SCTE·ISBE CABLE-TEC EXPO 2019 NEW ORLEANS, LA SEPT. 30-OCT. 3

2019 Fall Technical Forum SCTE·ISBE • NCTA • CABLELABS

The Evolution of Network Virtualization In The Home

Improving User Experience And Manageability

A Technical Paper prepared for SCTE•ISBE by

Patrick Goemaere
Chief Architect Cloud Services Connected Home CTO office

Technicolor
1626 Craig Pl 90732 San Pedro US
+1 (818) 442 7183
patrick.goemaere@technicolor.com

Table of Contents

List of Figures

Introduction

With the rise of Internet of things, the success of mobile computing and the consumption of more and more video related content, the home network has increasingly become a more complex environment, containing a rising amount of heterogeneous network devices that need to satisfy the demands of their habitants.

Compared with traditional enterprise networks, users typically don't have the technical skills, or want to be burdened with complex tasks as home network management, troubleshooting or configuration issues. The home infrastructure is predominantly cooperative and self-managed, with different type of devices often owned and controlled by different household members.

"The technical know-how required to set up a network and run music or video across cables or wi-fi, is the elephant in the room that no-one wants to talk about."

In this respect, home networks face 3 systematic challenges:

1. Hard to Manage: Since home users lack the technical savviness to configure their home networks, they typically operate with default settings, and as a result are poorly secured, difficult to extend with new devices and services, resulting in a lack of functionality and experience desired by users.
2. Hard to customize: The home network cannot be customized for the needs of specific applications (Ex Home WiFi coverage, Video streaming) and lacks the means to rapidly introduce new functionality that spurs innovation.
3. Hard to share: In most cases the infrastructure is managed by a single provider with bespoke integration of other siloed solutions, which limits the user's choice and prevents other solutions to share the same infrastructure to keep cost low.

The challenges mentioned above are structural and cannot simply be solved by just putting a nicer user interface on top of the current network architecture. They arise from a mismatch between the stable end-to-end nature of historical Internet protocols and the fast-evolving nature of the home environment. Home networks today are still using the same architecture and internet protocols as defined in the 1970s for the whole internet and carry many of the assumptions made at that time. These protocols where designed with the assumption of devices operating in a trusted environment, the availability of skilled network and system administrators, and tried to accomplish a set of goals that simply don't apply for a home network.

The evolution of CPE equipment design has been extremely slow over the last decades, mainly driven by the fact that they are manufactured by different vendors and combine the forwarding of IP packets with proprietary control software and API's to control and configure these network functions. As a result, different management protocols are used to control individual devices, which has led to a very fragmented CPE landscape.

In the last decade, we have seen a powerful paradigm shift in the networking domain towards Software Defined Networking (SDN) and network virtualization functions (NFV), which has become mainstream for optimizing the complex and dynamic interactions around networking in the datacenter and cloud infrastructure. This technology shift started around 2008 in research as a response on the difficulties to manage today's networks, cost of equipment and interoperability issues.

Thanks to this evolution, and the agility that this new approach delivered, a lot of focus and innovation have been achieved around SDN and NFV technology, applicable in the context of the datacenter. Nevertheless, this technological approach can bring a lot of innovation in other domains as well, and since 2012, we have seen a lot of research work and publications around applying SDN/NFV technology in the Home networking context, which faces similar complexities which cannot be handled by today's traditional networking technology.

This paper will cover a general introduction of the SDN architecture and it's evolution with a focus on OpenFlow as the enabling technology and will focus on the applicability and different use cases in the context of home networking scenarios found in various research work done in the last 5 years in this domain. Also, the evolution around NFV in the datacenter will be covered as it's applicability for consumer CPE devices.

Since these techniques are now becoming a commercial reality, with different vendors that provide solutions for CPE equipment, the paper will also cover the gaps and issues that still exist.

Finally, the paper will discuss a way forward in the industry to evolve current CPE design to allow these new technologies to complement the current network design in a more open way, leveraging a fully virtualized CPE architecture, complying with standards, and ultimately addressing the quest for more collaboration in the industry around open source and open standardization.

