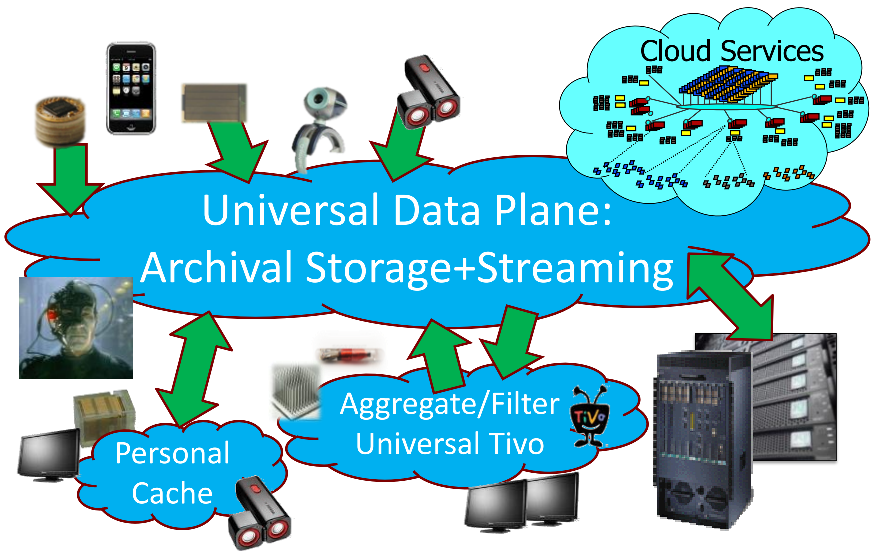
A Case for the Universal DataPlane

John Kubiatowicz and Palmer Dabbelt



I. The Problem: Reliability of Swarm Components

As researchers begin to discuss a universal integration of “Swarm” components (Sensors and Actuators) with the Cloud and “FOG” (cloud-like services at the edge of the network), they invariably encounter questions of reliability and data persistence. Typical swarm components are unreliable, have limited or intermittent energy resources, and are connected via ad-hoc communication networks that are subject to interference, packet loss, and frequent outages. At the same time Swarm components may produce or consume large volumes of data which must be time-stamped and routed efficiently in order to avoid excessive latency, communication energy, or interference with other components. Further, some of the data produced within the Swarm must be preserved for extended periods of time. Constructing useful applications using these swarm components becomes a very challenging task.

The obvious approach to building Swarm applications is to build custom stovepipe solutions which provide temporary storage at various parts of the network and which attempt to push data out to the core of the network as quickly as possible. Typically, security is ah-hoc (if it exists at all). Unfortunately, the construction of these stovepipes is time-consuming and error prone. Further, this methodology is typically counter to the desire for multi-application reuse of components.

II. The Universal Data Plane: An abstraction for Swarm construction

In this talk, we will make the argument that interposing a “Universal Data Plane” (UDPlane) abstraction between components in Swarm applications provides a clean mechanism for constructing such applications. The UDPlane is a distributed entity that persists beneath the physical components of swarm applications, as shown in the above figure. It provides location-independent naming for sources and sinks of data (called “Endpoints” for the remainder of this document). Endpoints are not permanently associated with IP addresses or other fixed routing addresses, but rather exist in a large flat namespace with sufficient identifiers (such as SHA-256 hashes). The basic abstraction is that of sending secure, immutable data “transactions” from a source Endpoint to one or more sink Endpoints. Transactions are time-stamped and (to the extent possible) securely associated with the source of the transaction via a signature. Sinks are specified by, among other things, their level of data persistence from transient to deep archival (years or decades).

A connection between a source and a sink is called a “Channel”. Communication along a Channel may occur with an arbitrary delay and with one or more levels of caching; the immutability of data transactions makes caching particularly straightforward. Channels maybe thought of as providing a secure log abstraction to Swarm components. From such a log, applications may produce a variety of data organizations, including queues, persistent files, etc. During failure, the serializing of transactions within the log becomes somewhat tricky. We will discuss several serialization options, some of which produce temporary or permanent branches in the data stream.

During communication, Endpoints are dynamically mapped to physical devices producing or consuming data. The UDPlane is responsible for routing data efficiently between sources and sinks. If the same Sink Endpoint is mapped simultaneously on multiple devices, then the UDPlane is responsible for constructing an efficient multicast tree to route data. Further, for Channels with low-latency requirements, the UDPlane will attempt to route data in a way that meets the QoS requirements of a given application; to the extent possible the UDPlane will provide an estimate of the QoS that can be expected for a given communication. Resource Discovery mechanisms compliment the UDPlane nicely, since such discovery can return the name of Endpoints with particular locality or operational properties.

The UDPlane is responsible for dynamically utilizing persistent storage within the edges of the network (such as SSDs or other low-power non-volatile components)—including flash RAM on sensors where available—to temporarily store data until it can made made more persistent by pushing it into the network. As components fail, the UDPlane will dynamically switch to utilizing other components or routes within the system, thereby providing application recovery semantics.

III. Constructing Swarm Applications with a UDPlane

As envisioned, the UDPlane is a large distributed system that shares characteristics with peer-to-peer storage systems. Potions of the UDPlane can occur on Swarm sensors and actuators; examples include hashing and signing of data, temporary caching of transactions, packet redirection (routing) as specified by other components that are optimizing routes. Use of middleware servers (the “FOG”) become an obvious place to run optimization and replication processes. Finally, the Cloud provides an obvious placed for storing copies or Reed-Solomon encoded data for long-term persistent data.

The location-independent namespace provided by the Endpoint abstraction and Channels provide an obvious mechanism for structuring Swarm applications. Data can be routed in a timely fashion between components while simultaneously archived for future use. On failure of a particular Swarm component, another may take over and assume the role of producing or consuming data without complex failover protocols.