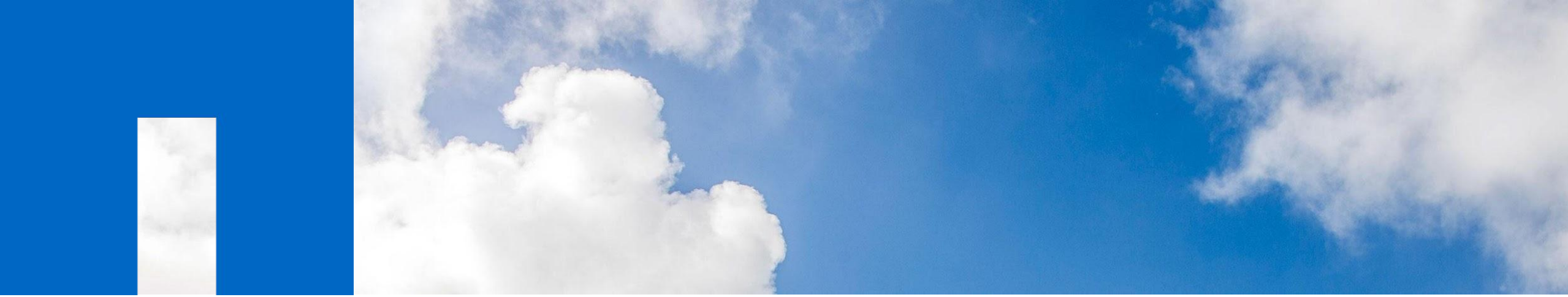


Increasing Reliability in Data Center Network Configuration

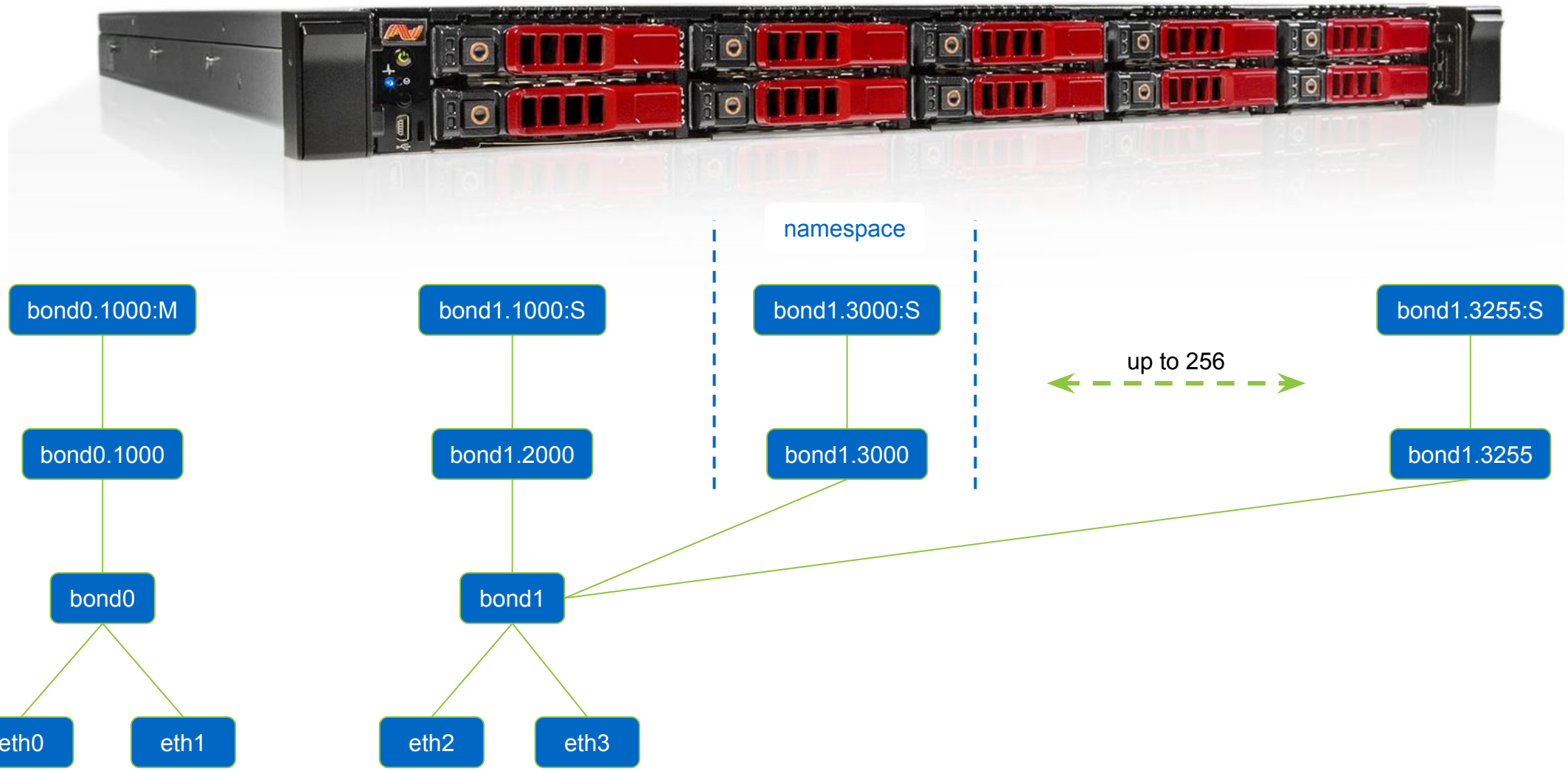
Tom Distler, Arthur Davis

NetApp

Who Are We?

- Platform team, working on SolidFire
  - Responsible for HW qualification, OS, drivers, system management, etc
  - Our focus is network configuration and management

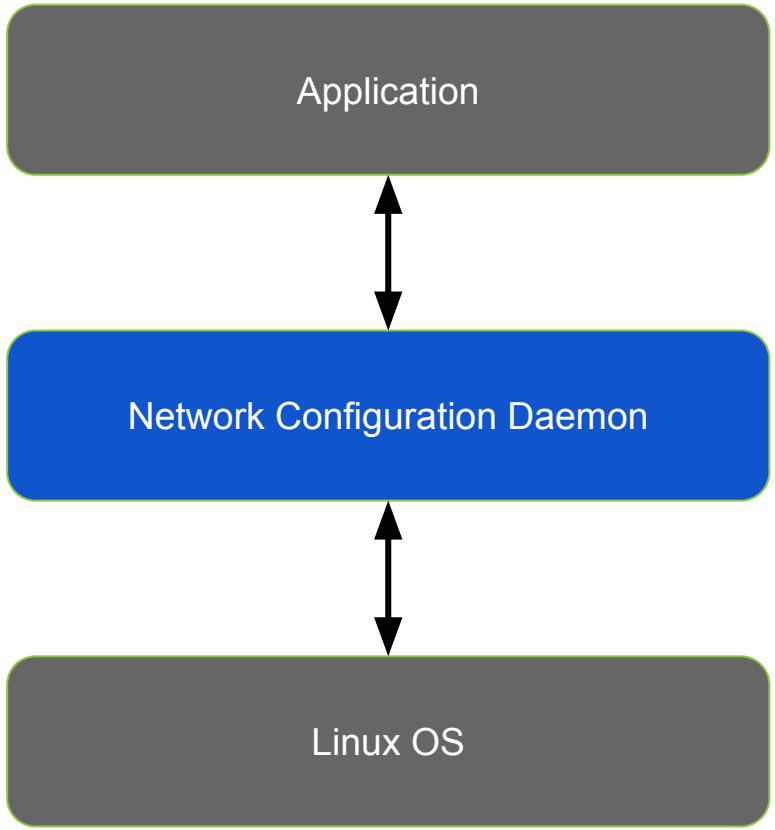
- SolidFire is a scale-out, distributed storage solution
  - Share-nothing architecture
  - Every node is an equal participant in serving storage traffic
  - Multitenancy is a key feature, especially for service providers (e.g. VLANs, VRFs, etc)
  - High-availability and predictable performance are paramount
  - Our architecture in detail: https://www.youtube.com/watch?v=AeaGCeJfNBg


