EOS: The Next Generation Extensible Operating System

Performance, resiliency and programmability across the entire network are now fundamental business requirements for next generation cloud and enterprise data center networks. The need for agility and deployment at scale with regards to provisioning and network operations requires a new level of automation and integration with current data center infrastructure. The underlying design of the network operating system provides the architectural foundation to meet these requirements.

Arista Networks has designed and delivered EOS (Extensible Operating System), the industry-leading, Linux-based network operating system, to address these requirements. EOS is a robust, programmable and innovative operating system, featuring a single software image that runs across the entire portfolio of Arista's award-winning network switches as well as in a virtual machine instance (vEOS). This guarantees consistent operations, workflow automation and high availability, while significantly reducing data center operational expenses.

This whitepaper discusses current network operating systems and explains how Arista's EOS architecture uniquely support next generation data centers.

Limitations of Legacy Network Operating Systems

Traditional enterprise data center networks have long been susceptible to software crashes and unplanned outages. Multiple, different software releases across switch platforms made deploying new features and services a lengthy and time-consuming process. Combined with error-prone manual configuration, inevitably network uptime was compromised.

At the core of this problem was the aging monolithic software that still runs on most of today's networking equipment. This presents a fundamental limitation to very high network reliability. The reality is that a single bug or defect anywhere in these shared-memory systems exposes the entire network to disruption, since there is no mechanism for software fault containment. At the same time, with no isolation between multiple tasks, it is difficult to scale these operating systems from a performance perspective or to reliably add new functionality. The fragile nature of this legacy software inherently prevents the extension of the network operating system to implement new capabilities such as integration with customer specific management processes or other systems and services deployed in modern data center environments.

Earlier network operating systems were incrementally built on top of customized kernels with state information arbitrarily distributed throughout the system or worse still, embedded into the kernel itself. In these environments all processes typically run in a common address space, which results in risky fate sharing – a single defect could crash the whole system and result in a severe network outage. The software state is maintained using synchronous or asynchronous polling, an inefficient mechanism that typically checks state every few milliseconds or seconds on every interface and internal data structure, resulting in wasted cycles, possible deadlocks and race conditions with multiple concurrent events.

Any changes in state are handled by fragile, carefully crafted code-paths so that software processes react in

a tightly ordered sequence depending on the underlying event. Testing all possible code-paths in such architectures results in complexity, long development and extended customer test and qualification cycles.

Additionally, the software functionality is layered with matrixed messaging occurring between processes, resulting in a maze of inter process communications and system complexity, as shown in the simplified Figure [1]:

Figure 1: Legacy Network Operating Systems Communications (circa 2000-2010)

Finally, the internal state cannot be exposed to users and applications and the only programmability that is possible in these stacks is via high-level API wrappers, which may provide the same information as SNMP or the CLI in text or XML formats for monitoring purposes.

Building cloud networks and software-defined data centers is impossible with such sub-optimal software architectures.

Arista EOS: The Modern Solution

In order to address the needs of public/private/hybrid cloud networks, while enabling new applications and ushering in the era of Software Defined Networks (SDN), Arista's Extensible Operating System (EOS) has been designed from the ground-up and is optimized for these demanding new data center environments.

Arista's Extensible Operating System is the industry's most advanced network operating system. It combines modern-day software and operating system (O/S) concepts, transparently restartable processes, open platform development, an un-modified Linux kernel, and a stateful publish/subscribe database model.

In order to fully understand the benefits a true and open operating system provides and explore the potential of a comprehensive automation solution, a detailed review of the underlying architecture of EOS is required. In the following sections we will examine the key components and attributes of EOS architecture as well as the rich array of network services built on top of EOS.

