BUILDING FAULT MANAGEMENT INTO NFV DEPLOYMENTS

Background and Purpose of OPNFV's Doctor Project

A Linux Foundation Collaborative Project

INTRODUCTION

Telecom services have very high requirements on service performance, often as high as the famous “five nines” or 99.999%, which equates to only five and one-half minutes of downtime per year.

As a consequence, these services must employ very stringent redundancy and high availability mechanisms. To give an example of the impact of downtime, any service interruption in (for instance) Evolved Packet Core (EPC) nodes could disconnect thousands of subscribers from the mobile network.

Carriers are migrating from utilizing specialized equipment to Network Functions Virtualization (NFV)-based networks where the same rules would apply. As they run applications in a virtualized environment, with software decoupled from standard hardware, they need a new fault management and maintenance framework to achieve these requirements. The Doctor project in OPNFV[1] was created to provide this support.

Doctor creates an open reference platform that features immediate notification of a wide range of failure events from the NFV Infrastructure (NFVI)[2], and supporting orchestration for virtual network functions (VNFs) to recover. This is provided through immediate notification to the Virtualized Infrastructure Manager (VIM) when infrastructure is unavailable.

As with all OPNFV projects (and in the spirit of NFV itself), Doctor is driven by multiple vendors and service providers. The project is highly active, and collaborates with widely recognized ETSI NFV ISG[3] and upstream open source projects (e.g., OpenStack[4]).

[1] See https://wiki.opnfv.org/display/doctor.
[2] NFVI includes the environment (e.g., physical machines, hypervisors, storage and network elements) in which VNFs are deployed.
[3] The composition and functionality of all components of the ETSI NFV Functional Architecture are detailed in the NFV Architectural Framework at http://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf.
[4] See https://www.openstack.org/.

DOCTOR USE CASES

Carrier-grade high availability support may be inbuilt or provided by a platform, but the key requirement is very fast detection and reaction time to minimize service impact.

The Doctor project focuses on fault management and recovery, ensuring that applications come back to a fully redundant configuration faster than before. By focusing on supporting these capabilities in the infrastructure, Doctor maximizes the chances that NFV infrastructure will meet the stringent “five nines” requirements expected of telecommunications equipment.

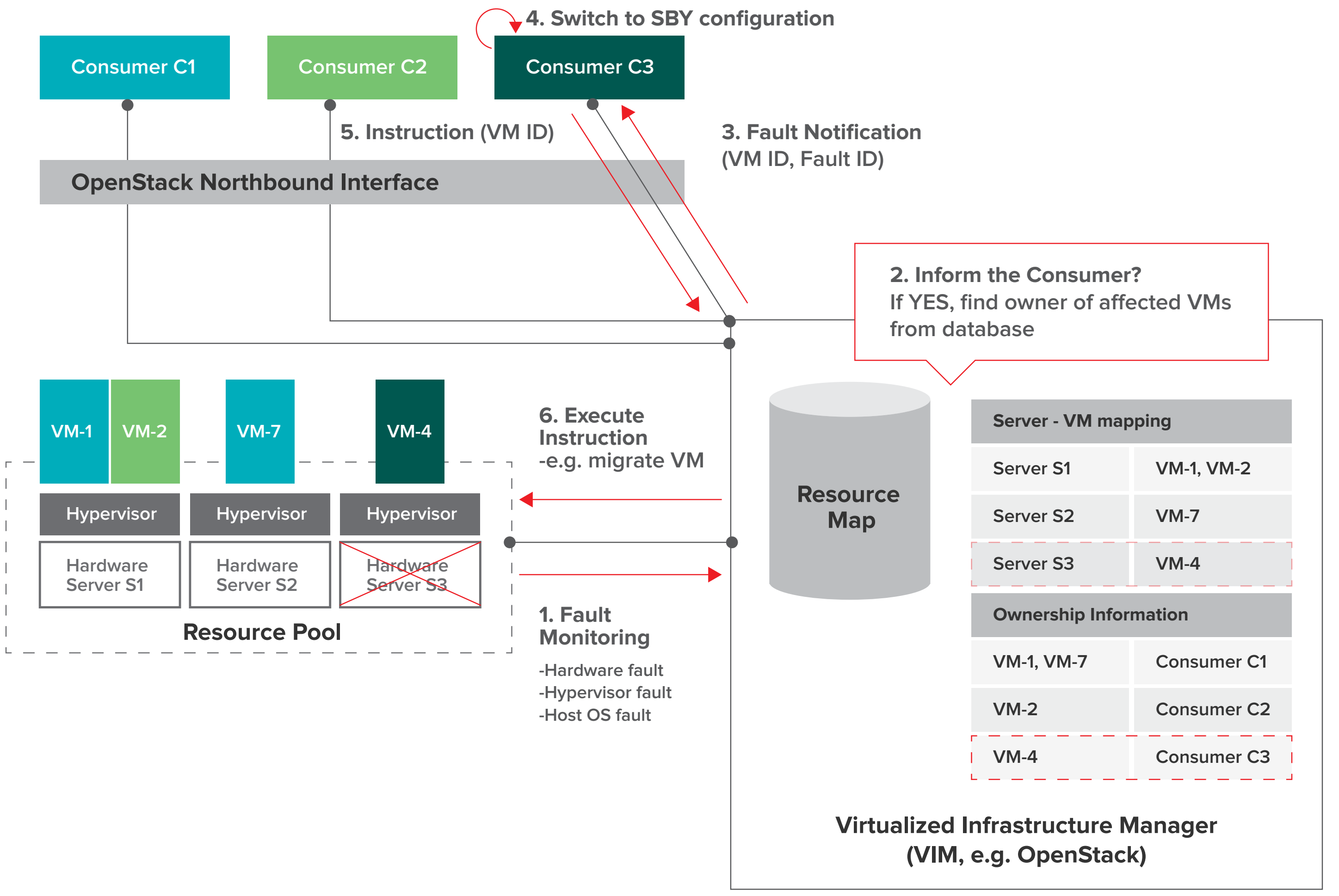
As an example, Telecom services typically come with an active-standby (sometimes abbreviated ACT-STBY) configuration which is a 1+1 (one ACT, one STBY) redundancy scheme. This is also known as a “hot standby” configuration—if an active node is unable to function properly for any reason, the standby node is instantly made ACT, and affected services incur no service interruption.

