Availability For Network Function Virtualization

Alon Bernstein
Cisco Systems

ABSTRACT

High availability (HA) is one of the most critical requirements of a service provider network and HA solutions for storage and compute have been deployed for some time. HA is also a major concern when considering network function virtualization (NFV) for service provider networks. The cloud culture of "just spin a new Virtual Machine (VM)" in case of a software or hardware fault seems at odds with the %99.999 availability requirements for service providera. This paper explores ways to meet the HA requirements of a service provider network with the relatively unreliable cloud infrastructure.

DATA CENTER AVAILABILITY

The following table translate the common benchmark for availability to time duration in year/month/week:

Availability	Downtime per year	Downtime per month	Downtime per week
99.99% "four nines"	52 minutes	4.3 minutes	1 minute
99.995%	26.2 minutes	2.2 minutes	32 seconds
99.999% "five nines"	5.2 minutes	26 seconds	6 seconds

Unfortunately some of the most visible data centers (DC) have suffered failures of several hours in recent years. In 2013 the Amazon cloud was down for 49 minutes exceeding it 99.995% claim. Google had a 4-hour outage in 2015. However the types of issues that caused those global outages, such as overload conditions or connectivity failures are out of scope for this paper and the assumption is that as technology matures this type of catastrophic events would became more and more rare. We will focus on basic single tenant availability where a single VM fails and another has to cover for it. Furthermore the focus is on packet forwarding VNF (virtual network function) as opposed to storage or compute availability.

NFV AVAILABILITY

The data center world likes to categorize VMs into "cows" and "pets". Cows are VMs that don't need much care, if one fails another quickly takes over. There are many cows and an individual failure is not critical. Pets are unique, relatively few and require care. If a pet fails it's a big deal.

A typical "cow" would be a VM that load shares a task with other VMs, for example serving web pages. Recovery of a cow is fairly straightforward. In case of a software failure the VM can be restarted and re-run. In case of a hardware failure a replacement VM can be started on another server. The process for creating a new VM is pretty much the same process as recovering from a failure and there is no state shared between the old and new VM. This greatly simplifies the handling of failures.

NFVs on the other hand are certainly under the "pet" category and have a few requirements for HA:

- Minimal packet loss: the role of an NFV VM is to process packets. It is obvious that in the case of failover packet loss must be minimal. Since the total number of drops depends on the time it took to detect the failure as

well as the recovery time it's hard to guarantee a zero packet loss.

- Stateful recovery: many NFVs require state. For example a virtual CMTS needs to keep track of E.911 calls. In case of a failure these states need to be carried on to the new VM.

A typical solution for achieving high availability that meets the requirements above is by means of redundancy; one VM is "active" and the other VM is "standby" – essentially the same solution used in the physical world but with one big difference. In the physical world this redundancy is typically intra-box and the implementation details are completely hidden from the user. In contrast the components of a virtualized redundancy solution are visible. This is not to say that it will be more complex from an end user point of view. The process of provisioning redundancy and handling failures can be fully automated. The components of the solution however are visible and can even be selected by the operator from either open source or vendor specific modules.

REDUNDANCY MODELS

There are several possible redundancy models. Most cable providers are familiar with the modes described below. All of which can be emulated with Virtual Machines (VM):

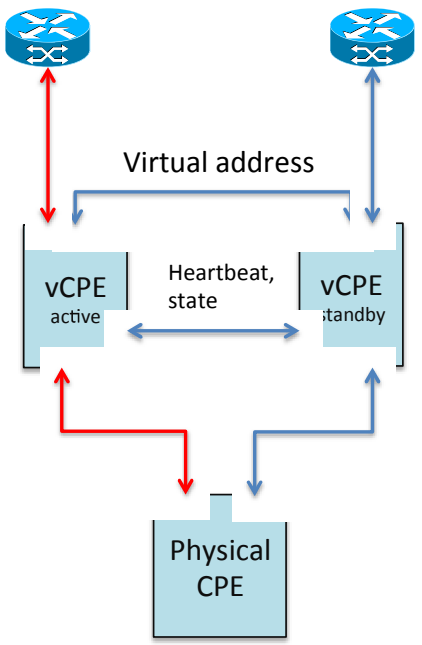
- Active/active: two separate instances share the traffic. If one fails then capacity is halved but selected services can still be prioritized and handled over the reduced capacity. In a physical CCAP device these could be two supervisor cards.
- Active/standby (1+1): as opposed to active/active the standby component carries no traffic and in case of a failure it assumes the identity of the failed card.
- N+1: in CCAP devices the linecards are relatively expensive and so to reduce the cost of high availability several linecards are backed up by a single linecard.