Content

1. Historical overview of SDN Technology

1.1. Introduction

Software defined Networking techniques have a long history that started more than 30 years ago in a pursuit to make networks more programmable (Feamster, Rexford, & Zegura). Today SDN is considered as a key enabler, enabling innovation in how we design and manage networks. Albeit the term SDN is relatively new, it became popular as a technology paradigm driven by the rise of cloud and the datacenter for solving virtualization of the network infrastructure.

Originally, networks were designed using network protocols that went through years of standardization efforts and interoperability testing. Network administrators traditionally configured individual network devices using configuration interfaces that varied across vendors and even between different products from the same vendors. Today Customer Premise Equipment (CPE), like residential routers and gateways, still operate at the level of different protocols, management and configuration interfaces.

By nature, this process of standardization and deployments is slow in convergence and has frustrated many researchers and network service providers with the timescales, which were necessary to develop and deploy new network services, let alone to experiment and innovate in the networking domain.

SDN technology today fundamentally changed the way on how we design and manage networks, and typically has three profound characteristics.

SDN separates the control plane, that defines how we handle traffic, from the data plane which forwards traffic based on decisions made in the control plane.

SDN consolidates the control plane so that a single software program can control multiple data-plane elements remotely.

The SDN control plane has direct control over the state of the network data plane elements, routers, middle boxes, switches or servers, via a well-defined Application Programming Interface (API).

OpenFlow is a prominent example of such an API, while there is a multitude of controller platforms and frameworks that have emerged (NOX, POX, Onix, ONOS, ODL, OpenContrail, FloodLight). Today programmers have used these platforms to create new network applications such as network virtualization, dynamic access control, load balancers, etc.

1.2. Active Networking

SDN borrow a lot of concepts and research from the area of active networking research (1996 – 2002), which was the first attempt to articulate a vision for programmable network infrastructure (Calvert).
In contrast with the current SDN approach, active networking focused on making the data plane programmable and flexible, focusing on two distinct models to distribute networking code to intelligent nodes.

- Capsule model, where the code to execute on network nodes was carried in-band in data packets.
- Programmable router/switch model, where the code to execute at the network nodes was distributed by out of band mechanisms.

The following picture visualizes the architecture of an active node, where each node in an active network runs one or more execution environments (EE) and where each of these EE defines a virtual environment that operates on packets. An example of such an environment was a JAVA virtual machine extended to parse byte code programs carried in packets and send running code as packets.
Users invoke Active Applications (AAs), which provide code to program an EE, to implement and end-to-end services.
An execution environment has access to node resources like computing, storage, hardware queues, etc, via a Node Operating System (NodeOS), which was responsible for managing and sharing these resources among EEs residing at that Node.

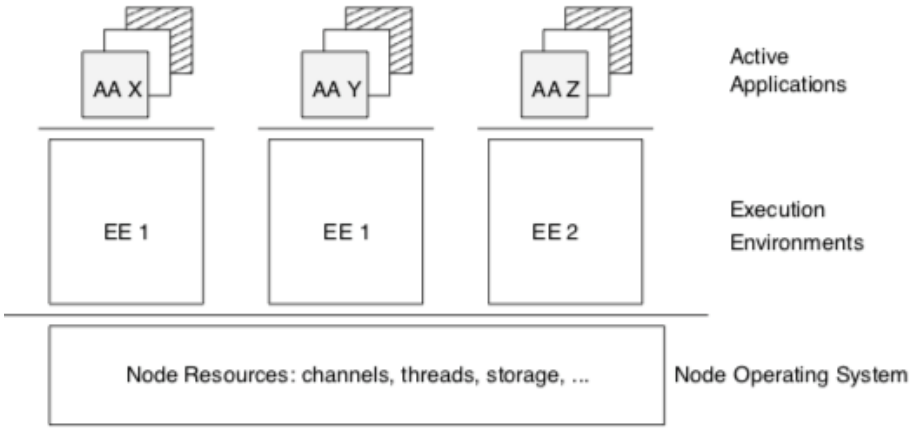

Figure 1 - Active Networking Architecture

The conception that packets would have to carry code written by end users created a lot of opposition in those days and made it possible to dismiss this evolution as inherently unsafe and too far removed from reality. This was further augmented by the lack of a clear business case or killer application for this approach.
Due to these argumentas, the technology did not see widespread deployment, although it was the first technology that articulated a vision around a programmable network.