namespace
bond0.1000:M
bond1.1000:S
bond1.3000:S
bond1.3255:S
up to 256
bond0.1000
bond1.2000
bond1.3000
bond1.3255
bond0
bond1
eth0
eth1
eth2
eth3

NetApp

What Are We Building?

- Network configuration daemon
  - Apply persistent config before application start.
  - Programmatic, bidirectional interface (requests in, notifications out).
  - Support multiple, concurrent actors.
  - Provide transaction semantics around config changes.
  - Monitor for network change events and notify the application of config/system mismatches.
  - Make a best-effort attempt to repair any discrepancies.



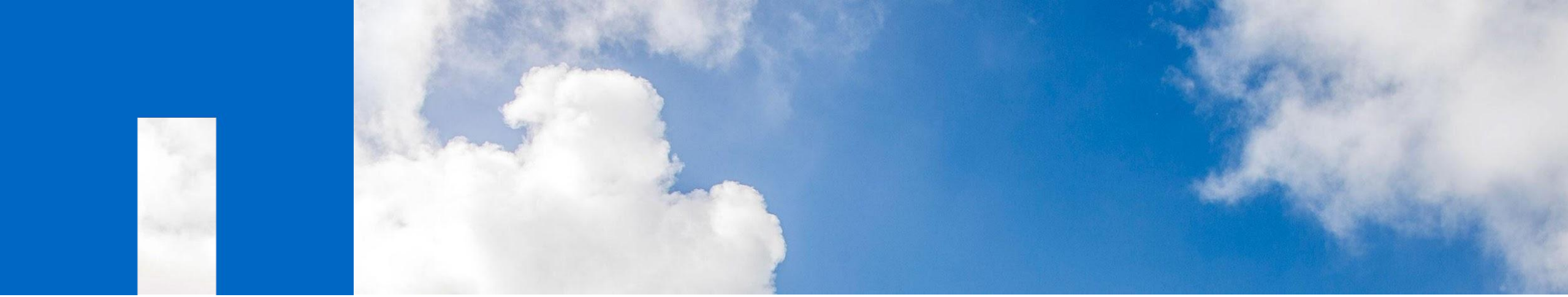
Why Are We Building It?

- What exists today doesn't facilitate the design patterns we believe are ideal for building robust networked solutions.

Application	Network Configuration Manager
Focused on product features and requirements.	Focused on the details of interacting with the network stack.
Defines the local network configuration.	Provides the implementation for enabling that configuration.
Handles errors from a product-level perspective.	Handles call-by-call, low-level errors.
Responsible for defining product-level network policy.	Does basic sanity checking on the configuration.
Reacts to network events (event-driven architecture).	Reliably and consistently detects network events (e.g. config mismatches) and notifies the application.

What's Motivating The Design?

