described later.

Freshness As a Service: The immutable structure of a DataCapsule history graph prevents substitution attacks (in which a malicious entity attempts to insert phony transactions into the history). Further, this structure transforms the traditional *coherence* problem between replicas into a *freshness* problem, namely the need to ascertain the latest transaction at the leaf of a history branch. In traditional, centralized storage systems, the storage provider is *implicitly* trusted to return the latest version of information during a query.

In contrast, the GDP makes freshness an *explicit* property of DataCapsule service. Owners of particular DataCapsules choose their storage providers and request application-appropriate freshness. Although not in keeping with the distributed philosophy of the GDP, a service provider might *assert* the correctness of a master copy, thereby guaranteeing fresh results to read queries. Alternatively, a service provider might provide serialization through distributed consensus such as Paxos [37, 14], RAFT [50] or Byzantine Agreement [12, 17], thereby providing distributed freshness. Or, the application may handle freshness on its own, via a service that tracks the latest version or by adding timestamps to transactions and requiring reads to have a certain minimal freshness. We will investigate the various options for data freshness.

Durability as a Service: The storage provider can vary the level of durability by choosing the placement and number of replicas, the frequency of anti-entropy (synchronization) operations, and the aggressiveness with which new replicas are introduced into the network when a replica fails. The structure of a DataCapsule (an immutable graph of transactions) greatly simplifies the complexity of replica synchronization.

3.2 Single Writer Semantics and Common Access APIs (CAAPIs)

In this section we sketch how threads running in secure enclaves provide a shim on top of DataCapsules to provide a variety of storage models and consistency policies to applications.

A DataCapsule may be written by any entity that possesses its secret signing key and (potentially) one or more encryption keys. The signing key is a cryptographic principal that we view as the *single writer* for this DataCapsule. In the simplest case, a single sensor or fixed computational device stores the signing key in secure on-board hardware and uses it to write to a DataCapsule. This single piece of hardware sets the ordering of transactions written to the DataCapsule[6]. For a piece of hardware with sufficient RAM buffering, the ordering of transactions is uniquely set by the single writer, guaranteeing no branches in the DataCapsule history–regardless of the number of replicas or network failures.

While a single, linear chain of transactions may make sense for streaming applications, the full power of DataCapsules is realized in the presence of secure computational containers (such as those discussed in Section 4.1 at the edge of the network or in physically secured servers). Such containers can be entrusted with signing and encryption keys for full access to underlying DataCapsules. As a result, such computational processes can export familiar APIs to applications (such as databases with SQL, file systems, key-value stores, *etc.*) while placing their persistent state into an underlying DataCapsule. We say that these services provide Common Access APIs (or CAAPIs for short) to upstream applications.

CAAPIs keep their permanent state in DataCapsules, but load this state into on-board RAM at startup time, reorganizing it to provide fast access to clients. As an example, a database CAAPI would keep the latest version of each column in RAM, loading a checkpoint and delta records from DataCapsules at startup. Subsequent SQL queries can operate at high speed directly from RAM.

Strong consistency can be provided in the presence of multiple writers in a number of ways: The simplest is to funnel all writes through a single thread running on a single piece of hardware. A higher-performance version could exploit multiple threads running distributed consensus with Paxos or RAFT, in which case each node would possess the same signing key. Finally, a paranoid application might run a Byzantine Agreement algorithm, in which case each participating node would possess a unique signing key with DataCapsule accepting writes signed by a threshold number of nodes [59], as specified by a commit script [5].

3.3 Trust Domains

To aid clients in making trust decisions, we consider the network infrastructure (and Global Data Plane) to be physically divided into a series of “Trust Domains” (TDs) that reflect the ownership, trustworthiness, and degree of maintenance of the physical elements (routers, DataCapsule servers, and Edge Computing

[6] Order is set by linking a hash of the previous write into a new transaction before signing it and sending it into the GDP network.

elements) within these domains. All of the physical hardware within a TD is considered to be owned and managed by a single entity. This entity typically has a unique economic, political, or other incentive structure.

TDs may be called upon to store DataCapsules, keep track of the locations of DataCapsules within their domain, or even forward transactions for remote DataCapsules into other TDs. We assume that TDs perform these actions according to their own motivations, such as being paid as a utility provider. Individual clients will trust some TDs and distrust others. TDs may also form pair-wise alliances with other TDs to shorten the path of transactions between TDs with "mutual trust" [49]. We propose to investigate how to best respect the trust preferences of individuals by confining their information to TDs that they trust.

One example of a TD might be a company that has deployed Fog Robots on its premises and owns its own networking, storage, and edge computing resources. Another example might be a cloud computing provider such as Amazon. In this model, the internal computational and networking environment of a given robot could even be identified with a TD. Many of the interesting technical mechanisms of the edge computing and Global Data Plane infrastructure discussed below can be investigated with a single (ubiquitous) TD. However, the expansion into multiple TDs reveals the full power of the proposal.