As a consequence, the next phase of research focused on a better demarcation between the functionality of the control and data plane, to enable a better focus on innovations in the control plane that presented a lower barrier for innovation than the data plane that also needed hardware evolution.

1.3. OpenFlow

Due to the initial failure of the first active networking research to address real business needs and the lack of acceptance, OpenFlow designs started from a more narrow and modest scope of problems to address, and initially started to focus more on routing and configuration management. The real innovation in these new approaches, was that there was a far cleaner separation between the functions in the data plane versus the control plane.

These first projects focused on trying to solve real problems in the network management plane, with a special attention on:

- Innovation by and for network administrators, rather than focusing on improvements for end users, or delivering an infrastructure for researchers.
- Programmability in the control plane, rather than in the data plane.
- Network wide visibility and control, compared to active networking, which was focused only at the device level.
- The concept of separation between control plane and data plane has been the basic architecture in all-further evolutions of SDN designs.

The key elements in this design and its success has been entered around:

- A logical centralized control plane using a well-defined and standardized open interface (OpenFlow API) towards the data plane.
- Distributed state management in the control plane, that could cope with resilience and scaling.
- Embrace of the Openflow standard in hardware switch design resulting in the increased availability of merchant-silicon chipsets providing commodity inexpensive hardware.

Due to the more pragmatic approach, Openflow provided a better balance between fully programmable networks and coarse grain control of the data plane, which could be enabled in hardware while still addressing real-world deployment use cases. Despite this somewhat limited flexibility in the data plane, Openflow was almost immediately deployable, thanks to the availability of existing hardware switches, which contributed greatly to its success. This success was accelerated due to the evolution of the modern data center and its needs towards network virtualization.

By design an OpenFlow capable switch has a table of packet handling rules (flows). Each rule includes a list of actions, drop, flood, forward to specific interface, modify the header field or send the packet to the controller and let the controller decide what to do with the packet. Each rule has a set of counters to track the number of bytes and packets and a priority to disambiguate between rules, which have overlapping

patterns. Upon receiving a packet, an OpenFlow switch identifies the highest-priority matching rule, performs the associated actions and increment the counters.

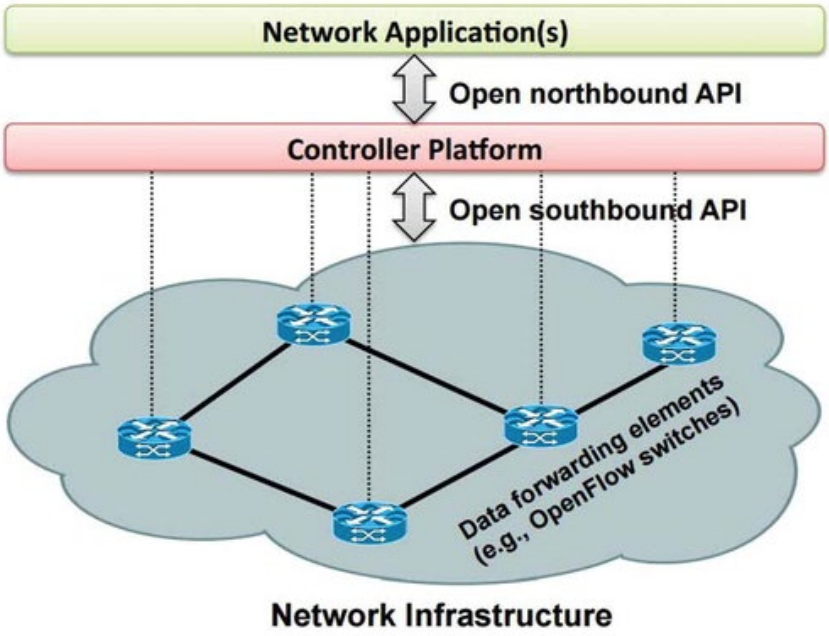

Figure 2 - OpenFlow Architecture

The decoupling of the control plane offers a better way to expose API's from the network to applications and middleware, and such API's are commonly defined as Northbound Api's. These Northbound API's define a central place in the infrastructure where the global application and network policies can be defined. These API's are independent from the southbound API's (E.g. Openflow), as they define a different level of abstraction, although the functionality offered by the Northbound Api's need to be translated and supported by the Southbound Api's.

