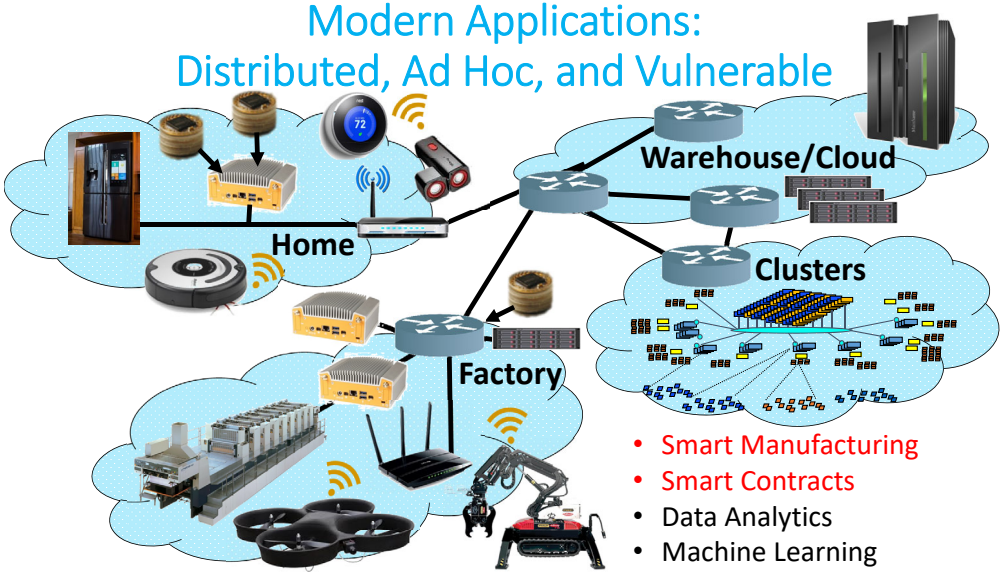
Changing the World with Cryptographically Hardened DataCapsules

Talk for HKN General Meeting
March 6th, 2019

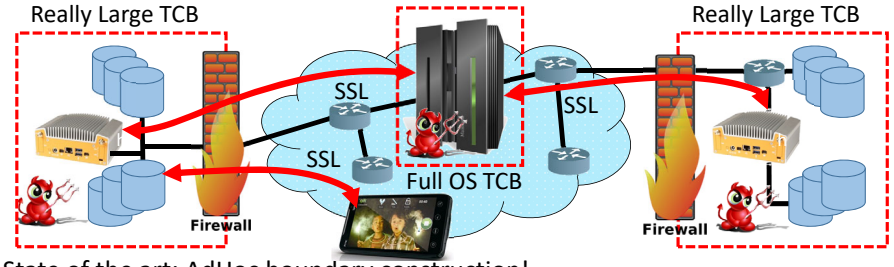
John Kubiatowicz
UC Berkeley SwarmLab/RiseLab

Modern Applications: Distributed, Ad Hoc, and Vulnerable



- Smart Manufacturing
- Smart Contracts
- Data Analytics
- Machine Learning

Why are Data Breaches so Frequent?



- State of the art: AdHoc boundary construction!
  - Protection mechanisms are all "roll-your-own" and different for each application
  - Use of encrypted channels to "tunnel" across untrusted domains
- Data is typically protected at the *Border* rather than *Inherently*
  - Large Trusted Computing Base (TCB): huge amount of code must be correct to protect data
  - Make it through the border (firewall, OS, VM, container, etc…) data compromised!
- What about data integrity and provenance?
  - Any bits inserted into "secure" environment get trusted as authentic ⇒ manufacturing faults or human injury or exposure of sensitive information