After recovery, either the previously active node is made standby, or a new standby node is configured. The actual operations to instantiate/configure a new standby node are similar to those involved in creating a new VNF, and would typically be handled by orchestration functions such as a VNF Manager (VNFM) or an NFV Orchestrator (NFVO).

In the following use cases, these Management and Orchestration (MANO) functions are referred to as “consumers” of VIM.

FAULT MANAGEMENT

The following figure presents a system-wide view of Doctor's fault management functionality. Consumers (VNFM or NFVO) manage the respective virtual resources (VMs in this example) shown with the same colors.



Server - VM mapping	
Server S1	VM-1, VM-2
Server S2	VM-7
Server S3	VM-4

Ownership Information	
VM-1, VM-7	Consumer C1
VM-2	Consumer C2
VM-4	Consumer C3

Figure 1: Doctor's Fault Management Functionality

The VIM controller detects faults in the NVFI (Step 1) affecting the proper functioning of the virtual resources (labeled as VM-x). Faults may be related to (for example) the hardware, hypervisor, or the host operating system.

At this point, the VIM locates the consumer of the affected virtual resources (Step 2), and sends a notification (Step 3). The consumer then switches the standby node to an active state (Step 4). Minimizing the reaction time to do this is a necessary basic ingredient for fast failover.

Once the consumer has switched to STBY configuration, it notifies (Step 5) the VIM. The VIM can then take necessary actions to restore the integrity and functionality of any affected VMs (Step 6).

Prediction: A Related OPNFV Project

The fault management scenario shown here may also be performed based on fault prediction, wherein a module predicts an imminent fault in the elements of NFVI. For example, the rising temperature of a physical server might trigger a pre-emptive recovery action. Requirements for this type of activity are detailed in a related OPNFV project called Prediction: “Data Collection for Failure Prediction.”

For more information, see the OPNFV Wiki page for Prediction: https://wiki.opnfv.org/display/prediction.

NVFI MAINTENANCE

All network operators perform maintenance of their network infrastructure. One of the additional Telco requirements presented by NFV is that virtualization may increase the number of elements subject to such maintenance, since NFVI holds new elements like the hypervisor and host OS.

For maintenance, the VNF running on the target server machines either needs to be paused or migrated. Pausing or migrating an active VNF will interrupt the services being provided by the VNF. In order to avoid this during maintenance, the correspondence consumer (e.g., the VNFM) needs to be notified so that it can switch to the standby VNF; it may also need to create a new standby as well to maintain the active-standby configuration, thus avoiding any service interruption. The VIM can then “empty” (clear the VNF on) those servers.