Every TD must be identified by one or more public keys issued by the maintainer of the domain. Ideally, every piece of hardware within a domain should possess its own identity in the form of a public/private key pair certified by the TD (*i.e.*, its owner). In addition, each piece of hardware may have properties certified by its owner, such as an ability to securely sign information or support secure enclaves (Section 4.1).

3.4 GDP Routing Infrastructure

The function of the Global Data Plane is to route conversations between endpoints such as DataCapsules, sensors, actuators, services, clients, *etc*. Each of these endpoints is defined by a unique, location-independent name, derived as a 256-bit hash over metadata including a public key; in fact, we assume that every distinct physical element of the GDP infrastructure has such a unique name (and cryptographic principal) as well. Conversations between endpoints can occur through the GDP routing infrastructure regardless of the location of these elements relative to one another. In keeping with our discussion of trust, we assume that conversations are routed through TDs compatible with endpoints.

To meet our goals of scalability and performance, we propose a two-layered architecture for the GDP; see Figure 2. The lower *routing* layer consists of overlay routers connected in a topology that matches the physical structure of the network and Internet; connections between trust domains roughly correspond to inter-ISP peering arrangements [49]. The upper *location resolution* layer tracks the location of endpoints by mapping names to locations and other nameable resources (such as active participants in multicast publication/subscription trees). The routing layer must only hold sufficient information in its routing tables (fast memory) to cover all *active communications* between clients and DataCapsules. In contrast, the resolution layer must hold state for *every replica of every DataCapsule, whether in active use or idle*; the large number of DataCapsules we imagine for the GDP favors slower-but-extremely scalable data lookup.

Interactions between clients and DataCapsules consist of *Fibers*, which are self-authenticated chunks of information that flow over the GDP routing network. Since Fibers are undoubtably larger than the minimum transfer unit (MTU) of the underlying network, and since Fibers must be reliably transmitted under many circumstances, we propose a UDP-based tunneling protocol at the lowest layer that we call "GDPinUDP" with a NAK-based retransmission mechanism for reliability over point-to-point and multicast conversations. The GDPinUDP packet format would be routeable in the flat address space, but could also be routed via short "flow-ids", thereby compressing large headers and leading to to lower overhead for ongoing communications.

We propose to utilize Click [48, 32] to build routers for GDPinUDP packets. Packets will be forwarded along neighbor links (see Figure 2) according to routing information acquired from the location infrastructure. Routing table entries are constructed by querying resolution servers, starting with the local TD and proceeding to the top-level TD (see Section 3.5). We will investigate a variety of ways to select overlay neighbors, including: (1) local links *within an edge TD* acquired from NetFlow, traceroute, or hand-configured links, (2) neighbors *within a large TD* acquired from a locality-aware DHT network [66, 39] in which each overlay router registers its unique ID in the resolution infrastructure, or (3) explicit inter-TD links configured according to pair-wise TD relationships. To guarantee that inter-router traffic comes from authorized neighbors, we propose to route GDPinUDP traffic using DTLS [58], which prevents insertion of traffic into inter-router links.

3.5 Advertisement and the Location Resolution Process

As shown in Figure 2, DataCapsules are stored on Capsule Servers spread throughout the network to exploit both physical locality and locality of trust. Capsule Servers that export a given DataCapsule must be owned/operated by a Storage Service Provider acceptable to the owner of the DataCapsule. Multiple replicas of a given DataCapsule may be exported in different parts of the network.

Each trust domain (TD) contains an instance of a *resolution service* that stores *advertisements*, which are certificates asserting the mapping between DataCapsules (by name) and the location of Capsule Servers exporting them. A Capsule Server exports advertisements, one per served DataCapsule, asserting that it can serve read or write requests for the DataCapsules. These certificates are verifiable by all network elements and include proofs that the given Capsule Server is allowed to advertise the given DataCapsules. These advertisements are transmitted to the TD-local resolution service, which forwards them *upstream* to Tier-1 domains and horizontally to TDs with GDP peering arrangements.

Top-level (Tier-1) TDs must be capable of locating all globally-visible DataCapsules and routing transactions to any global DataCapsule wherever it might reside, while lower-level TDs may only be required to locate DataCapsules within their own domains. In practice, this means that top-level TDs must use extremely scalable data location infrastructure (*e.g.*, a peer-to-peer infrastructure) while lower-level TDs may use a much simpler data location infrastructure (such as a single-node database). This resolution topology resembles that proposed by Donar [33], and is intentionally modeled after the current internet peering architecture [49]. Note that advertisements may be freely shared between routers and across trust boundaries. Further, for simplicity, advertisements are treated as soft state (*i.e.*, must be renewed regularly).