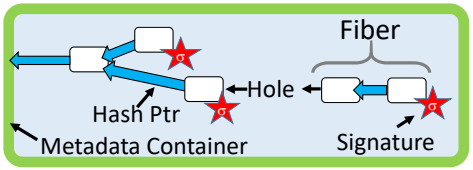
On The Importance of Data Integrity

- In July (2015), a team of researchers took *total control* of a Jeep SUV *remotely*
- They exploited a firmware update vulnerability and hijacked the vehicle over the Sprint cellular network
- They could make it speed up, slow down and even veer off the road

- Machine-to-Machine (M2M) communication has reached a dangerous tipping point
  - Cyber Physical Systems use models and behaviors that from elsewhere
  - Firmware, safety protocols, navigation systems, recommendations, …
  - IoT (whatever it is) is everywhere
- Do *you* know where your data came from? *PROVENANCE*
- Do you know that it is ordered properly? *INTEGRITY*
- *The rise of Fake Data!*
  - *Much worse than Fake News…*
  - *Corrupt the data, make the system behave very badly*

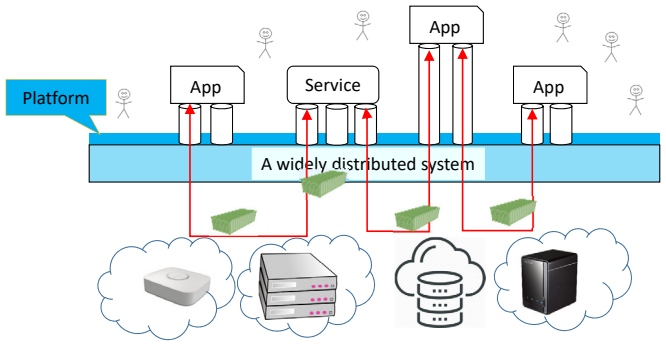
The Data-Centric Vision: Cryptographically Hardened Data Containers

- Inspiration: Shipping Containers
  - Invented in 1956. Changed everything!
  - Ships, trains, trucks, cranes handle *standardized format containers*
  - *Each container has a unique ID*
  - *Can ship (and store) anything*
- *Can we use this idea to help?*



- DataCapsule (DC):
  - Standardized metadata wrapped around opaque data transactions
  - Uniquely named and globally findable
  - Every transaction explicitly sequenced in a hash-chain history
  - Provenance enforced through signatures
- Underlying infrastructure assists and improves performance
  - Anyone can verify validity, membership, and sequencing of transactions (like blockchain)

A Platform Approach: the Utility-Provider Model [DataCapsule version of Ships, Trains, Trucks, and Cranes]



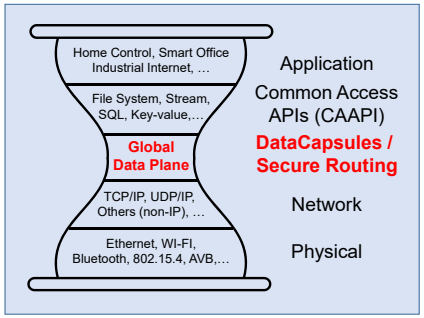
Platform Users:
Applications/Services

Global Data Plane:
Routing, Multicast,
Trust Domains, Accounting

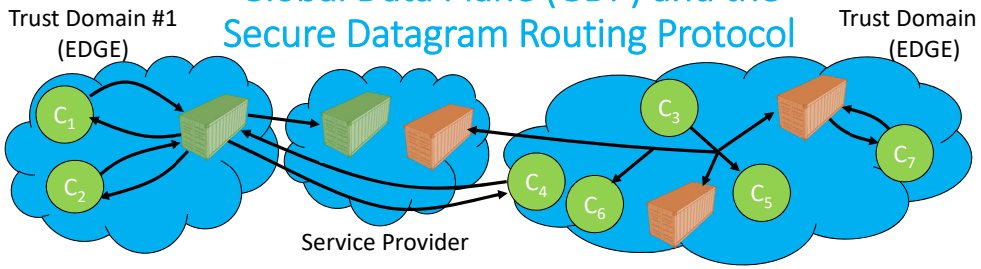
Utility Providers:
Heterogeneous Infrastructure
Storage, Transport, QoS