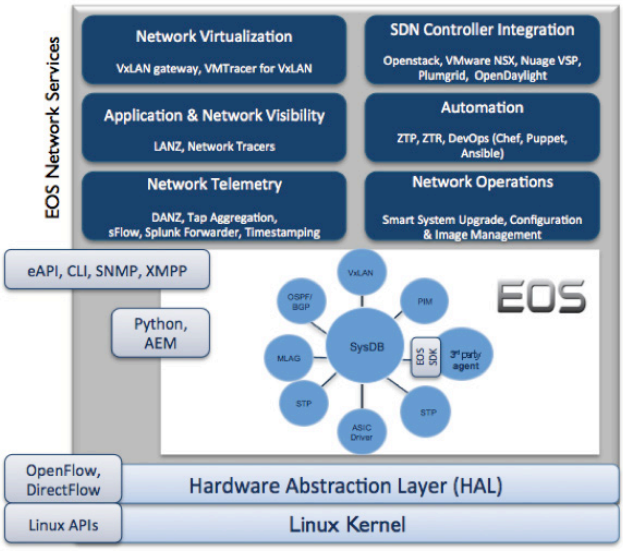

Figure 2: Arista EOS Architecture

Arista EOS: Use of Linux Kernel

At the underpinnings of Arista EOS is an unmodified Linux kernel. The main advantages of using a Linux kernel are:

- It resides in the public domain
- Hundreds of contributors, and millions of users contribute to its stability, capabilities and feature set
- It is extremely stable and modular in nature
- Availability of packages, application repositories and access to the source code
- Linux is pervasively used and therefore very widely accepted across many different systems and environments

Arista EOS: SYSDB

Separating functional control of the system into multiple processes greatly enhances resiliency and fault isolation, but requires a mechanism for coordinating actions within the system. This is the role of Sysdb.

As figure 2 shows, at the core of EOS is the system database. Sysdb is an in-memory database (machine generated at run time), which runs in user space and contains the complete state of the system. Like traditional databases, Sysdb does not contain any application logic and is only responsible for keeping state. However rather than being optimized for transactions, Sysdb is designed for synchronizing state among processes, also called 'agents', by notifying interested processes or agents when there is a change.

All agents in the system mount their configuration and status from Sysdb. This is very much like a file-system mount where read-only or read-write permissions are specified for each mount point. When an agent mounts from Sysdb, it receives its own local copy of all of the state in that mount point. As Sysdb is maintained in RAM, once the switch is turned off or restarted, information is lost.

Sysdb is like an event-driven publish/subscribe model. If the state of an agent changes, Sysdb will send an event notification to that agent, which will then update its local copy. Similarly when the agent writes to the mount point, it only changes its local copy and the write returns immediately.

Each EOS agent subscribes to Sysdb to be notified when the states of other agents change in Sysdb. When the state of an agent has been changed, this change notification is buffered and asynchronously sent to Sysdb, which then notifies all other agents who have subscribed to the changed agent.

This centralized database approach to passing state throughout the system, and the automated way the Sysdb code is generated reduces risk and error, improves software feature velocity, and provides flexibility for customers who can use the same APIs to receive notifications from Sysdb to customize and extend switch features.

Arista EOS: Key Advantages

EOS provides extremely robust and reliable data center communication services while preserving the Linux heritage of security, stability, openness, modularity, and extensibility. This combination is unique in the industry and offers the opportunity to significantly advance the functionality and evolution of next generation data center networks.

- Arista EOS is a unique multi-process state sharing architecture that separates state information from the processes itself. This reflects Arista's core software design philosophy and enables fault recovery and real- time software updates on a process-level basis without affecting the running state of the system.
- In addition, protocol processing, security functions, management services, and even device drivers run in user address space, not in the kernel itself. This greatly increases overall stability, and by maintaining the design discipline of keeping the Linux environment pure, Arista is able to easily extend the operating system with additional functionality.

- The same binary image of EOS can be deployed across any family of products. This improves the testing depth on each platform and keeps features and bug resolution compatibility across all platforms. It also makes it much simpler for users to deploy, certify and validate new releases in their data center environment.
- Most importantly, by providing users unrestricted access to the Linux shell, users can now start to leverage true data center automation features. Users are able to gain access to write Shell or Python scripts to start to leverage the network in a similar fashion other parts of the IT organization.
- Security, EOS supports authentication mechanisms such as TACACS+ and RADIUS AAA features, which allows any enterprise to fully lock down any switch for authorized & secure access.
- EOS provides a development framework, which enables the core concept of extensibility. An open foundation, and best-in-class software development models deliver feature velocity, improved uptime, easier maintenance, and a choice in tools and options.