Other redundancy modes include:

- N+M: a group of N active instances are backed up by M standby.
- N-to-1: Similar to N+1 but the back up entity switched back to the primary instance once its recovered. As we will see later this mode may work well for NFV.
- N-to-N: its possible to combine the concepts of N+M and active/active so that each element has some extra capacity to cover for a failure but no specific element is designated as standby or active.

In this paper we focus only on the 1+1 and 1-To-1 redundant configurations and refer to "active" and "standby" VMs.

The data center uses the term "cluster" to describe a group of servers that appear as a single entity to the outside world (for example an e-mail server) and "HA cluster" for a solution that includes availability. In NFV virtual devices tend to be implemented as a single VM. We can still relate to the DC term of cluster with the understanding that it's a very simple cluster: an active VM and a standby VM.



.

Figure 1 Redundant Virtual CPE use case

Figure 1 depicts a particular use case of a physical CPE (pCPE) with an active and standby virtual CPE (vCPE) providing extra services. It builds upon existing use cases in the data center. For example the case of two e-mail servers using the same database would look very similar (where the e-mail server corresponds to the vCPE and the database is the shared resource and equivalent to the pCPE). Both active and standby vCPE have the same address to the outside world and are connected to redundant routers. This can be done in active/standby cases since only one path actively carries data. The obvious benefit is that a switchover is transparent to the rest of the network.

The red path in Figure 1 represents the packet flow. Since it's an active/standby packets will flow only along this single path.

The two vCPEs exchange heartbeat to detect failure on the active VM. Note that the heartbeat is only assisting in making a switchover decision and a failure of the heartbeat does not automatically cause a switchover. The criteria for switching over will be discussed in more detail in the "split brain" section.

CONNECTIVITY

The "cluster" that both the active and standby VM belong to is a single L2 domain from a networking point of view (see Figure 2). Even if we use geo-redundancy, where the active and standby VMs are in physically distributed data centers we would still connect the standby and active VM with an overlay (typically a tunnel of some sort) that would make them appear as belonging to the same network.

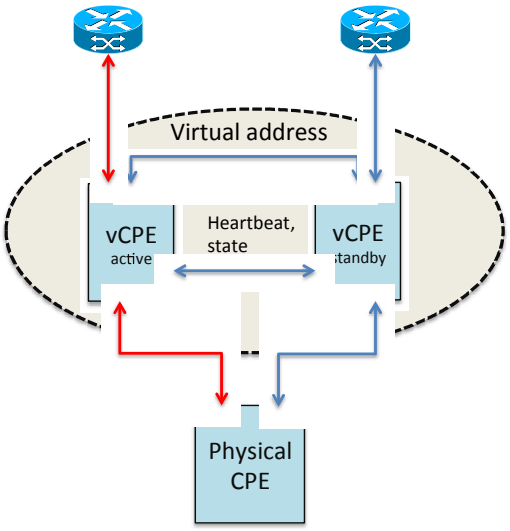


Figure 2 Connectivity overlay

It's the role of an orchestration system to create this overlay as part of the complete setup and life-cycle management of the active and redundant VM pair.

Note that in some cases, such as geo-redundancy, the redundant path may not be as optimal as the active path. For example, the redundant VM may be intentionally placed in a data center further away from the pCPE. For that reason the N-to-1 model may work better because as soon as the fault is resolved control will be returned to the optimal

instance as opposed to leaving the recovered VM as a "standby".

SPLIT BRAINS

When two VM try to decide which one is active there might be situations where both become active. The classical condition that can cause that condition is when the link they used for heartbeat check fails: the originally active VM stays up and the standby VM assumes the active is down (since the heartbeat stopped) and becomes active as well. When redundancy is managed within a box (either 1:1 or N+1) then the box management handles this ambiguity internally. Furthermore, in a traditional appliance the heartbeat is carried over a highly reliable internal path, which is not likely to fail in the first place. In the DC the "split-brain" is a well-understood issue and the solution used for redundancy of storage and compute can apply to NFV as well. As depicted in Figure 3 it requires the introduction of an object called "Quorum":

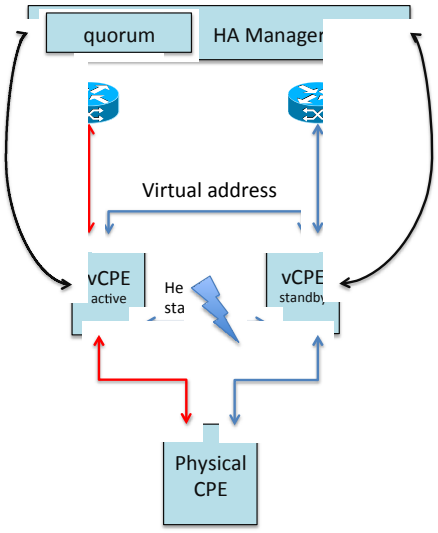


Figure 3 Redundancy Quorums

In order to become active a VM has to "own" the quorum. In the example outlined above the standby VM would try to acquire the quorum from the HA manager but would not succeed (because the active still owns it) and therefor it will stay in standby mode.
It is statistically possible that under certain conditions a couple of system are down, e.g. the redundancy manager itself and both vCPEs (and one would consider that as "double /triple faults" that are not covered by HA). Even in the above case it is impossible for two systems to be active at the same time.