Refactoring of Applications around Security, Integrity, and Provenance of Information

- Goal: A thin Standardized entity that can be easily adopted and have immediate impact
  - Can be embedded in edge environments
  - Can be exploited in the cloud
  - Natural adjunct to Secure Enclaves for computation
- DataCapsules ⇒ bottom-half of a blockchain?
  - Or a GIT-style version history
  - Simplest mode: a secure log of information
  - Universal unique name ⇒ permanent reference
- Applications writers think in terms of traditional storage access patterns:
  - File Systems, Data Bases, Key-Value stores
  - Called Common Access APIs (CAAPIs)
  - DataCapsules are always the *Ground Truth*

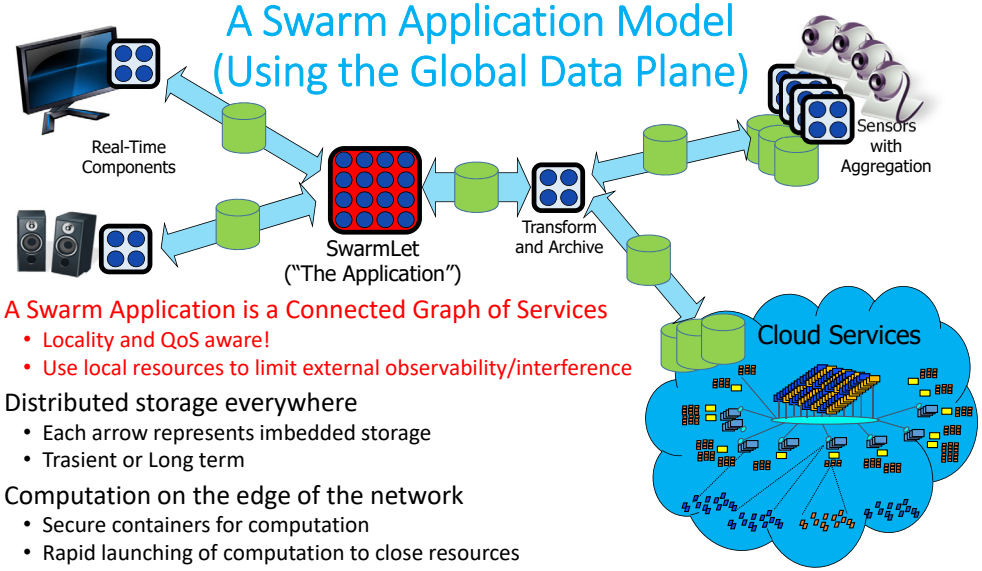


Global Data Plane (GDP) and the Secure Datagram Routing Protocol



- Flat Address Space Routing
  - Route queries to DCs by names, independent of location (e.g. no IP)
  - DCs move, network deals with it
  - Short-term Channels ("μ-SSL channels")
- Black Hole Elimination
  - Only servers authorized by owner of DC may advertise DC service
- Routing only through domains you trust!
  - Secure Delegated Flat Address Routing
- Secure Multicast Protocol
  - Only clients/DC storage servers with proper (delegation) certificates may join
- Queries (messages) are *Fibers*
  - Self-verifying chunks of DataCapsules
  - Writes include appropriate credentials
  - Reads include proofs of membership
- Incremental deployment as an overlay
  - Prototype tunneling protocol ("GDPinUDP")
  - Federated infrastructure w/routing certificates

A Swarm Application Model (Using the Global Data Plane)