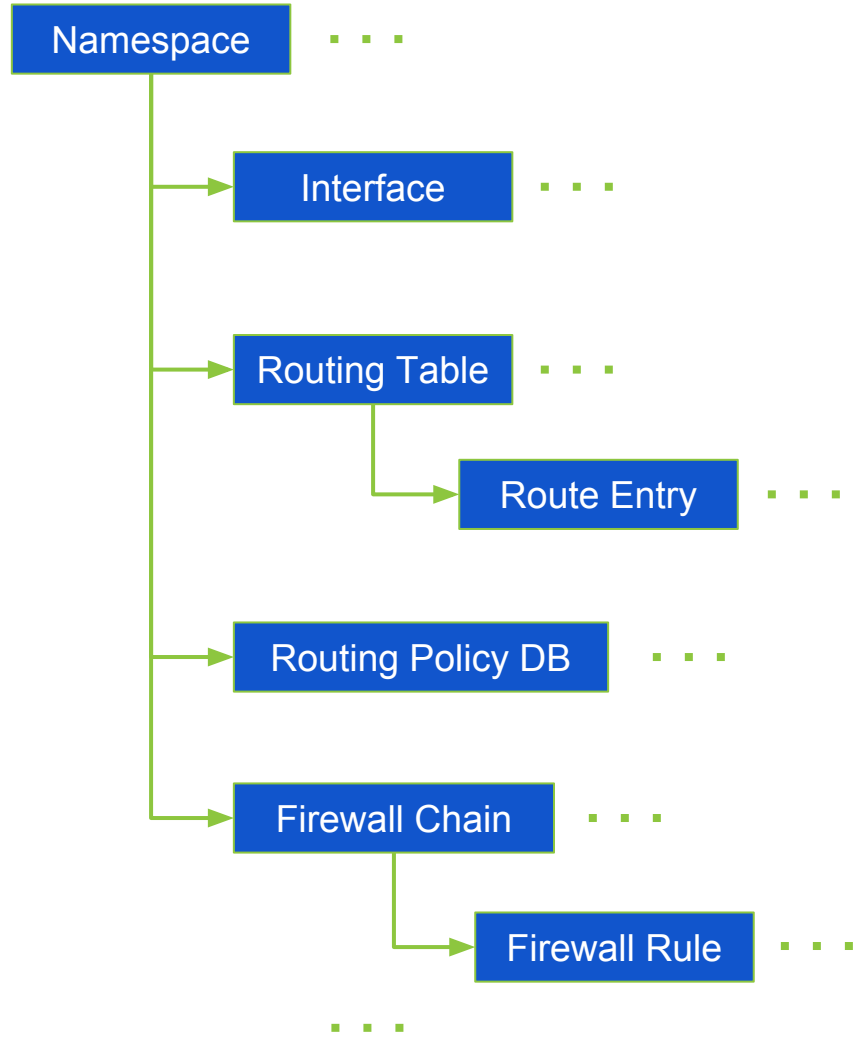
- Non-destructive network changes.
  - Even when network changes **can** be disruptive, life doesn't have to be that way…. we can do better.
- Support for concurrent actors.
  - Transaction semantics.
  - Provide ACID guarantees around configuration changes.
- Resiliency.
  - Recover from client and daemon crashes.
  - System should always boot to a consistent network state.
- Scoped management.
  - Only touch things in the network stack we are told to touch (i.e. "live and let live").
- Supportability.
  - Real-time and forensic analysis.