Once the target hardware servers are emptied (i.e., no virtual resources are running on top), the VIM can mark them with an appropriate flag (e.g., “maintenance” state) so that these servers are not considered for hosting of virtual machines until the maintenance flag is cleared (i.e., nodes are back in “normal” status).

A high-level view of this procedure is presented in Figure 2. The VIM receives a maintenance notification (Step 1) from a network operator, including information about which compute resources are subject to maintenance (i.e., replacement or upgrade of hardware or a hypervisor).

The VIM then determines which virtual machines are affected by this change (Step 2), and notifies the respective consumer (VNFM or NFVO) in Step 3. Based on this, the consumer takes necessary actions (Step 4), switching to STBY or switching VNF forwarding graphs, and notifies the VIM (Step 5).

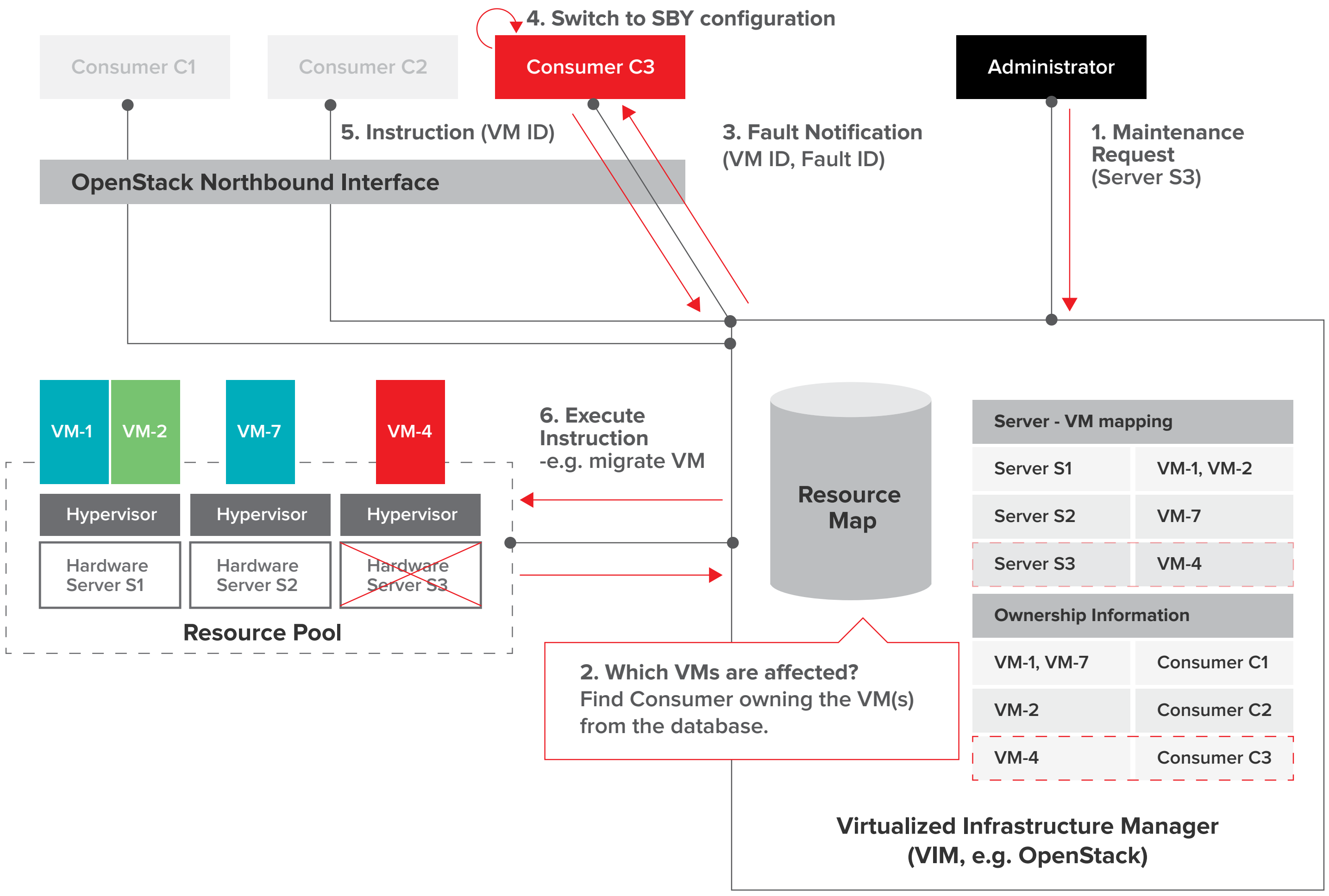


Figure 2: Doctor's Maintenance Functionality

Upon notification, the VIM empties the hardware (Step 6) so that consequent maintenance operations could be performed.