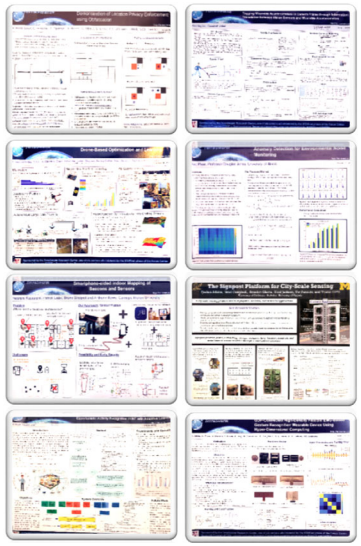
- A Swarm Application is a Connected Graph of Services
  - Locality and QoS aware!
  - Use local resources to limit external observability/interference
- Distributed storage everywhere
  - Each arrow represents imbedded storage
  - Trasient or Long term
- Computation on the edge of the network
  - Secure containers for computation
  - Rapid launching of computation to close resources

GDP in TerraSwarm

- Validation of the platform concept with a 5-year multi-university project.
- GDP as a middleware glue with uniform interface for diverse applications.
- Real people building real applications.
  - Nine (9) different research groups built interesting IoT-style applications using the GDP
- Signpost deployments, BWRC server room, Swarm Lab, and many more.
- Use of GDP as/for:
  - Data storage repository
  - Data exchange platform
  - Data visualization
  - Real-time control
- Tiny sensors to real-time video streams

Significant individual effort in high-level design, software, language ports, connectors/gateways, applications, user support, infrastructure maintenance, sensor integration, …

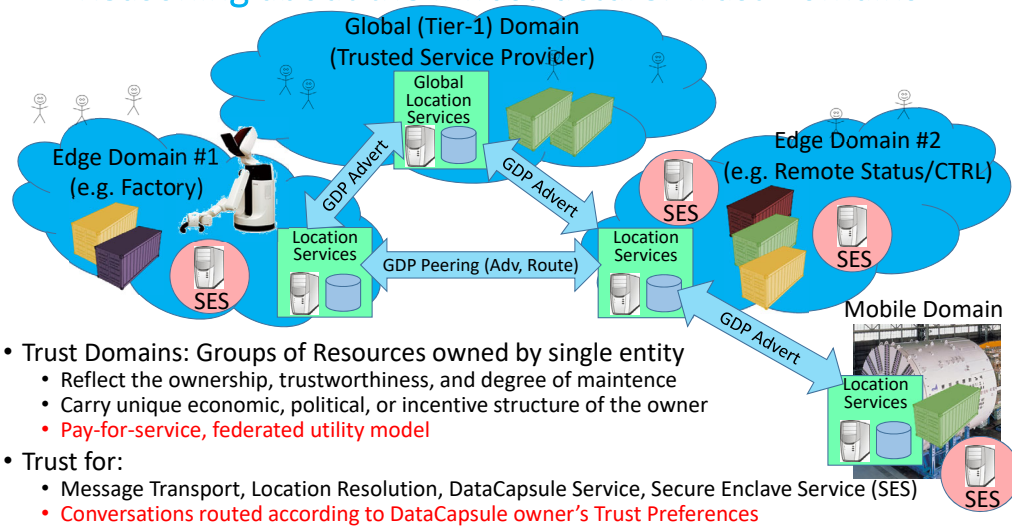
Projects using GDP: TerraSwarm posters



Why the Global Data Plane (GDP) ?

- Yes, you *could:*
  - *Provide your own infrastructure for everything*
  - *Provide your own storage servers*
  - *Provide your own networking, location resolvers, intermediate rendezvous points*
- But: *Why?*
  - Standardization is what made the IP infrastructure so powerful
  - Utilize 3rd-party infrastructure owned (and constantly improved) by others
  - Sharing is much harder with stovepiped solutions!
- The Global Data Plane provides *standardized infrastructure support*
  - It provides a standardized substrate for secure flat routing and publish-subscribe multicast
  - It provides a provides the ability to reason about infrastructure providers (Trust Domains)
  - It frees DataCapsules from being tied to a particular physical location
  - ⇒ Analogous to ships, planes, trains, and cranes that support shipping containers
- The GDP routes conversations between endpoints such as DataCapsules, sensors, actuators, services, clients, etc.
- Information protected in DataCapsules, but freed from physical limitations by the GDP
  - Correctness and Provenance *enforced* by DataCapsules
  - Performance, QoS, and Delegation of Trust handled by the GDP

