Accelerating Innovation with Lean NFV

Why did we write this document?

All new technologies have growing pains, particularly those that promise revolutionary change, so the community has waited patiently for Network Function Virtualization (NFV) solutions to mature. However, the foundational NFV white paper is now over six years old, yet the promise of NFV remains largely unfulfilled, so it is time for a frank reassessment of our past efforts. To that end, this document describes why our efforts have failed, and then describes a new approach called Lean NFV that gives VNF vendors, orchestration developers, and network operators a clear path forward towards a more interoperable and innovative future for NFV.

Where did we go wrong?

The short answer is that current NFV efforts are drowning in a quicksand of complexity that arises in several crucial aspects of NFV operations: current solutions are complex to deploy, because they are too closely coupled to how the

rest of the infrastructure is managed; they are complex to automate, because so many components must be coordinated; and the process of onboarding new virtualized network functions (VNFs) is extraordinarily time-consuming, because VNF developers were never given clear and practical guidelines for designing VNFs that can gracefully co-exist with general NFV management solutions. Moreover, these problems will not just fade away with time, as they are inherent in our current architectural approach to NFV. Thus, to make progress, we must completely rethink how we design NFV solutions. As we note later, this does not mean we have to throw out all our previous efforts; in fact, the Lean NFV approach should be seen as a complement to some of the existing open-source efforts. However, the core principles of Lean NFV, which we describe below, are very different from those we have previously followed.

What Is the alternative?

To clarify our terminology, we identify three main pieces of NFV solutions:

- **NFV manager:** This is the entity that handles common lifecycle management tasks for both individual VNFs and end-to-end NFV service chains.[1] The NFV manager is not necessarily embodied in a monolithic implementation; as we recommend later, it can be a set of interacting components that each handle one or more specific tasks.
- **The computational infrastructure:** This includes the compute resources (bare metal or virtualized) and the connectivity between them (provided by a physical or virtual fabric); the former is managed by a compute controller (e.g., Openstack) and the latter by an SDN controller.
- **Virtualized Network Functions (VNFs):** These can include both data plane and control plane components (e.g., an EPC's S/P-GW and MME). VNFs can optionally have an Elemental Management System (EMS), but only to handle VNF-specific configuration (leaving all other lifecycle management tasks to the NFV manager).

We contend that the complexity currently hindering NFV arises not from how any one of the above pieces is built, but instead from how they are woven together into an overall system. More specifically, the complexity arises when the NFV manager is integrated with the existing computational infrastructure, when VNFs are integrated with the NFV manager, and when coordination is required between the various components of the NFV manager.

Thus, we believe that rather than standardizing on all-encompassing architectures, or adopting large and complicated codebases, the NFV movement should focus exclusively on simplifying these three points of integration, leaving all other aspects of NFV designs open for innovation. To this end, we advocate adding a fourth element to the NFV solutions, a key-value (KV) store, that serves as a universal point of integration. As we argue later, this minimal approach, which we call Lean NFV, reduces complexity, eases deployment, encourages interoperability, and spurs innovation. We now describe how our proposed design addresses these three points of integration.

Integrating with the existing computational infrastructure: Infrastructure controllers can support a wide range of sophisticated features. Some current NFV efforts have exploited these capabilities to embed NFV management capabilities into existing infrastructure systems such as OpenStack. To the contrary, we strongly recommend that,

[1] Thus, the NFV manager spans the functionality provided by ETSI's NFV orchestrator and generic VNF manager (VNFM).

as a starting point, the NFV manager **only** rely on a core set of capabilities that are supported by **all** infrastructure managers, namely: provide NFV with computational resources, establish connectivity between them, and deliver packets that require NFV processing to these resources.[2] All of the remaining functionality (e.g., chaining, auto-scaling, failure handling) should be established directly by the NFV manager on the resources it has been provided without relying on additional infrastructure capabilities. Thus, the infrastructure can remain completely unaware of, and therefore undisturbed by, the presence of NFV functionality. Since only minimal infrastructure functionality is required, the Virtualized Infrastructure Manager (VIM) can be very lightweight.

We call this **plug-in integration**, because NFV solutions can be "plugged in" to any computational infrastructure. This makes deployment far easier than in current solutions, and enables innovation to proceed independently in both the computational infrastructure and the NFV manager (e.g., operators can change their SDN controller without impacting the NFV deployment, or switch to a new NFV manager without modifying their infrastructure controllers).

