TOWARD AN OPEN FABRIC MANAGEMENT ARCHITECTURE

Russ Herrell, Jeff Hilland

Hewlett Packard Enterprise
THE M X N FABRIC MANAGEMENT PROBLEM
where we are today

M x N combinations of tool stacks & fabrics
- Ethernet
- FC
- IB
- All different interfaces
- None covers FAM

No interoperability
THE M X N FABRIC MANAGEMENT PROBLEM
the cast of characters

The Application Developer: Creates an application suite that fulfills the Client’s needs
Relies on development tools

The Solution Provider: Integrates the application with suitable infrastructure and delivers the solution to the Client
Relies on orchestration tools

The Client: the end user of a total solution package
Relies on the access point

© OpenFabrics Alliance
THE FABRIC SW STACK
From the Client's View

The Client view
- Only sees the apps, not the stacks
- More than willing to pay for what they use. *Just not more.*
The Application Developer View

- The fabric exists only because the library exists
- The libraries exist to meet the app’s requirements
- The developer uses the library so the code is portable
- The customer intent defines the required platform and fabric characteristics
- It’s someone else’s problem to couple the resources the app needs to the right fabric at runtime

I need shared features, compute, memory, and storage

I need an MPI cluster and the libraries to program it with
The Solution Provider view

- Orchestration and Composition tools wrangle the virtual platform resources into an execution environment connected by the required fabric(s).
- Each fabric has its own management application, often not well integrated into the Orchestration and Composition tools.

A Fabric Admin is needed at runtime!
The Fabric Admin view

- Each fabric is different, and the configuration requirements are applied to many different components
- The Solution Provider / Fabric Admin needs M x N tool stacks

Need a common interface
The Open Fabrics Alliance View

• What OFA did for Application Developers
  • Created a common library and API to present fabric data plane services to applications and middleware
  • Developer needs drove library and API requirements
  • Fabric specific providers convert the standard calls into fabric specific traffic

• What OFA can do for Solution Providers
  • Create a common Fabric Management framework and interface to present fabric control plane services to orchestration and composition applications
  • Solution Provider needs drive the framework requirements
  • Fabric specific providers convert the framework calls into fabric specific management traffic
Solution Provider no longer sees low level fabric details

- The fabrics are presented to orchestration and composing applications via a standardized framework / interface
- Fabric Specific Providers translate the common function calls into fabric specific traffic to do configuration and management at runtime

What are the requirements of this framework?
Hosts
- Server / CPU / SoC / GPU / Fabric Adapter / Bridge
- Processing entity gives software threads access to fabric resources

Fabric Switches
- Packet Routing provider for the fabric
- Part of *shared infrastructure* on a multi-host, multi-tenant fabric

Endpoints (Memory, Storage, eNics, Service gateways)
- Single-tenant
- Multi-tenant
The Objective
- provide a north bound interface to describe the fabric and its resources and services (model)
- allow clients to control host, fabric, and endpoint resources (actions applied to model)
- enable fabric specific providers to program to a common set of APIs

Example functionality required in the northbound API:
- Crawl fabric, discover and enumerate resources
- Publish the resource inventory,
- service the client requests for resource configurations
- Provide methods to Enable / Disable Access
- Create / Delete Mappings of Resources to the Fabric
- Create Routes (endpoint A to endpoint B)
  - Host to endpoint
  - Endpoint to endpoint
  - Host to host
- Enable QoS, resiliency, etc. as given

Common Data Model?
Redfish is an industry standard which gives the various orchestration and automation layers a common data model for dealing with different fabrics.

- Redfish describes all fabric resources as objects
- Resource state and configuration is managed by manipulating the resource objects
- Redfish does not define fabric-level actions like enumeration or routing

Common model, common actions
WHAT IS REDFISH?

- **Industry Standard Software Defined Management for Converged, Hybrid IT defined by the DMTF**
  - RESTful interface using HTTPS in JSON format
  - Schema-backed but human-readable payload usable by GUIs, Scripts and browsers
  - Extensible, Secure, Interoperable
  - Accepted by ISO as [ISO/IEC 30115:2018](https://www.iso.org/standard/63292.html)
  - Developer hub at [redfish.dmtf.org](http://redfish.dmtf.org)

- **Initial release in 2015**
  - Additional features coming out approximately every 4 months
  - Started as secure, multi-node capable replacement for IPMI-over-LAN
  - Represent full server category: Rackmount, Blades, HPC, Racks, Future
  - Scope expanded to cover Storage, Networking, Fabrics, Datacenter Infrastructure
  - Shipping on almost every industry standard server shipped today

- **Current releases address the rest of IT infrastructure**
  - Alliances with multiple other standards bodies to define Redfish support
  - Working with [SNIA](https://www.snia.org) to cover more advanced **Storage** (Swordfish)
  - Working with [OCP & ASHRAE](https://www.ashrae.org) to cover **Facilities** (DCIM)
  - Adapt & translate YANG models to cover some level of Ethernet **Switching**
  - Work with Gen-Z & others to cover **Fabrics**
  - Work within the DMTF for internal support (MCTP/PLDM, RDE, SPDM etc.)
The Gen-Z Fabric is proposed as the first test case because it has *memory semantics* and supports disaggregated memory, storage, and compute. These features drive new requirements for the Redfish models and schema.

The Gen-Z Consortium has a work record with DMTF to do such enhancements.

- Zones embody path enablement between endpoints and fabric isolation policies
- Address pools dictate fabric ID assignment choices
- Memory Chunks embody memory regions and fine-grained access controls to multi-tenant media controllers
- Manager objects enable multiple types of manager entities, multiple manager instances for scaling and redundancy

Diagram Courtesy of DMTF.org
The OFA is contemplating the development of an “abstract fabric manager” built on the concepts of Redfish.

The intention is to use Gen-Z as a strawman target for such a fabric manager.

Similar to libfabric, such an abstract fabric manager would likely be built on a ‘framework/provider’ architecture.

### Summary
- There’s a problem
- There’s a known approach
- There’s an existing starting point
- There’s a development test case

### OFA and Gen-Z organizations have agreed to collaborate

### Next Steps
- Come to the BOF
- Refine a detailed requirements list for a ‘universal fabric manager’
- Form an OFA Working Group