Reasoning about the infrastructure: Trust Domains

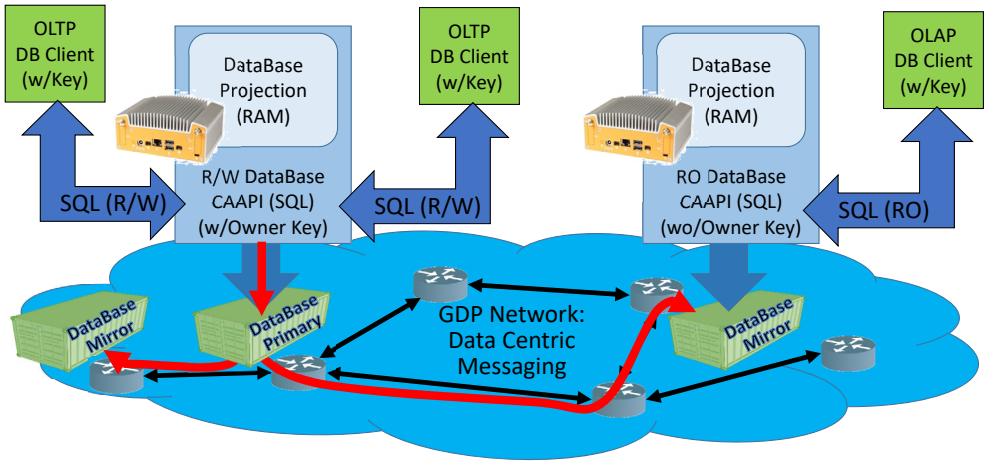


- Trust Domains: Groups of Resources owned by single entity
  - Reflect the ownership, trustworthiness, and degree of maintence
  - Carry unique economic, political, or incentive structure of the owner
  - Pay-for-service, federated utility model
- Trust for:
  - Message Transport, Location Resolution, DataCapsule Service, Secure Enclave Service (SES)
  - Conversations routed according to DataCapsule owner's Trust Preferences

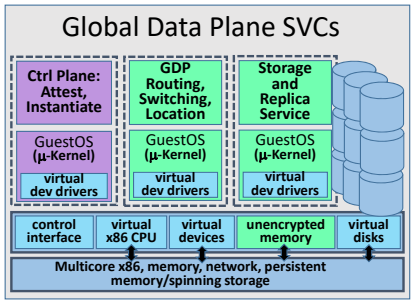
Common Access APIs (CAAPIs)

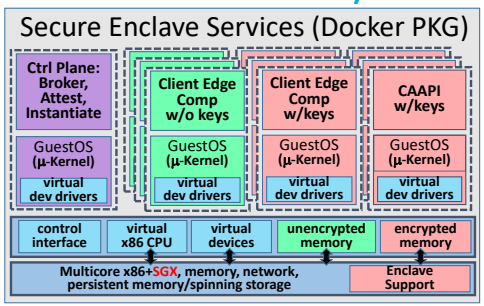
- Common Access APIs (CAAPIs) provide convenient/familiar *Storage Access Patterns*:
  - Random File access, Indexing, SQL queries, Latest value for given Key, etc
  - Optional Checkpoints for quick restart/cloning
  - Refactoring: CAAPIs are services or libraries running in trusted or secured computing environments on top of DataCapsule infrastructure
- Many Consistency Models possible
  - DataCapsules are "Conflict-free Replicated Data Types" (CRDTs): Synchronization via Union
  - Single-Writer CAAPIs prevent branches if sufficient stable storage (strong consistency models)
  - DataCapsules with branches: like GIT or Amazon Dynamo (write always, reader handles branches)
  - CAAPIs can support anything from weak consistency to serializability
- CAAPIs currently available to GDP users:
  - Streaming storage
  - Key/Value store with time-travel
  - Filesystem (changeable sequences of bytes organized in hierarchy)
- More sophisticated CAAPIs in planning stages:
  - Multi-writer storage using Paxos or RAFT
  - Byzantine agreement with threshold admission to DataCapsules
  - Bitcoin-like Admission scripting could form bottom-half of blockchain