Dynamic Route Construction: Conversations between clients and idle DataCapsules (*i.e.*, DataCapsules that are not currently being read *or* written) must start by locating relevant storage providers and building routes to them. We propose a dynamic process that queries local resolvers for information, then follows the advertisement path to build a custom route between clients and targeted DataCapsules. When multiple DataCapsules exist with the *same* name, this process will result in a multicast tree. Only DataCapsules with valid advertisement certificates and clients with proper read or write credentials are allowed to join this tree. Efficient versions of the resolution and route construction process is a deliverable of this proposal.

Secure Multicast Tree Construction: As hinted above, we propose to investigate a secure multicast tree-building primitive, supported directly by the GDP, that will be utilized for *both* replica update *and* publish/subscribe. The need for multicast in IoT has been recognized elsewhere [55]. Since these multicast trees may carry multimedia traffic, we propose to support a NAKing scheme for reliability – something well known in the literature [64]. To join a multicast tree, an endpoint must present a valid certificate such as an advertisement certificate (for Capsule Servers) or a subscription certificate (for clients). The dynamic tree-building process is one of the deliverables of this work and should follow the hierarchical architecture of the trust domains (TDs), with GDP routers at the borders acting as multicast distribution points.

3.6 Scalability

The current Internet architecture has shown great resilience and scalability. To the extent that *existing* applications can employ the GDP through a simple refactoring of network interfaces, we point out that our combination of a scalable location infrastructure on top of an efficient datagram-based switching plane is topologically compatible with the existing IP network and should be extremely efficient.

Further, considering *future* applications in which the convenience of the GDP leads to higher levels of communication, our hypothesis is that the graph of simultaneous connections between endpoints in the GDP is nowhere close to an “all-to-all” pattern. In fact, because of privacy and QoS concerns, many communications between applications and DataCapsules will be physically local, except for durability (replication) traffic. The vast array of running application instances will utilize disjoint subsets of DataCapsules and connect disjoint subsets of devices. One trend that we can surmise from existing Internet applications, however, is that the ease of sharing within the GDP is likely to lead to a (relatively small number) of high-fanout multicast trees for subscription; consequently, the multicast mechanism must be extremely efficient.

Finally, to the extent that most DataCapsules will hold historical or otherwise inactive information, the two-layered architecture is ideal because (1) idle DataCapsules consume no resources in the routing in-

frastructure; further (2) idle DataCapsules only consume storage on Capsule Servers that store them and Resolution Servers in the local TDs (with agreements to advertise them); and finally (3) idle DataCapsules consume advertising resources in TDs with peering arrangements (negotiated agreements) and in an associated Tier-1 TD (contracted and perhaps paid for). These advertising resources consist of scalable database lookups which are not in the critical path of any ongoing communications.

4 Secure Containers and Agile Function Deployment

In addition to the privacy and QoS benefits of confining information to edge domains, the bandwidth requirements of edge sensors such as cameras continues to increase – stressing bisection bandwidth to cloud. Thus, many object recognition, distillation, and anonymization tasks should occur at the edge. Further, when sensors are on mobile platforms such as robots, the intra-system bandwidth (within the robot) is vastly larger than the inter-system bandwidth (between neighboring robots or to the cloud). The obvious solution is to bring computation to the edge domains (or onto robots). Further, a mechanism for roaming entities to request transient computation as a service is an obvious addition to the Fog Robotics story.

Unfortunately, when viewed from the vantage point of privacy, security, and provenance, edge computing is risky since the physical computational elements are out of control of the owners or maintainers of the information on which they compute. For instance, one concern is a physical "stop and dump" attack, in which an attacker with physical access to an edge computing node stops the node (such as by stopping the processor clock) and dumps the active state of the computation. Such a dump might reveal cryptographic keys, proprietary executables, or decrypted contents of storage repositories. Such cryptographic keys must be present in order to (1) decrypt information from remote data repositories; (2) provide signatures for information generated by this edge computation; and (3) encrypt outgoing information headed to remote data repositories. Such "stop and dump" attacks may be mitigated in a number of ways including hardware enforced containers, built on technology such as Intel SGX and future variants, or trusting the owner of the hardware to physically secure their servers and promise not to attack them in this way.

4.1 Secure Enclaves for Computation at the Edge

In order to support our vision of secure attribution, we propose to develop technology that provides an agile, secure container (or micro-virtual machine) environment that can withstand "stop and dump" attacks while instantiating edge computation on demand. When coupled with the GDP, such containers provide on-demand computations that move with physical devices/systems/people or toward large repositories of information or specialized compute engines. The devices that support such secure containers must be (1) multi-tenant and (2) capable of lightweight instantiation of secure computations running within the containers. Second we propose to investigate the role of more traditional container infrastructures (such as Linux containers or Docker) in delivering, service, routing, and locating encrypted data containers such as DataCapsules.