Integrating the various components of the NFV manager: These components need to share information, but this sharing must be done in a way that is fully extensible, vendor-agnostic, and secure (i.e., supports strong access control). Modern KV stores provide exactly these capabilities. Thus, rather than implement the NFV manager monolithically, or build it out of separate components that coordinate via tightly-coupled pairwise APIs, we strongly recommend that NFV managers be comprised of components (which we call microcontrollers) that coordinate through the use of an open-source KV store. Each microcontroller can write information into the KV store (e.g., current status or various configuration parameters) in the form of key-value pairs; these entries can be watched by other microcontrollers which may take appropriate action (e.g., launch a new VNF instance or update internal configuration) as the values change. This preserves great flexibility of implementation while enabling interoperability of microcontrollers from multiple vendors: as long as they agree on key semantics, or use separate keyspaces, they are compatible.[3]

Integrating with VNFs: On-boarding existing VNFs is very difficult, because most are merely repackaged versions of older implementations that rely on proprietary or obsolete interfaces. This situation has persisted because, despite six years of NFV development, vendors still have been given little practical guidance for how to rewrite their VNFs to make integration easier. With Lean NFV, integration merely requires VNFs to use the KV store to read configuration information and expose their operational data. The KV store thus provides a universal and scalable mechanism for bidirectional communication between NFV management systems and VNFs (and also, as noted above, between microcontrollers). Thus, the use of a KV store makes it much easier to onboard new VNFs and introduce new management functionality.

[2] Of course, an SDN controller can eventually play a more active role if desired, but our proposed starting point does not require that the SDN controller be aware of individual VNFs or chains; instead, the SDN controller is merely told to deliver NFV-bound flows to the compute nodes assigned to NFV. We expand on this point later in the paper. This simplicity helps avoid premature standardization of the features required in the computational infrastructure.

[3] For scalability, large datasets (such as telemetry or other long-running time-series data) could be stored elsewhere, with the KV store allowing components to exchange information about where these large datasets can be found.

Lean NFV

The resulting architecture is depicted in Figure 1 below. Reflecting the above design principles, the NFV manager is responsible for the management of both individual VNFs and end-to-end service chains, including lifecycle management tasks such as placement, launching, configuring, chaining, scaling, healing, monitoring, and upgrades. Note that while we show an NFV solution in a single cluster, the same principles can be applied in a hierarchical fashion for multi-site management.

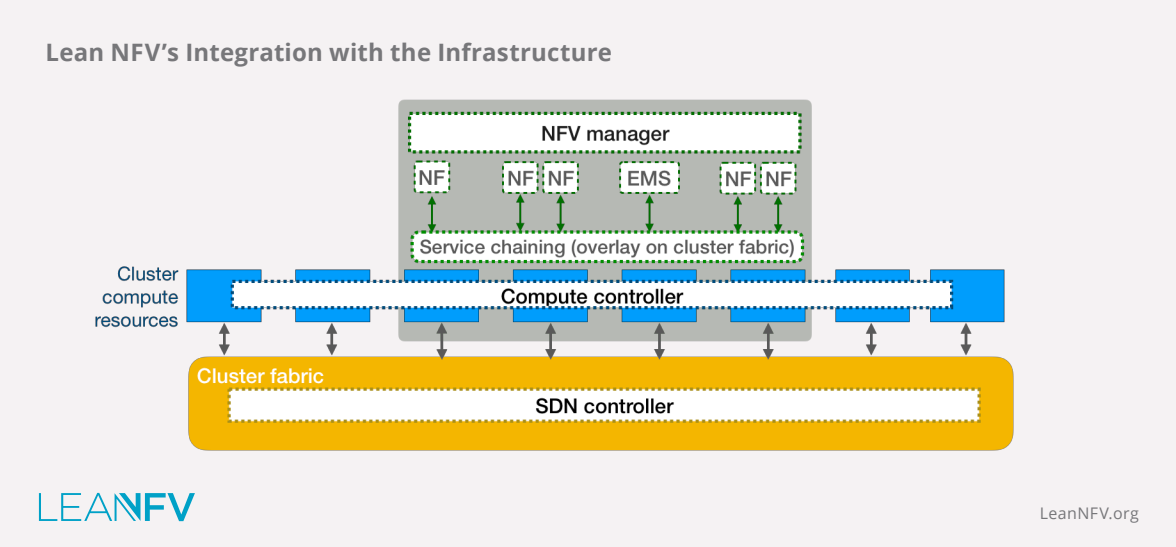

Figure 1.

All components of the NFV solution run on the infrastructure (compute resources and connectivity fabric) that are assigned to NFV processing by the infrastructure controllers as shown in Figure 2 below. The NFV manager is given a set of service chain definitions (in some form of declarative specification), and periodically reports on the status of these chains.