Example Using DataCapsules to build more sophisticated data access patterns (e.g. DataBase)



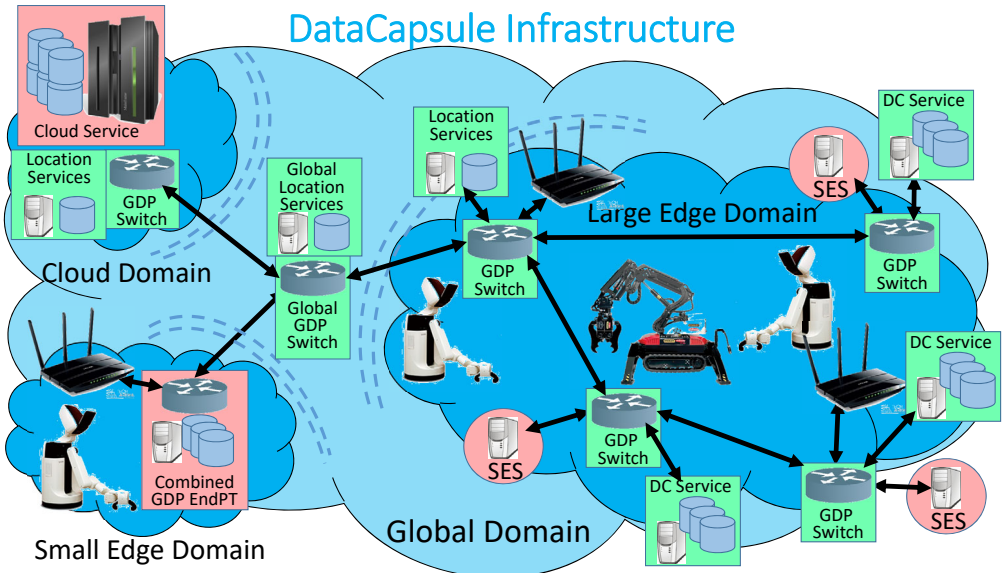
How to make DataCapsule Vision a Reality?



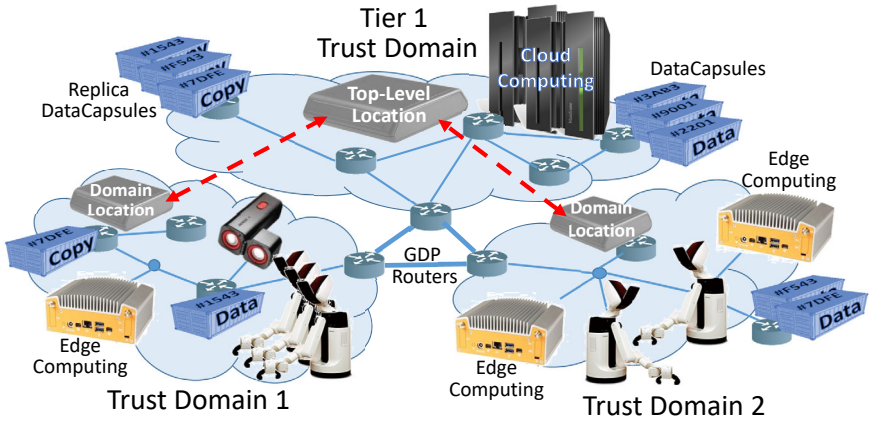


- Active Routing/Switching Components
  - Federated/Utility storage infrastructure
  - Edge-local support for multicast
  - Data Location Services
- Owned by service provider (trust domain)
  - Secure boot/validated code in DataCapsule
  - Multiple providers may own equipment in single physical environment
- Multi-Tenant Secure Computation Services
  - Secure Enclaves on Demand with specified attributes (e.g. GPU, special accelerator, etc.)
  - Standardized packaging (e.g. Docket)
  - Trustable computation through attestation, key exchange, resistance to physical attacks
- Computation is *fungible:*
  - Executable and state stored in DataCapsules!