Arista EOS: Resiliency

This section details how the EOS multi-process state-sharing architecture results in higher availability, reduced maintenance windows, improved manageability, and tighter security.

The key to high system availability is fault containment and software self-healing. On most embedded platforms, any software fault results in a reload, resulting in seconds or even minutes of downtime. Under EOS, any fault is contained within the agent or driver where the fault originated. If the fault causes the agent to crash, then the EOS process manager (ProcMgr) restarts it immediately. If the fault causes the agent to hang or loop, ProcMgr detects the condition and restarts the agent. Thus, faults within EOS are self-healing. This process is shown in Figure [3] below.

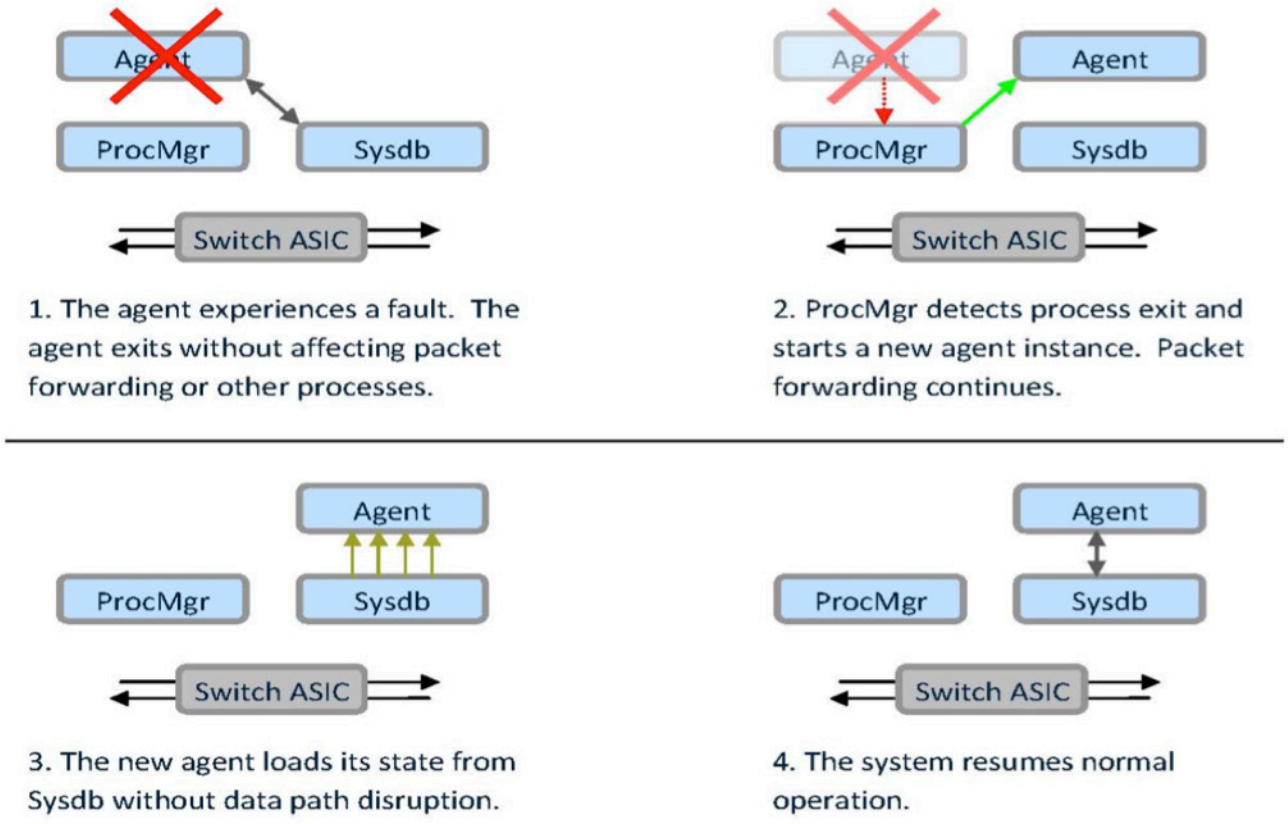

Figure 3: EOS Fault Containment and Self-Healing

The EOS multi-process state-sharing architecture is also the key to reducing maintenance windows by allowing more maintenance tasks to be performed during normal switch operation. Any EOS agent can be patched live and can be restarted without disrupting switch operation. Data flow is unaffected by the module upgrade process, so there is no user-perceptible downtime.

The key to improved operations is the capability to support third-party management and orchestration integration and task automation software. Because EOS provides a robust, protected environment for subsystems and agents, it is safe to run validated third-party agents as well, tailoring switch behavior to optimize manageability or automating common tasks within a specific customer environment. Therefore Arista EOS customers can deploy with confidence knowing that their network operating system will protect against any third party software problems.

The key to improved security is containing the impact of a security vulnerability within the vulnerable agent. For example, if the SNMP subsystem has a vulnerability, then the exploit may read all SNMP-accessible state; however, the exploit will not be able to create additional user accounts, reconfigure interfaces, or run external software. In other words, just as the EOS architecture contains faults to a single module, it also contains the impact of security vulnerabilities. Finally, through the same extensibility mechanisms that improve management and orchestration support, third-party software may implement custom security policies or intrusion detection to further enhance security.

Arista EOS: Programmability and Extensibility