For reliability, performance and scalability Controllers are typically designed in a distributed fashion. Depending on the implementation and deployment requirements, different solutions make different tradeoffs between the distribution granularity, the function partitioning, the data replication scheme and the consistency choices.

Since controllers provide a high level of abstraction, they are typically referred to as the network operating system. Due to the wide variety of SDN applications, different operating environments (Datacenter, Operators Core network, access network and on premises infrastructure), a lot of different implementation exists with different architectures, written in different programming languages (java, C/C++, Python, etc) both in the open source domain as well as in commercial solutions. It is beyond the scope to detail all these different frameworks, but two prominent and popular open source examples in this space are Open Network Operating System (ONOS) and OpenDaylight. They run cross-platform, are full featured and present high modularity. Besides support for OpenFlow they also support a multitude of different alternative Southbound API's like OVSDB, SNMP, NETCONF, COAP, etc.

The picture hereunder shows more or less a full stack SDN architecture and references some of the most popular technologies used in the space, like OpenvSwitch (OVS) which is the most common open source implementation of the SDN data plane.

Figure 3 - SDN Full stack architecture

1.4. SDN Evolution

OpenFlow as defined in 2008 is one of the most prominent technologies and has been used by most of the hyper scale cloud providers. It is also the most popular networking backend for Openstack deployments. One of the most mature open source implementations is OpenVswitch (OVS). OVS can operate both as a soft switch running within the hypervisor, and as the control stack for switching silicon.

One of the key components that differentiate OVS from other software switches is the native support of the OpenFlow protocol with multiple extensions. For this reason, OVS can operate either as an L2/L3 switch, a hybrid OpenFlow switch or a pure OpenFlow switch. See Figure.

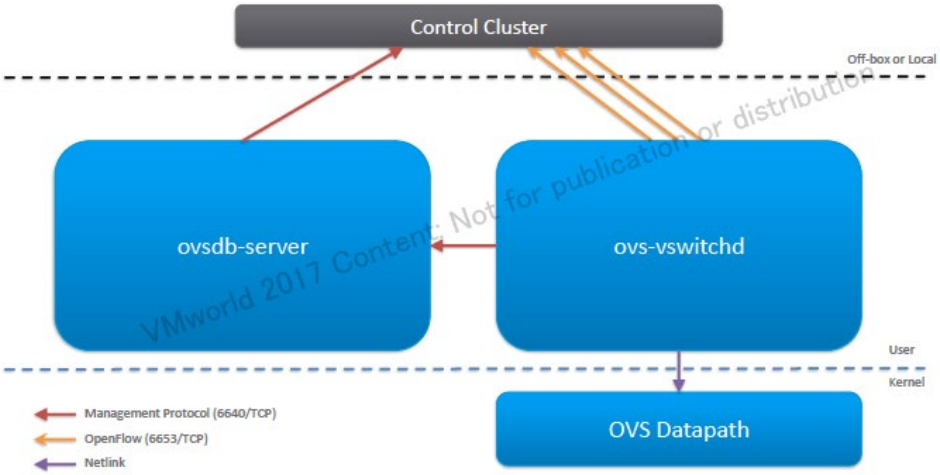

Figure 4 - OpenVSwitch (OVS) architecture

Google has been one of the most active users of OVS in their public cloud, where OVS runs on all their servers in a distributed way to manage their network overlays and configure the networking between the local VM[1]s or container[1]s running on that host. For scaling purposes this platform, which is called Andromeda, has a in house developed distributed control plane that enables them to scale out their global network.

Azure is using their own virtual switch implementation called Virtual Filtering Platform (VFP) which is very close to the design of OpenFlow and follows a similar Match-Action Table (MAT) programming model, but using connections as a base primitive rather than packets and stateful rules as objects to allow use cases like network address translation (NAT) and better security with stateful access control list.