AVAIABILITY AND ETSI-NFV MODEL

ETSI-NFV is an industry consortium that defined a unified framework for NFV deployments. The first phase of ETSI-NFV did not address the issue of availability in detail, but it is addressed in phase 2 (see ref 1). While this paper is not meant as an ETSI-NFV tutorial, it will outline the main functional components that are directly involved in creating a high availability solution.

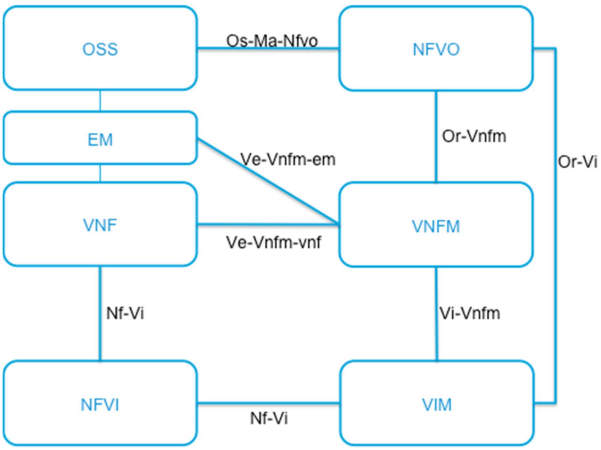


Figure 4 ESTI-NFV architecture

- The NFVI (Network Function Virtualization Infrastructure) should be unaware of a redundancy solution. A VM is a VM regardless of whether its active or standby
- The VIM (virtual infrastructure manager) should also be relativily unaware of redundancy. From the VIM perspective all it has to do is

create/destroy a VM regardless of its state.

- The VNF (Virtual Network Function) itself should know if its active or standby but should be the same SW package for both states.
- The VNFM (Virtual Network Function Manager) can act as a health monitor point for both the active and standby VM and will report the health monitor results to the NFVO layer that would make the final decision on making a switchover. In that sense even the NFVM is not that aware of the states of the VM (active/standby) and from its point of view its just two VMs it had to manage.
- The NFVO (NFV orchestration) is the element that triggers the creation of a redundant instance, the ownership of the quorum and the setup of the network overlay between the active and standby VM. Furthermore, the orchestration system has the final say on weather or not to perform a switchover. While it may seem risky to have the orchestration system be so involved in redundancy (as opposed to lower layers or pure peer-to-peer between the active/standby instances) it is the entity that can make the best decisions because of its system wide view.

STATEFUL RECOVERY

In some cases active and standby VM need to share state, e.g. routing tables or session information.
The active instance can update the standby either in a peer-to-peer fashion by sending state directly from the active VM to the standby or through a shared database where the active is a writer and the standby is a reader. RESTCONF/YANG can be a good transport of state between the VMs.
Some virtualization environments offer "VM mirroring" where the whole memory context of an active VM is synchronized to a redundant one. This greatly simplifies the SW since no state synching code has to be written and the applications are completely unaware of the fact they are synchronized, however, it takes a lot of BW to pass memory blocks around. It may make sense to do that in compute or storage use-cases where network bandwidth it not a bottleneck, but mirroring clearly does not make sense for NFV packet processing.

PLANNED FAILURES

Once a redundancy framework is in place it can be used as a platform for "planned failures" as well.
A prominent example is software upgrade. Instead of bringing the service down during an upgrade procedure the following workflow can achieve the upgrade with a minimal service disruption:

- Update standby VM with new SW.
- Switchover and verify operation. If failed switch back and downgrade
- If upgrade operational then update the other VM
- Switch back. Now both instances run the new software.

CONCLUSIONS

Typical cloud deployments are not %99.999 available. Redundancy for NFV can help in reaching this availability goal.
In traditional appliances redundancy is an internal implementation issue. In the cloud the details of managing redundancy are visible and the modules that comprise them interchangeable. Thanks to automation this extra visibility and flexibility need not make operation more complex.

REFERANCES

1. ETSI-NFV phase 2 : http://www.etsi.org/news-

events/news/850-2014-12-news-etsi-network-function-virtualization-enters-phase-2