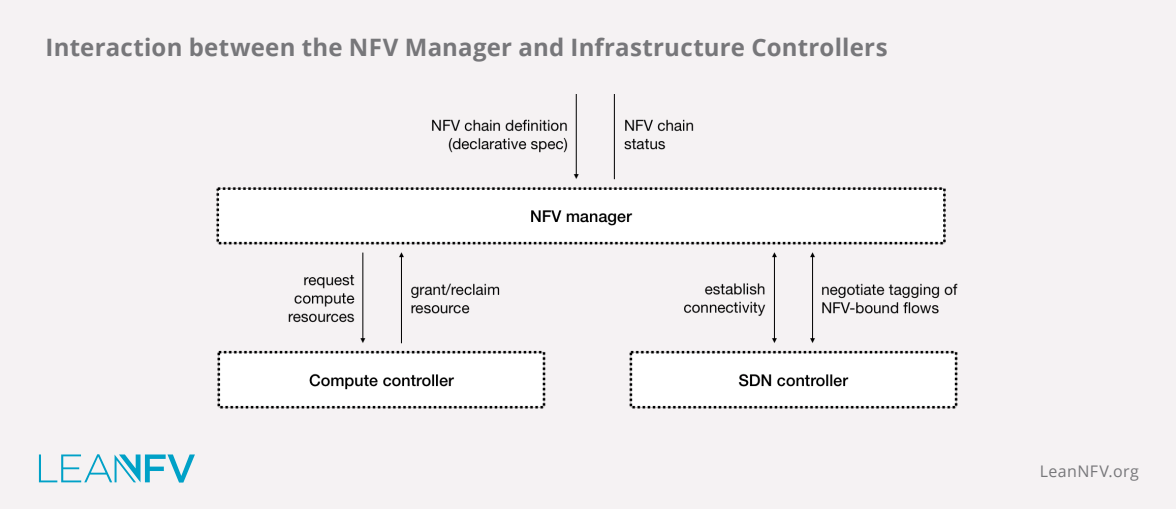

Figure 2.

Finally, as mentioned before, the NFV manager itself should be architected as a collection of microcontrollers each of which addresses one or more specific aspect of management. These microcontrollers must coordinate only through the KV store, as shown in Figure 3 below (depicting an example set of microcontrollers). This approach improves modularity and simplifies innovation, as individual management features can be easily added or replaced. The KV store also functions as the integration point for VNFs (depicted on the left) and their (optional) EMSs; in such cases, coordination through the KV store can range from very lightweight (e.g., to notify the EMS of new VNF instances or to notify the NFV manager of configuration events) to actually storing configuration data in the KV store. Thus, use of the KV store does not prevent NF-specific configuration, and it accommodates a range of EMS implementations. Note that because the KV store decouples the act of publishing information from the act of consuming it, information can be gathered from a variety of sources (e.g., performance statistics can be provided by the vswitch). This decreases reliance on vendor-specific APIs to expose KPIs, and hence allows a smooth migration from vendor- and NF-specific management towards more general forms of NFV management.

Note that this approach echoes the 3GPP notion of control and user plane separation (CUPS) in the 5G reference architecture, which also separates the publishing of information by VNFs from the consumption of information by other entities. Thus, Lean NFV already embodies the Service-Based Architecture required by 5G.