Today, OVS is not the only open source SDN capable software switch. Thanks to the evolution around network virtualization, the experience around different use cases and requirements that provides a better understanding related to cost and scaling issues and the evolution towards white boxes and move to Linux user space for data processing (DPDK), several new innovative opensource implementation[1]s have come into existence.

Some of the most notable examples are FD.io (VPP), OpenContrail (now Tungsten Fabric), Distributed Data Path (DDP) and the P4 programming language.

OpenContrail originally developed by Juniper and later rebranded as Tungsten Fabric focuses on cloud networking and NVF use cases. It consists of two components, the controller and the data plane, called vRouter, which is conceptually similar to OVS but also provides routing and higher layer services. Opencontrails control plane addresses better scalability due to the distributed and horizontal scale-out design.

FD.io which stands for the Fast Data project, primarily focusses on drastic performance improvements. It uses Vector Packeting Processing (VPP) originally contributed by CISCO, which conceptually works on a graph structure (DAG), similar to many big data frameworks like Hadoop and Spark, to allow better parallelization of network data plane code, to exploit better utilization of caching and the parallel execution of code either by different CPU cores or hardware assisted logic.

One of the most promising innovations of the last years is the introduction of the P4 language, a programming language specifically designed and optimized for SDN network programming.
the P4 Language Consortium has devised a solution to specifically handle forwarding instructions. This means it can exploit any advantages built into merchant silicon including everything from common ASICs and CPUS to advanced FPGAs and GPUs giving it the ability to perform at the same level as proprietary switches and routers, and even interoperate with them using the appropriate plug-in. At the same time, it can utilize standard SDN formats like OpenFlow while also supporting hardware-optimized legacy devices, providing a nifty bridge between old-style infrastructure and advanced virtual environments.

Today we are witnessing a fast adoption of the technology by switch manufactures and the white box ecosystem with recent announcements from Google to incorporate P4 in their edge infrastructure called Stratum and inclusion in AT&T's DANOS operating system.

With the ever-increasing network bandwidth to 25GB then to 50 GB and then to 100GB or even 400GB, hyperscalers like Amazon, Google and Microsoft realize that normal CPU¹s can¹t cope cost effectively with this evolution. Most application performance gains will no longer come from the linear Moore¹s Law increases in CPU power but from network accelerators (Bianchi & Bonola).
Network Interface Cards (NIC)s have been used for more than 30 years to connect servers and other computers to networks. Over the last 10 to 15 years NICs have become more capable, supporting higher-speed network interfaces, offloading basic network functions, such as TCP/IP, and more recently offloading virtualization. Smart NICs take this development one stage further by integrating a programmable resource that can be configured to provide additional CPU offload functions for different applications. Smart NIC¹s today can offer a mix of general processors' (CPU, FPGA or ASIC) specific capabilities to provide the right balance between good price performance, easy programmability and highly flexible.

The OVS project has followed many of these evolutions and today supports different configurations, either using DPDK for acceleration or/and using hardware offloading capabilities for their fast path. Using the Linux kernel traffic classification (TC) subsystem, Openflow provides an offloading mechanism called TC Flower, to allow exploitation of hardware capabilities. SmartNic¹s even allow it to run the complete fast path of OVS in hardware. Experimental support for the P4 language has been added in OVS as well.

It is clear that there is still quite some evolution in the SDN area, but the industry has finally agreed that there is not one solution that fits all, and that different use cases require different implementation approaches.

2. A survey of SDN solutions in the Home

Although the concepts of SDN have been successfully applied over the last decade to solve the management complexities around Datacenter and cloud deployments, these concepts are so powerful that they can be applied in other context and environments as well.
In this chapter we explore the state of the art of research work found on using SDN concepts in the context of home networking scenario¹s.
The main driver for doing so, is the fact that the current networking approach expects low-level knowledge which goes far beyond the expertise and willingness of the average home user to configure

their home environment and the failure to improve this task by means of adapting the current networking stack due to the lack of flexible open interfaces (APIs) and programmability of the forwarding path.