Our approach will start with existing attempts to produce agile and secure container infrastructures, such as SCONE [7] which combines SGX [4] and Docker to produce low-overhead secure enclaves. We will extend DataCapsule access libraries into the enclaves (for ease of use by edge computations) as well as demonstrate ability to launch active GDP elements (routers, resolution elements, DataCapsule servers).

4.2 Agile Deployment on Multi-Tenant Hardware

Further, we propose to build a service-location facility on top of the GDP with a specific goal of serving secure computational enclaves to clients at the edges of the network. This facility will be built on top of a native discovery facility such as ZeroConf [65, 15] to export the locations, trust credentials, and capabilities of local multi-tenant computing platforms. New computations can then be dynamically instantiated by presenting DataCapsules (representing code and data) as well as cryptographic keys (for signing and encryption). In addition to delivering a secure container programming environment, we will investigate the use of this environment to provide computational services for high-bandwidth processing of edge sensors (such as cameras, etc) as well as deploying CAAPIs (See Section 3.2) for application-specific data semantics[7].

[7]Note that our approach is entirely complementary to the approach of the NSF Expeditions project in the Berkeley RISE Lab. The focus of edge computing here is to allow secure containers (enclaves) to be instantiated on the fly in an edge trust domain (TD) and utilized to perform computations on encrypted data in DataCapsules as well as to produce data for DataCapsules.

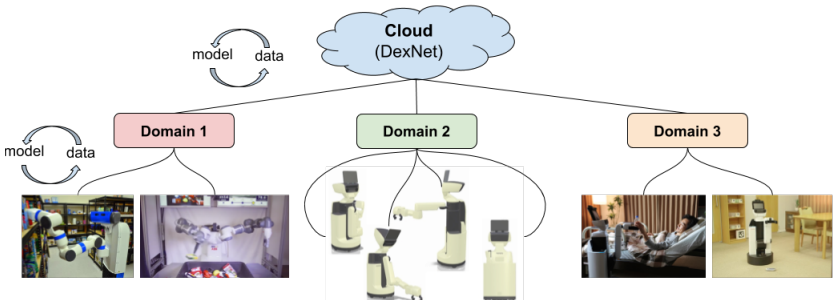

Figure 3: Fog robots collect information which needs to be protected and not sent through untrusted domains. Data collection and model adaptation frequently occurs with secure enclaves at the edge within each domain, and occasionally over the cloud service. Local processing of high-bandwidth sensors may also occur at the edge.

5 Applications and Experimental Deployment

To investigate the power of the Global Data Plane, we have selected two exciting new application domains, namely Fog Robotics and Data Twins. By redesigning applications in these two domains to exploit the Global Data Plane, we investigate the performance, QoS, and privacy advantages of operating with a secured information infrastructure. In this section, we detail our proposed research in this domain.

5.1 Fog Robotics and Decluttering

In the distributed network of fog robots, the robots collect information which needs to be protected and not sent through untrusted domains. Each domain would like to utilize models that have been built across many domains, but would like to trust the provenance of that information. The data shared by the robot is used to learn/update the grasping model for decluttering with secured containers at the edge (see Figure 3).

Secure and Large Scale Distributed Robot Learning: Consider a robot in each different location such as a home, office and a warehouse. The task of the robot is to declutter the objects in the environment using point cloud images for planning grasps. We consider both *distributed acting*, where multiple simultaneous input streams serve as data sources, along with *distributed learning*, where heterogeneous resources are used to facilitate learning (*i.e.*, distributed SGD) [61]. Combining Fog Robotics with the GDP brings a unique opportunity to study these processes together, along with a variety of other design options:

- **Secure Learning:** During decluttering, the robot accumulates the failure cases for performance monitoring and improving its performance over time. Such information may be sensitive and is not required to be shared outside the trusted domain. Within the trusted domain, the robot can frequently share data for retraining the grasping model to avoid the potential failure cases. The access control and data provenance is implemented with the GDP infrastructure. The cloud provides a central repository for providing signed updates to each specific robot for a set of tasks. Each organization may have different incentives to share the data and get centralized updates across the trust domains in a secured manner.
- **Shared learning over confidential data:** Security concerns are on the rise with robots encroaching the personal space in homes and organizations. Fear of misuse of personal data and leaking competitive information to competitors poses a unique challenge to a shared learning framework. Learning must be personalized to each robot in respective home and organization, but increasing the pool of data used to feed data hungry deep learning algorithms can result in better representation and better performance of all robots. Due to security concerns, we will investigate data placement strategies (*i.e.*, what data should be sent to improve the model on the server, what data should be collected and kept on the edge) along with filtering out the data considered confidential [72].
- **Distributed Multi-agent Learning:** With a large scale distributed network of robots, we plan to explore a variety of training schemes such as how the robots can learn from each other across different environmental situations, bootstrapping off our prior work in learning from demonstration which makes use of a human demonstrator [67, 34]. Robots can also learn to collaborate on certain tasks by optimizing a global objective, in addition to sharing learned representations across on-device models, while preserving privacy in the data transferred. Finally, we can observe emergent behaviors of multiple agents to better enable planning and design of large robot networks.