STANDARDS AND UPSTREAM PROJECTS

Fault management is not native to OpenStack, so this support must be built into the NFVI (for instance) for Evolved Packet Core (EPC) VNFs such as mobility management entities (MME) or service/packet gateways (S/P-GW) to switch from active to standby mode as soon as relevant failures are detected.

For this reason, a gap analysis between relevant upstream projects (e.g., OpenStack) and Doctor's requirements is continually performed, and Doctor contributes accordingly to the identified project. Some examples of this are shown in the Appendix.

On the standardization side, Doctor is having an impact on fault management topics in ETSI NFV, namely in the areas of synchronization and alignment with Interfaces and Architecture (IFA) and Reliability (REL) working groups.

CONCLUSION
MILESTONES, UPSTREAM CONTRIBUTIONS, AND DEMONSTRATION

The Wiki page for Doctor lists the key project facts, including a list of the major contributors.

The Doctor project provides the necessary functionality and gap analysis for fault management within upstream projects to ensure that this support is available. Another major use case for this functionality is maintenance of the NFV infrastructure.

APPENDIX

NFV infrastructure based on virtualized environments using standard servers cannot natively meet carrier-grade availability expectations that were previously handled by specialized hardware.

The Doctor project provides the necessary functionality and gap analysis for fault management within upstream projects to ensure that this support is available. Another major use case for this functionality is maintenance of the NFV infrastructure.

Milestones and Goals

Since its creation on December 2014, the project has accomplished some important milestones:

- Doubled membership to 15 company members (5 Telco operators, 10 vendors and IT)
- Proposed, implemented and had 3 OpenStack blueprints accepted on a total of 13 code contributions in two projects (Aodh, Nova)

Doctor's goals for 2016 are to continue fostering openness and collaboration opportunities among all OPNFV projects and members, as well as upstream open source communities and standardization bodies.

For the upcoming OPNFV Colorado release, users can expect more features in various OpenStack projects (e.g., Nova, Congress, and Neutron) contributed by the Doctor project, addressing additional and much-needed fault management and maintenance requirements. Users can also expect greater functional testing scenarios coverage, and improved installation support and documentation.

The Doctor team is also involved in collaboration planning with other existing OPNFV projects (e.g., the Software Fastpath Quality Metrics project - SFQM[6]) aiming to build up an integrated platform for NFV. These activities will demonstrate open collaboration across OPNFV and upstream projects.

[6] See https://wiki.opnfv.org/display/fastpath.

Upstream Contributions

In the Doctor project, the team has developed failure event collection and immediate notification features in OpenStack Liberty (released in October 2015). Anyone looking for immediate alarming can now leverage these contributions submitted and accepted in OpenStack.

There is an *OPNFV-OpenStack Community Page*[6] on the OPNFV Wiki that is jointly maintained by OPNFV and OpenStack community members. This page, which is updated regularly, describes community engagement practices to manage joint contributions between the two open source communities.

There are many items listed for the Doctor project; the following table contains the ones that are completed.

Table 1: Examples of Upstream Contributions from the Doctor Project

Project	Blueprint	Specification	Developer
Aodh	Event Alarm Evaluator	Ryota Mibu (NEC)	Ryota Mibu (NEC)
Nova	API to mark nova-compute down	Tomi Juvonen (Nokia)	Roman Dobosz (Intel)
Nova	Support forcing service down	Tomi Juvonen (Nokia)	Carlos Goncalves (NEC)
Nova	Get valid server state	Tomi Juvonen (Nokia)	Tomi Juvonen (Nokia)
Nova	Notification for server status change	Balazs Gibizer (Ericsson)	Balazs Gibizer (Ericsson)

Other OpenStack projects that are under discussion include contributions from Doctor include Cinder and Neutron.

[6] See https://wiki.opnfv.org/display/COM/Openstack.

Demonstration

A demonstration of Doctor’s functionality is available at https://www.youtube.com/watch?v=4IlW1bkgi4o.