The complexity of current home environments on the other hand keeps on increasing, since more and more heterogeneous devices are installed in the home environment, while also more innovative applications are getting introduced in the home besides the more traditional initial internet browsing activities of home users. Integrating these new devices and applications is challenging since there are two contradicting requirements that are hindering the acceptance of these evolutions: Ease of use versus tight control of these information flows to enforce user privacy and user preference; meaning controlling who is using these devices and where the information that is collected is sent.

In this respect, the main driver that made SDN successful in the datacenter context, the simplification of networking management, makes it interesting to be applied in the home networking context as well by virtualizing the home networking infrastructure. As explained in the previous, chapter SDN separates the control plane from the data plane which provides an abstraction from the lower networking layer into a logical network view that can be understood and programmed more easily by network programmers. By allowing access to low level network configuration by software programs, users can manage their home network by more user centric high-level applications (Mortier & Rodden, 2012). Thanks to the standardization and openness of the SDN approach these applications can be developed by third parties, allowing users more freedom and choice so that they can outsource network configuration and management to trusted third parties. To the extreme, the user's network could be partitioned or sliced to allow each application to work in full isolation but still have programmatic access to core networking functions.

The need for a more open ecosystem for home devices, more in particular for the home gateways is not new, and has been previously addressed by trying to align on standardized execution environments like OSGI. Nevertheless, these environments do not offer the flexibility to easily change network specific functions, are written for a specific programming language and have to rely on a fragmented landscape of network management protocols and APIs which are hardwired in the legacy networking control functions. As a result` most of the research of the last years have focused more on an SDN approach and technologies as OpenFlow to address the problem of programmability of these type devices.

2.1. Taxonomy of surveyed works

Based on a survey on research topics on applying SDN techniques in the home (Alshnta, Mohd, & Al-Haiqi, 2018), the authors have classified 42 articles written over the last 7 years around the use of SDN techniques in a residential environment and have observed different patterns and trends.
The following figure shows a taxonomy of the different use cases they encountered in these research papers.
Roughly two major themes have been identified, where one category is more generalized and centers more around overall improvements for home network management, while the others are more specialized around a heterogeneous set of different use cases.