- **Large scale Representation Learning:** Parallel data collection across multiple heterogeneous agents also enables faster learning. We propose to investigate learning invariant mappings that can generalize across different environmental situations such as size, position, orientation of objects, viewpoint of the observer, *etc*. We will study how to learn the library of invariant segments (also termed as sub-goals, options or actions) hierarchically and reuse them to adapt to different timescales and robotic platforms.

Load balancing of computation from the Cloud to the Fog In order to speed up the process of learning and updating the deep learning models for control tasks such as grasping, the compute and storage would be distributed over the GDP infrastructure. We investigate the trade-offs for both continual inference and continual training on different devices (from the cloud, to smaller computers on premise, to microprocessors on the edge (*i.e.*, the robot)) to facilitate robot learning in a federated manner.

5.2 Data Modeling and Inference with Digital Twins

Digital twins will be a natural extension of the intelligent connected applications enabled by the Global Data Plane. A digital twin can be maintained lightly on the edge, while a more expressive and heavyweight computational model can reside in the cloud. These partitioned models that enable intelligent planning and execution of complex physical systems will need to balance computation and latency requirements with security and privacy. Within this scope, we will explore the following research directions:

- **Shared Learning over Confidential Data and Secure Learning**: Maintaining Digital Twins draws many parallels from the Fog Robotics research agenda. We will investigate a specific instance of transfer learning, where we examine whether there are latent representations that can be used across different physical systems. Models developed in one context may need to be applied to sensitive data streams in another context. This movement of models across trust zones requires that we protect the integrity of both the models and the queries. Due to the usage of data and information across different untrusted domains, we will investigate how to enforce requirements of confidentiality with minimal degradation of model accuracy.
- **Continual Learning and Model Maintenance**: Maintaining a Digital Twin for a physical system requires an extremely tight feedback loop. It is well known that correlated updates to expressive approximators such as deep neural networks can result in severe performance degradation and destabilized learning [46]. We will investigate techniques to enable stable robust adaptation to rapid and sudden environment changes.
- **Secure Model Planning, Inference, Serving**: Digital twins can be used for monitoring system vitals and preventing critical failures. A natural extension would be to also model both the dynamical effects on the digital twin along with the environment within which the physical system is placed. A broader model will enable robust planning and actuation, along with a variety of trade-offs to investigate. On the edge, one may want to use a smaller, more lightweight model, perhaps with lower floating point precision and decreased accuracy, in order to provide predictions within latency requirement. In the cloud, we will want to combine models to improve accuracy in exchange for heavier computation and relaxed latency requirements. The project will utilize the GDP to explore and characterize trade-offs in placing computation

5.3 Application Partitioning and Dynamic Adaptation

Both fog robotics and distributed digital twins are compute and data intensive applications. For example, to operate a robotic vehicle, data from multiple sensors (*e.g.*, LIDAR and cameras) must be combined with advanced perception and planning algorithms to render actions in real-time. Similarly, a Distributed Digital Twin may require fusing observations from multiple vehicles with a physical model of the environment to estimate future states of the world. In both cases high bandwidth data feeds must be streamed to compute intensive models and simulations and often requiring specialized compute accelerators (*e.g.*, GPUs).

How this computation is split across the cloud and the edge can have a significant impact in latency, privacy, power consumption, and cost. By moving the computation closer to the data we can reduce latency and eliminate the security vulnerabilities associated with data moving across trust zones. However, computing closer to the sensor increases the cost and power consumption of edge devices and mitigates some of the potential benefits conferred by data fusion. Alternatively, more centralized solutions allow expensive compute resources to be amortized across applications and increase the opportunity to share models and computation. As a consequence, the optimal balance of cloud and edge computation can depend heavily on the available resources and latency requirements and may change over time.

The introduction of DataCapsules addresses security challenges associated with data movement and enables computation to be moved to third party networks and even sensing devices. We plan to investigate how to optimally partition complex modeling and simulation pipelines across edge and cloud devices. This partitioning could be automatic or policy driven based on performance, QoS or privacy needs. Note that specialized hardware accelerators are often needed to enable machine learning models to reach the latency goals required to support real-time applications. However, many of these accelerators have limited support for secure enclaves. Hence, we plan to investigate hybrid methods for secure execution of neural networks with a combination of existing enclaves and partially homomorphic encryption (PHE) protocols [28, 21].