DataCapsule Infrastructure



Fog Robotics on the Global Data Plane: A Brand-New Project SwarmLab/RiseLab/Robotics

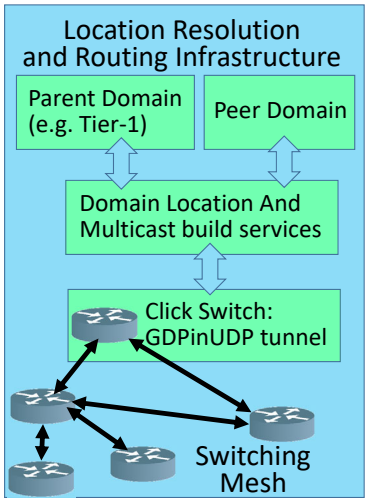


Research Agenda: What is Hard?

- Biggest Challenge: Convince People to Refactor their applications around DataCapsules
  - Incremental Deployment encouraged via (1) overlay networking followed by (2) "native" GDP datagram routing – possibly even without IP service
  - CAAPIs provide standardized storage "patterns" for naïve and domain application writers
- DataCapsules provide extremely flexible storage (intended as a primitive element upon which to build a wide array of storage systems)
  - The trick is to provide understandable semantics with good performance
  - Consider wide range of Google storage systems (GFS, BigTable, Megastore, Spanner…)!
- DataCapsule placement: Edge vs Cloud
  - Placement based on Performance, Privacy Constraints, Durability Requirements, BW, QoS, ….
- Replication and Failover semantics
  - Basic Replication simple since DataCapsules are CRDTs (Conflict-Free Replicated Datatypes). Thus, synchronization is via union of DataCapsules is easy
  - Providing quick adaptation in (routing) network as DataCapsule servers fail and recover while still providing understandable semantics is tricky
- Replication in the presence of network partitions and malicious agents
  - Can provide multi-writer storage using Paxos or RAFT
  - Can use Byzantine agreement with threshold admission to DataCapsules

Research Agenda (con't): What is Hard?

- Flat Address Space Routing is Dead, long live Flat Address Space Routing
  - No physical hierarchy in the names of DataCapsules
  - Each advertising certificate (Delegated Flat Name) is unforgeable (RO) and easily exported using a scalable DHT
  - Using Redis key-value store for initial prototype
- Adaptable, Authenticated, Automatic Multicast construction
  - Multicast is an old topic, but secure, performant, multicast that respects trust domains is essential to DataCapsule/GDP
  - Can leverage ideas from prior Bayeux multicast DHT work
- Only Active Conversations Stored in Switches!
  - Provides hope of scalability, but challenge of routing
- QoS-Aware Routing problem: Efficiently routing while respecting QoS and exploiting hardware (e.g., TSN)
  - Can leverage ideas from prior Brocade landmark overlay DHT work



What does short- and long-term success look like?