EOS is programmable across all layers – Linux kernel, hardware forwarding tables, switch configuration and CLI, switch control plane as well as management layer. This programmability of EOS enables rapid integration with a wide range of third-party applications for virtualization, management, automation, orchestration and network services. Arista EOS offers a rich set of programmable interfaces including:

- Linux shell access and APIs
- OpenFlow, DirectFlow
- Sysdb APIs
- Python, Perl scripting, Advanced Event Management (AEM)
- EOS SDK
- JSON based eAPIs, CLI, SNMP , XMPP

Arista EOS has full Linux shell access for root-level administrators and makes a broad suite of Linux based tools available. OpenFlow and DirectFlow allow customers to program the forwarding state of the switch in order to fine- tune packet forwarding based on application needs. Sysdb APIs provide access to all internal state, including low- level counters, temperature measurements, power supply status and all other parameters necessary to monitor and manage the system natively. JSON-based EOS APIs (eAPI) provide easy web-based integration with tools that are commonly used to manage compute and storage resources as well as orchestration systems. Even the CLI written in Python is customizable. Scripts based on Python, Perl, etc., can also be developed as third party or native integration with virtual and physical applications, SDN-type controller platforms and Layer 4-7 services.

With EOS Software Development Kit (SDK), customers can develop their own customized EOS applications in C++ or Python. This EOS development model allows these third party applications to be first-class citizens of EOS along with other EOS agents. The SDK provides programming language bindings to software abstractions available in Arista EOS, so third party agents can access switch state and react to network events. These applications can, for example, manage interfaces, IP and MPLS routes, Access Control Lists (ACLs), as well as use a range of APIs to communicate between the switch and monitoring or network controllers. The SDK targets both long-running processes requiring event-driven notifications and scripts requiring high-performance interactions with other EOS agents. The state separation through Sysdb and the inherent fault isolation enabled by the modular architecture is what enables customers to develop and install their own applications without fear of disrupting the entire system.

This rich programmability can be leveraged by customers to build customized applications, integrate with ecosystem tools or even address feature gaps with a quick turnaround time that truly makes EOS the right network OS for modern, agile, Cloud Data Centers.