The Design

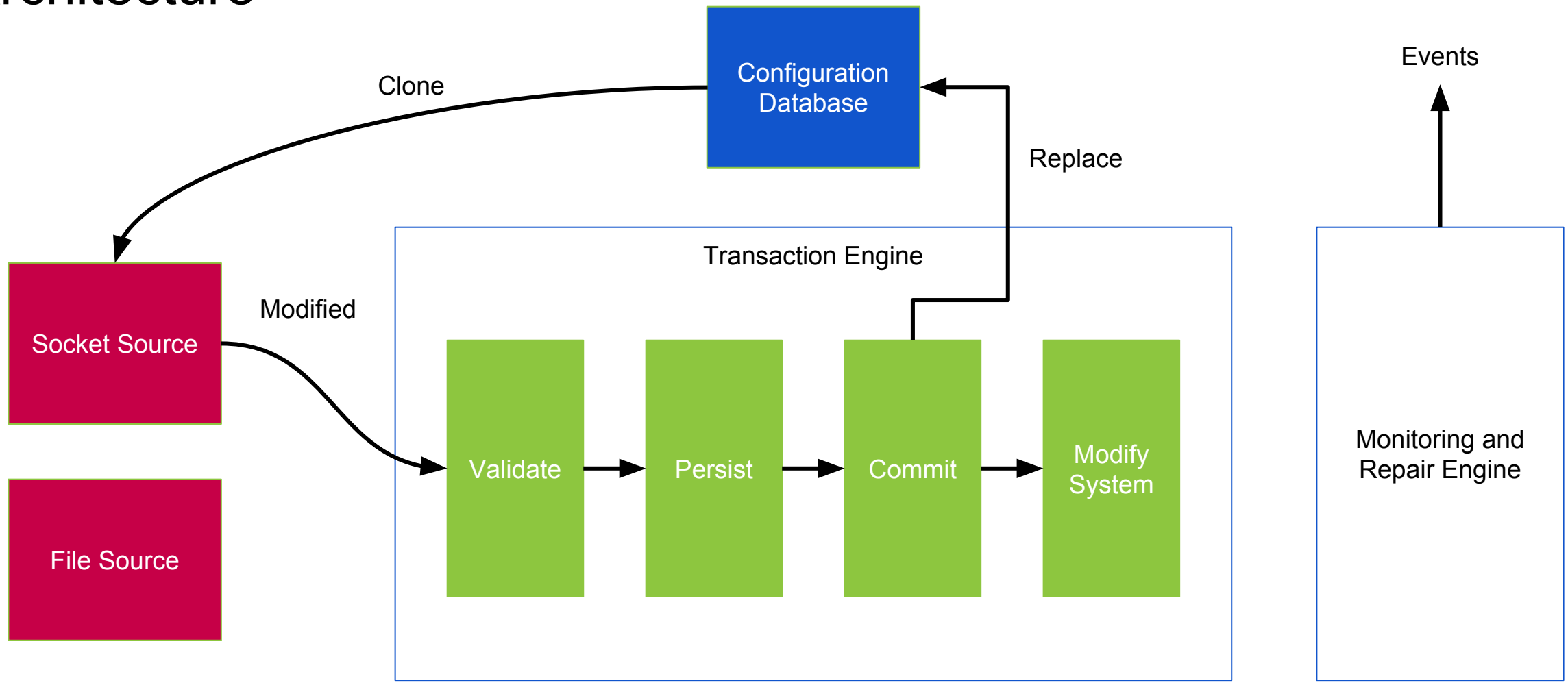
Data Model

- Reflects the hierarchy of network entities that are exposed by the kernel
- Client interactions are read-modify-write
- Our internal config database reflects this hierarchy:
  - Versioned database
  - Each object in the hierarchy is responsible for managing its respective entity in the kernel.
  - E.g. NetworkInterfaceBond encapsulates all logic for dealing with bonding.



NetApp

Architecture



NetApp

Sources

- All configuration changes originate from a source.
- Decouples communication formats and protocols from the transaction engine.
- Changes from sources processed sequentially.
- File source:
  - loads persistent config to apply on boot
  - loads internal state crash recovery if available
- Socket source:
  - Unix domain socket
  - user/group/network-namespace of client used for authorization

Transaction Engine

- Clone the current configuration when a source requests a change.
- Source applies changes to the clone.
- If the clone is modified, validate the configuration
  - The entire configuration under any modified namespace is validated.
  - Syntax (range, values, etc.) and semantic (inter-dependent objects)
- Persist the proposed config
  - Logging any changes
  - Save persistent configuration
  - Save internal state
- Make the proposed config the active config.
  - Increment the version
- First pass attempt to modify system state.

Monitor and Repair

- Background activity to compare expected configuration with running system.
- System events (e.g. netlink, udev) can trigger immediate detection of configuration problems.
- Generates notifications to registered clients.
- Attempt to reconfigure the system to repair inconsistencies.

Final Comments

- Ecosystem
  - Application support
  - Tools/utilities
  - Other integrations (CLI)
- Testing
- Open source plans

Questions / Comments?

tom.distler@netapp.com

arthur.davis@netapp.com