5.4 Experimental Deployment on the EdgeNet Infrastructure

We propose to investigate the properties of the GDP prototype through deployment on top of the EdgeNet infrastructure, an experimental testbed being developed by US-Ignite [70]. Several of the PIs for this proposal have a close working relationship with Rick McGeer, one of the principal developers of EdgeNet[8]; consequently, collaboration between Berkeley and US-Ignite will be mutually beneficial.

EdgeNet is a distributed edge cloud, in the family of PlanetLab [16], GENI [8], Canada's SAVI [30], Japan's JGN-X [29], Germany's G-Lab [60], and PlanetLab Europe [51]. EdgeNet is a software-only infrastructure; sites download and install the EdgeNet VM and run it on available hardware, reducing cost and maintenance burden. This software-only architecture enables dense deployment of nodes in edge environments, allowing cooperating organizations to provide EdgeNet regions with low diameter (*i.e.*, low latency to an EdgeNet node from anywhere in the region). EdgeNet utilizes Kubernetes [35] to deploy Docker containers [20, 45, 9] across the wide area, thereby serving as an ideal experimental testbed for this project.

With EdgeNet we propose to investigate several aspects of the Global Data Plane: its ability to use untrusted nodes for transport, its scalability and overall robustness under churn, and its ability to provide a global substrate for data applications. We discuss each of these in turn: First, EdgeNet is designed to use whatever hardware a site is willing to donate on a part- or full-time basis. Most of these sites will not support secure enclaves, although some might. The GDP accommodates untrusted nodes by not decrypting data so the worst that an attacker gets is encrypted data or the ability to observe updates. Second, EdgeNet is designed to function with nodes that join and leave the network; we will investigate the behavior of GDP routers and routing state under normal EdgeNet churn as well as adaptation of multicast-building algorithms. Third, we will support the development of data-intensive applications and services on EdgeNet, including (1) custom deployments of EdgeNet nodes in a laboratory or industrial setting as a base for a Fog Robotics deployment and (2) a PlanetFlow-like monitoring infrastructure for EdgeNet built on top of the GDP. As an added benefit of this collaboration, Berkeley and US Ignite will partner to make the GDP on EdgeNet rapidly-deployable for Smart Cities applications developed by US Ignite Smart Gigabit Communities

[8] EdgeNet is being developed in part through NSF Eager #1820901 with Rick McGeer as PI (and Justin Cappos of NYU as co-PI). See http://edge-net.org/

References Cited

[1] Fog-robotics - a distributed cloud for robots. 2018. (in press).

[2] A fog-robotics approach to distributed robot surface decluttering. 2018. (in press).

[3] Bengt Ahlgren, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, and Borje Ohlman. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7), 2012.

[4] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. Innovative Technology for CPU Based Attestation and Sealing. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, August 2013.

[5] Andreas M. Antonopoulos. *Mastering Bitcoin: Unlocking Digital Crypto-Currencies*. O'Reilly Media, Inc., 1st edition, 2014.

[6] Noah Apthorpe, Dillon Reisman, and Nick Feamster. A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic. *arXiv preprint arXiv:1705.06805*, 2017.

[7] Sergei Arnautov, Bohdan Trach, Franz Gregor, Thomas Knauth, Andre Martin, Christian Priebe, Joshua Lind, Divya Muthukumaran, Dan O'Keeffe, Mark L. Stillwell, David Goltzsche, David Eyers, Rüdiger Kapitza, Peter Pietzuch, and Christof Fetzer. SCONE: Secure Linux Containers with Intel SGX. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, November 2016.

[8] Mark Berman, Jeffrey S Chase, Lawrence Landweber, Akihiro Nakao, Max Ott, Dipankar Raychaudhuri, Robert Ricci, and Ivan Seskar. Geni: A federated testbed for innovative network experiments. *Computer Networks*, 61:5–23, 2014.

[9] David Bernstein. Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, 1(3):81–84, 2014.

[10] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 13–16, New York, NY, USA, 2012. ACM.

[11] Kirill Borodulin, Gleb Radchenko, Aleksandr Shestakov, Leonid Sokolinsky, Andrey Tchernykh, and Radu Prodan. Towards digital twins cloud platform: Microservices and computational workflows to rule a smart factory. In *Proceedings of the10th International Conference on Utility and Cloud Computing*, UCC '17, pages 209–210, New York, NY, USA, 2017. ACM.

[12] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, pages 173–186, Berkeley, CA, USA, 1999. USENIX Association.

[13] Scott Chacon and Ben Straub. *Pro Git, 2nd Edition*. Apress, 2014.

[14] Tushar Chandra, Robert Griesemer, and Joshua Redstone. Paxos made live – an engineering perspective. In *Proceedings of the Symposium on Principles of Distributed Computing*, PODC, 2007.

[15] Stuart Cheshire and Marc Krochmal. Multicast DNS. RFC 6762, February 2013.

[16] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.

