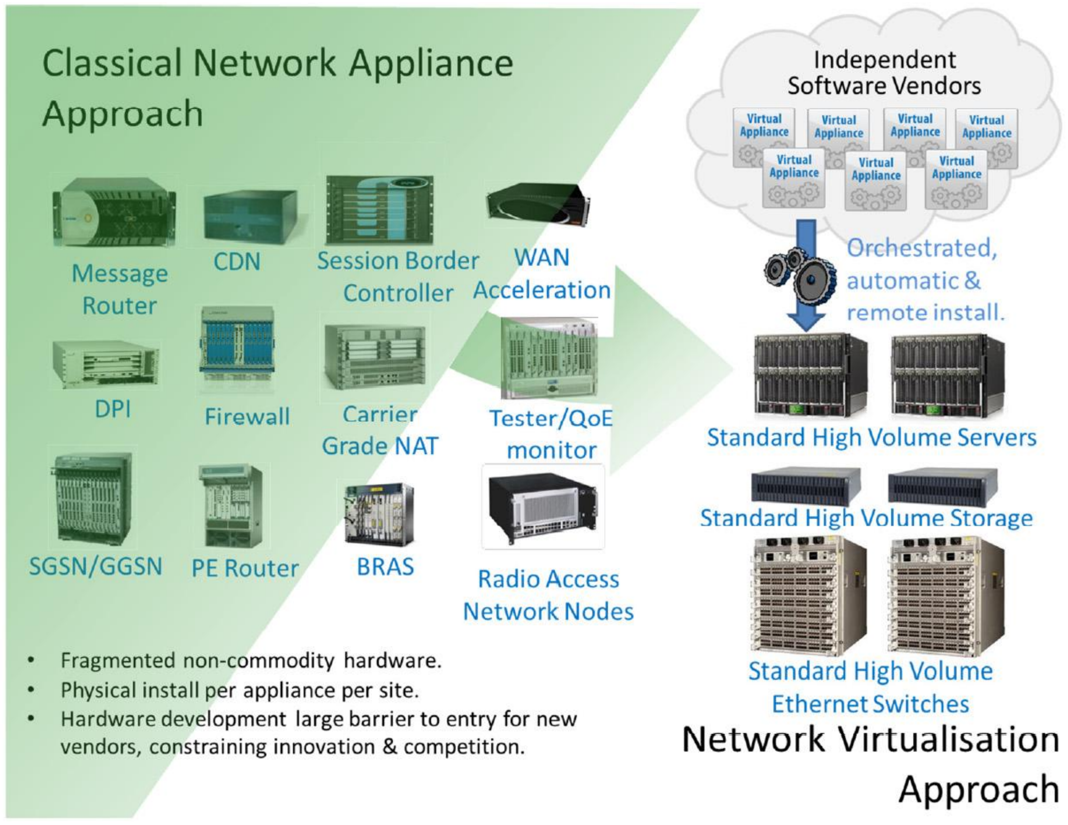
Comments on the NFV White Paper

The paper entitled "Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges & Call For Action" represents the outline of an operator-driven project aimed at creating a model of network services that shifts more service logic and intelligence from proprietary devices to standard servers. The paper identifies two primary goals; improve the "cost curve" of network services to increase return on infrastructure and create a more agile framework for creating new services, especially those with functional components beyond those needed for simple transport and connection.

The basic model of NFV from the paper (below) illustrates a transition from an "appliance" approach to network services to a model where software components create "virtual appliances" whose functions map to those of both current appliances and (potentially) future requirements.


Classical Network Appliance Approach
Message Router
CDN
Session Border Controller
WAN Acceleration
DPI
Firewall
Carrier Grade NAT
Tester/QoE monitor
SGSN/GGSN
PE Router
BRAS
Radio Access Network Nodes
• Fragmented non-commodity hardware.
• Physical install per appliance per site.
• Hardware development large barrier to entry for new vendors, constraining innovation & competition.
Independent Software Vendors
Virtual Appliance
Orchestrated, automatic & remote install.
Standard High Volume Servers
Standard High Volume Storage
Standard High Volume Ethernet Switches
Network Virtualisation Approach

These virtual functions are then composed into services/experiences by an orchestration process that would also be expected to deploy the elements on a pool of resources. We believe that this pool is best characterized as a "cloud" even though the virtual functions may or may not include business services now characterized as elements of cloud computing, because the IT technology being advanced for cloud computing would offer operators the most logical platform for NFV and would do much to insure that components were as plug-and-play and non-proprietary as the paper indicates they must be.

Experience in the IPsphere Forum (IPSF, now part of TMF), the Telemanagement Forum (TMF) and CIMI Corporation's own open-source Java-based service element abstraction project suggests to us that the key to achieving the goals of the paper is a structured top-down function virtualization process. This process could be directed both at capturing as much of current network functionality as possible, preserving investment in current equipment, and moving NFV to the ultimate goal of fully hosted functionality. We suggest a mechanism here, but we want to point out that this suggestion is advanced primarily for illustration purposes; other approaches could achieve the same goals.