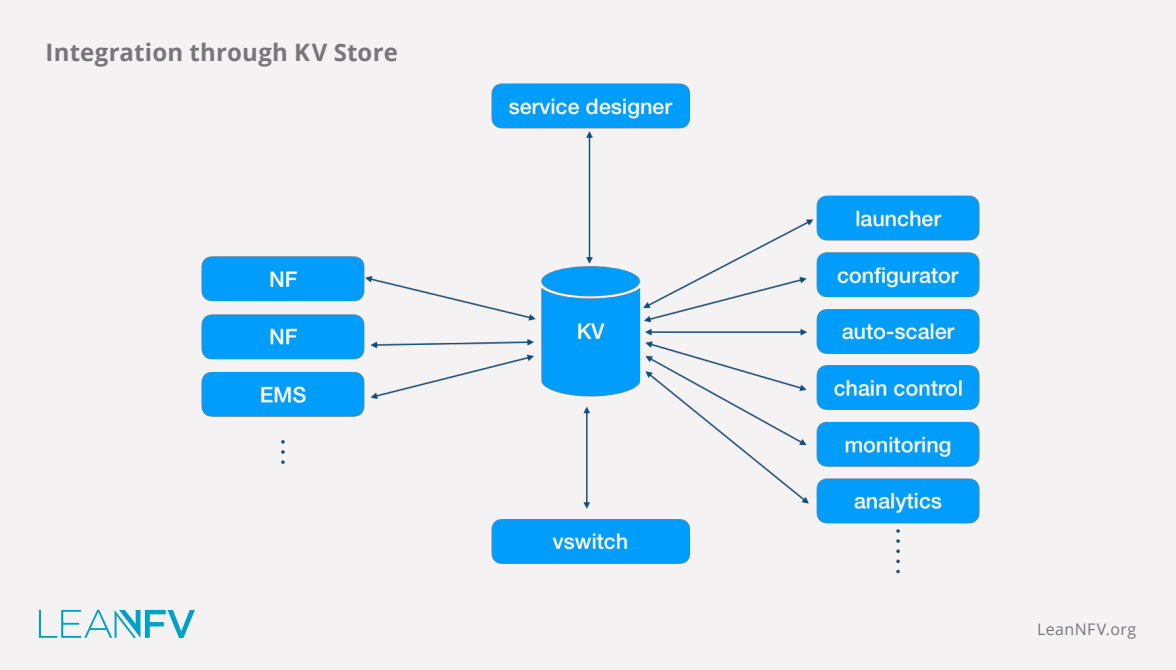

Figure 3.

Summary

The Lean NFV approach relies on two technically straightforward design decisions: using an open-source KV store for coordination within NFV solutions (management components and VNFs), and using a plug-in approach to integration with the existing infrastructure. *Our central claim is simple: by adopting these two simple measures, the community can create a new NFV ecosystem that achieves the best of both worlds: easy deployment and rapid innovation.*

In terms of deployment, Lean NFV implementations are highly portable, in that such designs can be easily inserted into any reasonable computational infrastructure (including bare metal resources) merely by plugging them in. In terms of innovation, Lean NFV enables operators to transition from specialized EMSs to using general NFV management without changing the VNF itself. It also allows NFV managers to easily add new microcontrollers (whether from the same vendor or another) without adding any new APIs.

More generally, this approach is well-suited to cloud-native designs and the proposed 5G service-based architecture, both of which envision deeply disaggregated VNFs and highly modular management. Our Lean NFV proposal aligns perfectly with this vision by providing, via the KV store, a clean way of coordinating these various components.

In contrast to previous efforts, Lean NFV is neither a monolithic open-source effort nor a highly prescriptive architectural blueprint: instead, Lean NFV is an open architecture that only specifies the minimal requirements needed for interoperability. We expect that the components of the resulting ecosystem will come from both commercial vendors and open-source efforts, and can be mixed and matched as desired by operators. To initiate this process, we are developing open-source VNFs compliant with the Lean NFV approach so that all interested parties can see operational examples of this new approach.

Since this document only described Lean NFV in general terms, we will soon be sharing with the community several additional design documents providing more details on the approach described here. These documents will address topics such as:

- **Deployment models:** For brevity, our description of Lean NFV discussed the infrastructure components only in very general terms. In future documents we will elaborate on how Lean NFV would be deployed in different infrastructure scenarios, ranging from bare-metal to containerized (e.g., with Kubernetes as an infrastructure manager) or different implementations of virtualized environments.
- **5G:** We will describe in more detail how the Lean NFV approach supports 5G's Service-Based Architecture and implements the requires slicing.
- **Relation to ONAP and OSM:** While Lean NFV is derived from different principles than ONAP and OSM, we do not propose Lean NFV as a replacement. Instead, in a forthcoming document we will describe how Lean NFV designs can complement these existing approaches.
- **Leveraging enhanced hardware capabilities:** A future document will describe ways in which the Lean NFV approach can leverage advanced hardware features or accelerators.

Moving Lean NFV Forward

The technical barriers to adopt Lean NFV are few and relatively minor. The far more challenging task is to create a community consensus around this new approach, so that vendors and operators can focus their efforts on creating this new ecosystem. We ask all interested parties to endorse this document, engage in a broader discussion of this approach, and consider providing components adhering to its design principles. Together we can move forward with Lean NFV.

Signatories

The Lean NFV approach has been endorsed by the signers below (ordered by the name of their associated establishments). They are expressing their individual opinions, which do not necessarily represent the official position of their current or former employers.

Christian Martin (Arista Networks)

Krish Prabhu (Formerly with AT&T)

Mirko Voltolini (Colt)

Nagesh Nandiraju (Comcast)

James Feger (F5)

Ying Zhang (Facebook)

Bikash Koley (Google)

Uri Elzur (Intel)

Raj Yavatkar (Juniper)

Shyamal Kumar (Lavelle Networks)

Travis Ewert (LightRiver)

Sylvia Ratnasamy (Nefeli Networks and UC Berkeley)

Scott Shenker (Nefeli Networks and UC Berkeley)

Nabil Bitar (Formerly with Nokia)

Jian Q. Li (NTT Global)

Ravi Srivatsav (Formerly with NTT)

Constantine Polychronopoulous (VMware)