[17] Byung-Gon Chun, Petros Maniatis, Scott Shenker, and John Kubiatowicz. Attested append-only memory: Making adversaries stick to their word. In *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*, SOSP '07, pages 189–204, New York, NY, USA, 2007. ACM.

[18] Christian Dannewitz. Netinf: An information-centric design for the future internet. In *Proc. 3rd GI/ITG KuVS Workshop on The Future Internet*, pages 1–3, 2009.

[19] A.V. Dastjerdi, H. Gupta, R.N. Calheiros, S.K. aGhosh, and R. Buyya. Chapter 4 - fog computing: principles, architectures, and applications. In Rajkumar Buyya and Amir Vahid Dastjerdi, editors, *Internet of Things*, pages 61 – 75. Morgan Kaufmann, 2016.

[20] Docker. https://www.docker.com/.

[21] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. Technical report, February 2016.

[22] Alex Drozhzhin. Internet of Crappy Things. http://blog.kaspersky.com/internet-of-crappy-things/, 2015.

[23] Branden Ghena, William Beyer, Allen Hillaker, Jonathan Pevarnek, and J Alex Halderman. Green lights forever: analyzing the security of traffic infrastructure. In *8th USENIX Workshop on Offensive Technologies (WOOT 14)*, 2014.

[24] Ali Ghodsi, Scott Shenker, Teemu Koponen, Ankit Singla, Barath Raghavan, and James Wilcox. Information-centric networking: seeing the forest for the trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, page 1. ACM, 2011.

[25] Andy Greenberg. Hackers Remotely Kill a Jeep on the Highway, With Me in It. http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/, July 2015. [Online; accessed 27-March-2016].

[26] Tim Greene. Biggest data breaches of 2015. https://www.networkworld.com/article/3011103/security/biggest-data-breaches-of-2015.html, 2015. [Online; accessed 11-May-2018].

[27] Seena Gressin. The equifax data breach: What to do. *Federal Trade Commission, September*, 8, 2017.

[28] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Cryptodl: Deep neural networks over encrypted data. *CoRR*, abs/1711.05189, 2017.

[29] What's JGN-X? http://www.jgn.nict.go.jp/jgn-x_archive/english/info/what-is-jgn-x.html.

[30] Joon-Myung Kang, Hadi Bannazadeh, and Alberto Leon-Garcia. Savi testbed: Control and management of converged virtual ict resources. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 664–667. IEEE, 2013.

[31] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg. A survey of research on cloud robotics and automation. *IEEE Transactions on Automation Science and Engineering*, 12(2):398–409, April 2015.

[32] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18(3):263–297, August 2000.

[33] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A data-oriented (and beyond) network architecture. In *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '07, pages 181–192, New York, NY, USA, 2007. ACM.

[34] Sanjay Krishnan, Animesh Garg, Sachin Patil, Colin Lea, Gregory Hager, Pieter Abbeel, and Ken Goldberg. Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning. *The International Journal of Robotics Research*, 36(13-14):1595–1618, 2017.

[35] Kubernetes: Production-grade container orchestration. https://kubernetes.io/.

[36] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishan Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, et al. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS, November 2000.

[37] Leslie Lamport. The part-time parliment. *ACM Transactions on Computer Systems*, 16(2):133–169, May 1998.

[38] John Leyden. Samsung smart fridge leaves Gmail logins open to attack. `http://www.theregister.co.uk/2015/08/24/smart_fridge_security_fubar/`, August 2015. [Online; accessed 27-March-2016].

[39] Jinyang Li, Jeremy Stribling, Robert Morris, and M Frans Kaashoek. Bandwidth-efficient management of dht routing tables. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation*, NSDI, January 2005.

[40] Jon Loelinger and Matthew McCullough. *Version Control with Git, 2nd Edition*. O'Reilly Media, August 2012.

[41] Peter Lucas, Joe Ballay, and Mickey McManus. *Trillions: Thriving in the emerging information ecology*. John Wiley & Sons, 2012.

[42] Jeffrey Mahler and Ken Goldberg. Learning deep policies for robot bin picking by simulating robust grasping sequences. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 515–524. PMLR, 13–15 Nov 2017.

[43] Jeffrey Mahler, Matthew Matl, Xinyu Liu, Albert Li, David Gealy, and Ken Goldberg. Dex-net 3.0: Computing robust robot suction grasp targets in point clouds using a new analytic model and deep learning. *arXiv preprint arXiv:1709.06670*, 2017.

[44] Yuya Maruyama, Shinpei Kato, and Takuya Azumi. Exploring the performance of ros2. In *Proceedings of the 13th International Conference on Embedded Software*, EMSOFT '16, pages 5:1–5:10, New York, NY, USA, 2016. ACM.

[45] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239):2, 2014.

[46] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[47] Nitesh Mor, Ben Zhang, John Kolb, Douglas S Chan, Nikhil Goyal, Nicholas Sun, Ken Lutz, Eric Allman, John Wawrzynek, Edward A Lee, and John Kubiatowicz. Toward a global data infrastructure. *IEEE Internet Computing*, 20(3):54–62, 2016.