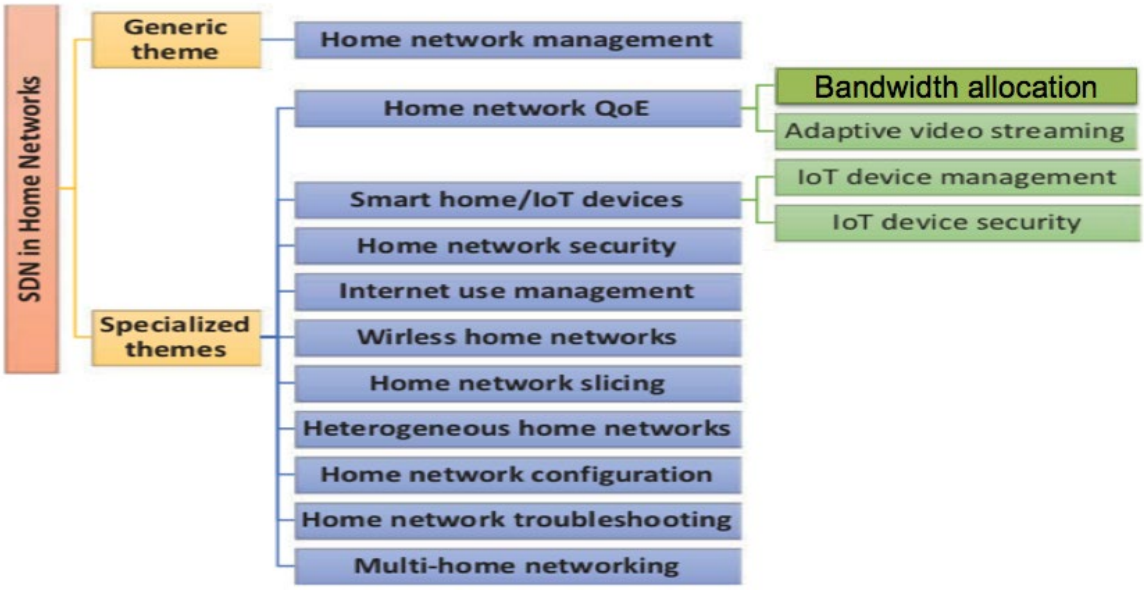

Figure 5 - Taxonomy of SDN use cases in the home

2.2. Generic theme around Home Networking Management

A few years after the initial publication of the Openflow specification in 2008, researches started to wonder on how these technologies could be applied in the home context. The first articles on this topic were developed as part of the homework project conducted in the University of Nottingham (Mortier & Rodden, 2012). This project was aimed as a fundamental redesign of the home-network infrastructure centered around the home gateway and based on applying SDN concepts to provide the home users with a much better view and understanding of their home network, while enable them to control their gateway with novel user friendly user interfaces.

Most of the other works in this category since then have followed the same line of thinking, whilst proposing other technical architectures and solutions to achieve the same improvements around usability of the home gateway. They all agree on virtualizing the home network and delegating the management and control of the home network to an entity in the cloud or datacenter, most probably the Internet Service Provider (ISP) or dedicated third party. Removing the management burden from the end user, and providing them with easy to understand user interfaces to control their home environment was typically implemented using the power and flexibility that Openflow provided to intercept different packet[1]s and acting upon them, while leveraging dedicated configurations in standard network daemons like DNS and DHCP to stay compatible with the legacy network protocols.

The necessity of having to stay compatible with existing devices and network protocols in the home is a specific constraint in the home context which differs from the evolution of SDN in the datacenter and even more particular in the application of Openflow in Campus networks, where SDN provides more flexibility to allow a total clean slate approach to experiment and deploy new networking technologies or implementations.

2.3. Specialized SDN in the Home themes

The rest of the surveyed research papers around SDN in the home, put the emphasis of their work on particular aspects of home networking and are summarized in 10 different subcategories.

The most popular subject in this subcategory is around improving Quality of Service (QoS) or more general improving the Quality of user Experience (QoE), typically by optimizing the bandwidth allocation for different applications in the context of video streaming or multimedia applications. These types of optimizations are either done in a more static way by allowing the user to express their preference related to these applications or can be done dynamically or adaptive by a combination of local traffic shaping based on collected traffic statistics. Most of the articles describe implementations that are running these algorithms from the cloud, while some of the articles run these algorithms locally using an in-home SDN controller to avoid latency issues.

Another theme is to address the issues related to the proliferation of IoT devices in the context of smart home. Most of these articles focus either on troubleshooting the smart home, or address the difficulties related to on boarding and integrating this heterogeneous set of devices in the smart home. Some of these papers are focusing on drastically improving the security aspects of these IoT devices, again relying on machine learning capabilities in the cloud to categories the vast amount of different device types based for example on fingerprinting their traffic patterns observed in the home (Miettinen, Marchal, & Hafeez, 2016). Although most of the existing Low power Wireless protocols are not IP based and hence are difficult to integrate with and OpenFlow approach, the evolution of new IoT radio protocols to all IP based like Zigbee and Thread will likely make these types of approaches more viable in the near future.

Independent from IoT, network security in general is a common theme in some of the research papers. In most of these security solutions users out-source the management task related to security to a third-party controller who has the expertise and capacity to monitor and coordinate these activities over the Internet. It is mostly in this context that the limitations of the current Openflow specification are becoming very clear, due to the lack of very fine grain control and flow definitions in the current spec, which are due to the pragmatic choices that have been made originally.

Because the capacity of internet usage is a particular concern for home users and their family, several white papers address the management of internet usage through an SDN architecture, where specific configured policies can dictate how different users, devices and application can consume a fair share of the internet bandwidth. Most notable example here is off course providing parents a view and control mechanism on how their children can have internet access at dedicated time slots of the day.

Another group of papers address the specific issues arising from managing home WiFi access points, or in general all multi-technology wireless network devices. As wireless has become the dominated in-home networking technology, it is not a surprise that Openflow get's extended more and more to cover as well the Wireless space. Besides management of Wireless access points and clients, SDN concepts are also used to address improvement in finding algorithms that obtain a better usage and allocation of WiFi channels to improve the WiFi coverage in the home. Although there is new standardization, Ex. Easymesh to interact with WiFi extenders, using extenders with an Openflow controller can be seen as an alternative to provide more flexibility between WiFi routers and extenders.