A Suggested Approach to Refining NFV

Today's network services are created by **service logic elements** hosted on **service platforms**. The latter are largely proprietary purpose-built devices that, as the paper points out, are both susceptible to changes in requirements and likely to fit into a vendor-specific ecosystem that operators must either buy into completely or reject completely. NFV seeks to break this lock by creating a standard function/device relationship that would be both more flexible and more competitive. The technical goals of NFV can then be presumed to be:

- The separation of these service logic elements from their associated platforms, and their definition as virtual functions.
- The definition of a set of interfaces that could link these virtual functions to generic switches and probes to unite the service logic with the data plane.
- The specification of a set of generalized platforms, including industry-standard servers but also including generic switches and packet probes, that would fulfill the functional requirements of NFV and support the interfaces.
- The specification of a set of interfaces that could be used to orchestrate multiple virtual functions into a wholesale or retail service/experience and provision the components onto infrastructure.

Note that where we say “definition” above, we include the identification of existing standard or evolving bodies of work that would meet NFV requirements, and also the liaison with other bodies for the purposes of guiding such work.

Based on past experiences we believe that meeting these goals can be best managed by defining three phases of definition, each of which would also define a level of “NFV compliance”.

The first phase would seek to identify the functional elements of network services, the targets for virtualization. Examples would be “topology mapping”, “address mapping”, “status reporting”, “path computation”, and “selective forwarding”. Each of these elements would be defined by function and assigned a set of interfaces that would indicate in topical form the nature of the inputs expected and the outputs provided. Path Computation, for example, might have as its input a topology map with element weights/costs, a source/destination pair, and a set of restriction policies. The output might be an ordered set of node/path vectors.

Vendors, at this point, would be encouraged to divide their current functionality according to the outputs of this phase. Where the vendor could meet these general requirements their solution would be “NFV Phase 1 Compliant”. Elements that met this compliance standard could be integrated but would almost certainly require custom coding and integration because the interfaces would be only broadly defined.

The second phase would seek to refine the interfaces needed for NFV, which would require determining how NFV virtual functions were addressed (RESTful URLs, etc.) and how the data was exchanged (XML schema, for example). This phase would produce API specifications that, when met, would allow for direct integration of virtual functions regardless of their source. Vendors who had components meeting Phase 2 Compliance rules could be interconnected providing that an overall platform for orchestration and deployment were developed and the components were organized within this platform framework.

The final phase would be to define a baseline set of NFV platform services, including management interfaces, OSS/BSS, orchestration and deployment interfaces, etc. This phase would create what was effectively an NFV “Platform-as-a-Service” definition into which software components from any source could be integrated, providing they implemented an accepted virtual function definition.

Some Specific Observations

Reviewing the document we had a few specific observations regarding the way that NFV goals could be met most flexibly.

First, the document seems to presume that the generalized server resources would be organized into a pool using virtual-machine principles. There are two primary VM-hosting models recognized today; the model where a hypervisor operates at the hardware level, below the operating system, and creates a highly partitioned view of the server resources. The other model uses OS facilities to create a VM partition, and here the partitioning of virtual machines is less rigorous. While the hardware-hypervisor approach offers greater isolation, it also tends to waste resources because VMs are assigned resources in a fairly fixed/rigid way. An OS-VM model might offer ample isolation and improve resource usage, and in addition this model could adapt to hosting native virtual functions as the specifications evolve.

The second point is that we believe the ultimate goal of NFV cannot be met except by defining what is effectively a "cloud operating system" or PaaS that supports NFV virtual functions. Resources could "join" the pool if they met NFV requirements and applications would be written not to an arbitrary OS/middleware standard but to the NFV PaaS standard. If this is done, then virtualization gradually becomes less necessary since all the functions would be running in the same platform. This might also permit some functions to be migrated flexibly between a vendor appliance and a generic server to reflect how operators balance the advantages of local execution on a network device against cheaper server-hosted execution in the cloud.

The third point is that orchestration of functionality is a much more complicated problem than it appears. Traditional componentized software (the Service Oriented Architecture or SOA model) is orchestrated through a service or message bus. This mechanism is widely used but we have seen test implementations for it in real-time systems where performance was so bad as to render the whole approach invalid. Operators have told us that they do not see services being composed dynamically, meaning that service logic would be modified ad hoc based on service order parameters. Instead they saw a "service architect" building something that would then be ordered in mass quantities. This process is more suited to use of an Interactive Development Environment (IDE) to "drag and drop" virtual functions into a service/experience framework, and this would generate considerably better performance and also reduce testing and validation problems.

Finally, we found in the past that the activities that had the greatest success were those that made the best initial progress. As all the operators who contributed to the NFV paper realize, the problem of current network costs and operator success in future (more OTT-like) services is already here. In fact, operators in our surveys reported these issues first **five years ago**. Vendors have not responded to what we believe were obvious market challenges for their buyers the operators, and insuring they respond now will depend in part on the activity creating and sustaining momentum in the media. No vendor wants to be declared "behind" in a highly visible process. While PR and promotion may not seem an important element in a technical project, we believe it to be a critical element in this one.