[48] Robert Morris, Eddie Kohler, John Jannotti, and M. Frans Kaashoek. The click modular router. In *Proceedings of the Seventeenth ACM Symposium on Operating Systems Principles*, SOSP '99, pages 217–231, New York, NY, USA, 1999. ACM.

[49] Willima B. Norton. *Internet Peering Playbook 2014*. DrPeering Press, January 2014.

[50] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, USENIX ATC'14, pages 305–320, Berkeley, CA, USA, 2014. USENIX Association.

[51] PlanetLab Europe. `https://www.planet-lab.eu/`.

[52] J. M. Porup. "Internet of Things" security is hilariously broken and getting worse. http://arstechnica.com/security/2016/01/how-to-search-the-internet-of-things-for-photos-of-sleeping-babies/, 2016. [Online; accessed 27-March-2016].

[53] Morgan Quigley, Ken Conley, Brian P Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, volume 3, January 2009.

[54] Morgan Quigley, Brian Gerkey, and William D. Smart. *Programming Robots with ROS*. O'Reilly Media, December 2015.

[55] Rahim Rahmani and Theo Kanter. Layering the Internet-of-Things with Multicasting in Flow-sensors for Internet-of-services. *International Journal of Multimedia and Ubiquitous Engineering*, 10(12):37–52, 2015.

[56] Eric Rescorla. HTTP Over TLS. RFC 2818, May 2000.

[57] Eric Rescorla and Tim Dierks. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, August 2008.

[58] Eric Rescorla and Nagendra Modadugu. Datagram Transport Layer Security Version 1.2. RFC 6347, January 2012.

[59] Sean C Rhea, Patrick R Eaton, Dennis Geels, Hakim Weatherspoon, Ben Y Zhao, and John Kubiatowicz. Pond: The OceanStore Prototype. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, FAST, March 2003.

[60] Dennis Schwerdel, Bernd Reuther, Thomas Zinner, Paul Müller, and Phouc Tran-Gia. Future internet research and experimentation: The g-lab approach. *Computer Networks*, 61:102–117, 2014.

[61] O. Shamir and N. Srebro. Distributed stochastic optimization and learning. In *2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 850–857, 2014.

[62] GE Power Digital Solutions. GE Digital Twin Analytic Engine for the Digital Power Plant, October 2016.

[63] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on ssh. In *USENIX Security Symposium*, volume 2001, 2001.

[64] Tony Speakman, Jon Crowcroft, Jim Gemmell, Dino Farinacci, Steven Lin, Dan Leshchiner, Mike Luby, Todd L. Montgomery, Luigi Rizzo, Alex Tweedly, Nidhi Bhaskar, Richard Edmonstone, Rajitha Sumanasekera, and Lorenzo Vicisano. PGM Reliable Transport Protocol Specification. RFC 3208, December 2001.

[65] Daniel Steinberg and Stuart Cheshire. *Zero Configuraton Networking: The Definitive Guide*. O'Reilly Media, December 2005.

[66] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: A scalable peero-to-peer lookup protocol for internet applications. *Transactions on Networking*, 11(1), February 2003.

[67] A.K. Tanwani and S. Calinon. Learning robot manipulation tasks with task-parameterized semitied hidden semi-markov model. *Robotics and Automation Letters, IEEE*, 1(1):235–242, 2016.

[68] Trevor Timm. The government just admitted it will use smart home devices for spying. http://www.theguardian.com/commentisfree/2016/feb/09/internet-of-things-smart-devices-spying-surveillance-us-government, February 2016. [Online; accessed 27-March-2016].

[69] Slavica Tomovic, Kenji Yoshigoe, Ivo Maljevic, and Igor Radusinovic. Software-defined fog network architecture for iot. *Wireless Personal Communications*, 92(1):181–196, Jan 2017.

[70] US Ignite. `https://www.us-ignite.org/about/what-is-us-ignite/`.

[71] Shanhe Yi, Cheng Li, and Qun Li. A survey of fog computing: Concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data*, Mobidata '15, pages 37–42, New York, NY, USA, 2015. ACM.

[72] Shanhe Yi, Zhengrui Qin, and Qun Li. Security and privacy issues of fog computing: A survey. In Kuai Xu and Haojin Zhu, editors, *Wireless Algorithms, Systems, and Applications*, pages 685–695, Cham, 2015. Springer International Publishing.

[73] Ben Zhang, Nitesh Mor, John Kolb, Douglas S Chan, Ken Lutz, Eric Allman, John Wawrzynek, Edward Lee, and John Kubiatowicz. The cloud is not enough: saving iot from the cloud. In *7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15)*, 2015.

[74] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. Named data networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73, July 2014.