Arista EOS: Network Services

On top of the industry-leading foundation of EOS, Arista has added a comprehensive suite of network services including rich Layer2/Layer3 functionality, network virtualization support, network visibility and telemetry, automation, SDN Controller integration. All these services capitalize on the rich extensibility that EOS offers to integrate with third party tools and take a pragmatic, network-wide approach at addressing some of the customer pain points. E.g. ZTP helps automate and templatize the provisioning for large number of switches, Smart System Upgrade (SSU) helps perform non-disruptive software upgrades, CloudVision enables network wide integration with SDN controllers using OVSDB, eAPI or OpenFlow.

Arista EOS: Smart System Upgrade

One example of a network service is Smart System Upgrade (SSU). SSU delivers graceful software upgrades for the network. The ability to perform non-disruptive system software upgrades to introduce new network services or bug fixes is critical to maintaining the availability levels that data center environments demand.

Earlier network operating systems typically followed a telco-like model of in-service software upgrades (ISSU), where a single platform in production would at best be able to cut over to an incrementally higher version of code, perhaps containing a few new features or fixes. The traditional ISSU approach has historically been burdened by the significant amount of complex software development required. Special ISSU code had to be written to account for all possible checks and balances while the system attempts to maintain and convert all hardware and software state as it jumps between two different versions of software. This challenge is increased as the feature-set increases, features interact with each other, and the associated state information grows. The side effect of this complexity is that corner cases are common and it is not possible to address all scenarios. Even additional testing cycles, which further add an 'ISSU tax' to the release timeline, are not guaranteed to catch all cases

Leveraging the superior architecture of an open, programmable EOS and direct integration with other applications and infrastructure components, SSU takes a more holistic network perspective to software maintenance by allowing a network element to be transparently removed or added while traffic is either diverted or impact is altogether avoided. Designed to be a complete solution for data center infrastructure maintenance, Arista's SSU provides the following key benefits:

- Intelligent insertion and removal of network elements, customized to the spine role or the leaf role
- Programmatic upgrade to new software releases without causing systemic outages
- Open integration with all application and infrastructure elements
- Simplified solution: Intentionally avoids the complexity of heavy state maintenance and state conversion process needed with other approaches

Therefore Arista's provisioning model allows for an initial deployment or replacement of switches via Zero Touch Provisioning (ZTP) or Zero Touch Replacement (ZTR), after which the SSU framework ensures continuous operation by supporting seamless upgrades either at the Leaf or Spine layer as appropriate.

Arista vEOS: EOS as a Virtual Machine

Arista vEOS extends the EOS software platform running on the physical switches to a virtual machine-based offering. vEOS inherits all the EOS architecture attributes and benefits discussed in this paper. vEOS is an actual EOS image, not a simulator.

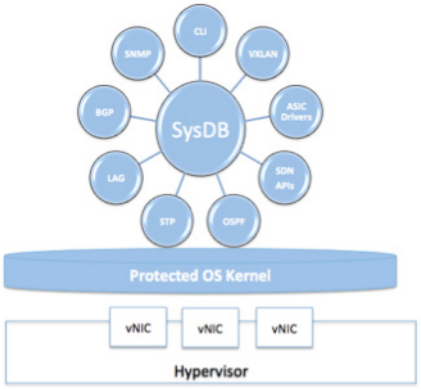

Figure 4: vEOS

vEOS can be used to build virtual network topologies and validate new EOS features and functionality. Customers leveraging the rich programmable interfaces of EOS can use vEOS for development and testing of their scripts or applications without the need for physical switches. vEOS also makes it easy for a wide array of Arista partners to access EOS to design, build and test extensions that enable best-of-breed integrations for our customers. vEOS is also an effective training tool – the simplicity and flexibility of a VM-based image provides an easy approach to familiarize with EOS and learn various EOS features, tools and troubleshooting techniques.

EOS Architecture Summary

Event-driven Architecture: All state-changes trigger a notification through Sysdb to all processes registered for that event. This allows the system to operate under intense load with great efficiency and higher resiliency.

Granular Modularity & Self-Healing Resiliency: EOS provides software fault containment and stateful fault repair of individual modules for superior system stability. EOS also allows in-service software upgrades of individual modules without any impact to application traffic.

Rich Network Features: EOS consists of rich, industry-leading Layer 2, Layer 3, Quality of Service (QoS), Access Control List (ACL), manageability, security and virtualization features.