- Short-term success: DataCapsules in IoT
  - TerraSwarm IoT applications: DataCapsules+GDP were the Great Integrator
  - For low-power devices: simple gateway talks to devices with BlueTooth, ZigBee, WiFi and speaks GDP protocols to infrastructure
  - Potentially: Every camera, video, sensor, produces DataCapsules, thus enforcing provenance and consistency of output data!
- Short-term success: working distributed applications and that could be experimentally tested under a variety of conditions and threats
  - Fog Robotics: Robots using and generating models at the edge
  - Smart manufacturing in which a 3D additive fabrication facility (at the edge) is communicating securely with a digital twin demonstrating secure and timely exchange of data between the facility and the cloud.
- Long-term success: widespread usage
  - Adoption of DataCapsules and the GDP as the underpinning for many devices and applications both on the edge and in the cloud.
  - Example: Adoption of DataCapsules as new standard for IoT devices and applications
  - Similar to the widespread use of shipping containers in ports today

Conclusion

- The most game-changing element of this agenda is the presence of ubiquitous, secure and mobile bundles of data: DataCapsules
  - Provably authentic and self-consistent
  - Only authorized writers can add information; anyone with possession can verify integrity
- The power of DataCapsules are in *standardization*
  - If everyone uses DataCapsules, then everyone reaps the benefits—
    No malicious information, no fake news, no breached passwords
  - Eliminate rampant "roll-your-own" philosophy that yields data breaches
- Naturally Coupled with Secure Edge Computing (Enclaves)
- Burden of standardization reduced through careful design:
  - Incremental, flat-address-space routing (no IP addresses!)
  - Efficient refactoring of communication around storage
  - Familiar storage patterns (facades): File Systems, DataBases, Key-Value Stores, Streams,…
- Exciting new applications: Robotics and Machine Learning

Extra Slides

DataCapsules provide a better path to Secure Multi-tenant Edge Computing

- A primary concern in these environments is that information remains unaltered, durable, properly sequenced, and unobserved by others
  - Multi-stakeholder/multi-tenant systems typically "roll their own" security
  - Security is focused on preventing (virtual) hardware breaches to protect data
- DataCapsules take a different approach: Harden the data against breaches
  - Producers and consumers of shared information confident that their information retains its integrity and provenance even while transiting through shared components
- DataCapsules expose just enough meta-data to aid in managing information without requiring the information itself be exposed
  - Service providers shepherd information by automatically rejecting corrupted information and replicating as necessary
  - DataCapsule metadata provides wrapper (*and labelling*) around which to provide differentiated service, information flow, and privacy
  - Management granularity is at level of information units rather than messages (packets)
- DataCapsules coupled with Secure Enclaves provide a complete solution

Comparisons with Named Data Networking

- The GDP controls information at a coarser granularity than NDN
  - The GDP provides a standardized container (the DataCapsule) that encompasses provenance, ordering, history, and update semantics
  - This is object-level integrity/authenticity rather than packet-level integrity/authenticity
- The GDP is oriented around a federated service provider model
  - Delegation of storage, routing, durability to third parties
  - Routing only through domains of trust
- The GDP has a direct projection on secure edge computing
  - Secure Enclaves allow development of CAAPIs supporting many storage patterns
  - In principle, the DataCapsule vision provides a way for edge computing applications to delegate their secure, private, and persistent storage to providers within the network

Comparisons with Named Data Networking (Con't)

- The GDP manages information rather than simply routing packets
  - DataCapsules reside in the network where the GDP helps clients *interact with them*
  - The flat-address-space routing provides better privacy and locality by restricting interactions with DataCapsules to desired edge domains
  - The routing process is not tied to a hierarchical choice of data names, but rather to the policy-driven placement of DataCapsules
- Controlled admission to DataCapsules
  - DataCapsules have specific admission/update criteria – similar to blockchain systems
  - The NDN model provides a pull-based, caching-only model (CDN?), leaving the actual storage and update functionality outside the NDN
  - Push-based update of DataCapsules is fundamental the GDP, while it is difficult with NDN
  - Secure Publish/Subscribe is a fundamental capability of the GDP design
- Controlled replication, placement, and routing
  - The GDP seeks to control replication based on privacy requirements
  - The GDP seeks to control information flow based on trust domains
  - The GDP seeks to control level of replication such that a service provider could realistically provide durability guarantees