Programmatic across all layers: EOS is programmable across all layers – Linux kernel, hardware forwarding tables, switch configuration and CLI, switch control plane as well as management layer for automation or network monitoring or integration with third party tools and orchestration systems. A wide range of programmatic interfaces from eAPI to Linux APIs or OpenFlow and DirectFlow to EOS SDK that interface with the system state can be utilized to develop custom applications, interface with third party tools or management systems or even influence switch forwarding behavior.

Extensibility: The state separation through Sysdb allows users to run their own applications directly on the Arista portfolio for native integration with Linux, and in-house tools, scripts and applications. Scripts based on Python, Perl, etc., can be developed as third party or native integration with virtual and physical applications, SDN-type Controller platforms and Layer 4-7 services.

Smart System Upgrade: The Smart Systems Upgrade (SSU) application within EOS allows for graceful insertion and removal of switches from a network, for a zero-downtime upgrade of the network by taking advantage of network-redundancy and integration with advanced ECMP and MLAG topologies.

Network Visibility: Tracer and monitoring capabilities for Big Data/Hadoop (MapReduce Tracer), Virtualization (VM Tracer), Network Path (Path Tracer), Latency Analysis (LANZ) and overall system health monitoring.

Network Telemetry: Network TAP Aggregation and Data Analysis (DANZ) allow for traffic mirroring and monitoring of the network without any impact to user-traffic. Integration with Splunk, sFlow-based collectors, and application monitoring tools such as Corvil or ExtraHop provide traffic visibility.

Network Automation: EOS natively supports Puppet, Chef, CFEngine and Ansible, which enables network configuration in the same manner as servers and storage within data center environments. In addition, EOS supports tools that greatly reduce network operational costs. For example Zero Touch Provisioning (ZTP) automates the provisioning of network infrastructure and speeds time to production for new services while eliminating the risk of human error and Zero Touch Replacement (ZTR) provides automated provisioning of replacement switches, significantly reducing mean-time-to-replacement of a failed switch.

Network Virtualization: Virtualized environments such as NSX from VMware, System Center from Microsoft, KVM with OpenStack, as well as bare-metal physical servers can all be interconnected through EOS. Overlay technologies such as VXLAN allow for a larger scale, multi-tenant network in a mixed environment containing virtual and physical network infrastructure.

vEOS: Arista vEOS extends the EOS software platform running on the physical switches to a virtual machine-based offering. It enables network design and validation, provides an easy way to develop and test extensions and applications on EOS and is an effective training tool to increase familiarity with the features and tools that EOS has to offer.

Conclusion

Arista's EOS Software is the key foundation for deploying Cloud Data Centers. It is the most advanced, resilient and programmable operating system and provides industry leading network services, operational innovations and integration capabilities.

For more information, visit: http://www.aristanetworks.com/en/products/eos

References

Linux as a Switch Operating System: Five Lessons Learned

https://eos.aristanetworks.com/2013/11/linux-as-a-switch-operating-system-five-lessons-learned/

Santa Clara—Corporate Headquarters
5453 Great America Parkway,
Santa Clara, CA 95054

Phone: +1-408-547-5500
Fax: +1-408-538-8920
Email: info@arista.com

Ireland—International Headquarters
3130 Atlantic Avenue
Westpark Business Campus
Shannon, Co. Clare
Ireland

Vancouver—R&D Office
9200 Glenlyon Pkwy, Unit 300
Burnaby, British Columbia
Canada V5J 5J8

San Francisco—R&D and Sales Office
1390 Market Street, Suite 800
San Francisco, CA 94102

India—R&D Office
Global Tech Park, Tower A & B, 11th Floor
Marathahalli Outer Ring Road
Devarabeesanahalli Village, Varthur Hobli
Bangalore, India 560103

Singapore—APAC Administrative Office
9 Temasek Boulevard
#29-01, Suntec Tower Two
Singapore 038989

Nashua—R&D Office
10 Tara Boulevard
Nashua, NH 03062