Few papers adopt the concept of network slicing (Yiakoumis, Kok-Kiong, Katti, & McKeown) (Boussard, 2018). Network slicing is a promising technique that creates different slices over the same physical home network, so that each slice is independently controllable and can be isolated for different services. In this context the current limitation in the OpenFlow specification around multi tenancy support for different stakeholders become apparent. Several suggestions are made as for example using a hypervisor approach like FlowVisor on top of Openflow to avoid conflicts with flow configuration of different tenants while still providing compatibility with the current OpenFlow API. Recent papers on this topic (ex. Nokia) suggest an even more novel approach by introducing the concept of a Software Defined

Lan, where Openflow access is more partitioned per different service (Spaces) and controlled by a separate cloud service.

Finally, the last 4 papers in the survey are focused on specific applications. The first one is the most generic one and uses the Openflow APIs for instrumentation to enable troubleshooting. The second one uses multihoming for different types of multimedia applications, while the third one is focused on using an SDN controller for home device recognition and registration. The last one is interested in the advent of upcoming 5G networks and subsequent evolution towards more heterogeneous wired and wireless technologies and uses Openflow for utilizing redundant links for flow rerouting and performing link switching between wired and wireless technologies both under normal conditions and in case of link failures

2.4. Some basic observations

Despite the slight regression in 2017, the last 3 years witnessed an increased interest of researchers in the topic of software-defined home networking, see figure.

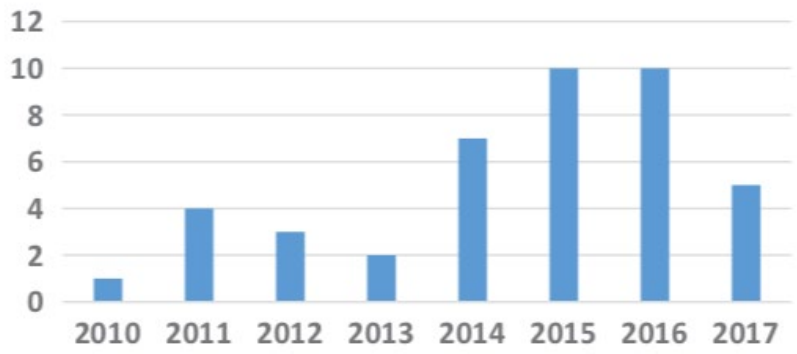

Figure 6 - Statistics on SDN in the Home related research papers

Following Figure compares the number of proposals that are based on a third party or the cloud (e.g. an ISP) to the number of proposals that manage the home network locally within the home itself without outsourcing any task to the cloud.

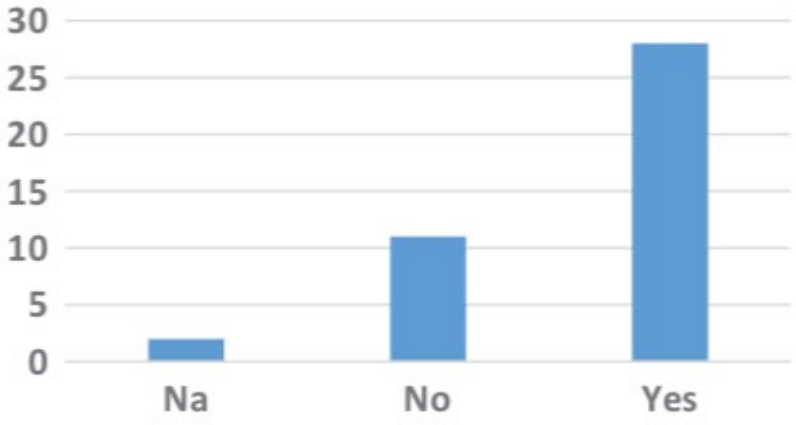

Figure 7 - Statistics on Cloud solutions versus CPE local solutions

Following figure depicts the number of works based on the OpenFlow protocol. As expected, excluding a few works where there are no exact implementation details, OpenFlow is used in most of the works. This comes as no surprise for the most popular open protocol in SDN implementations.

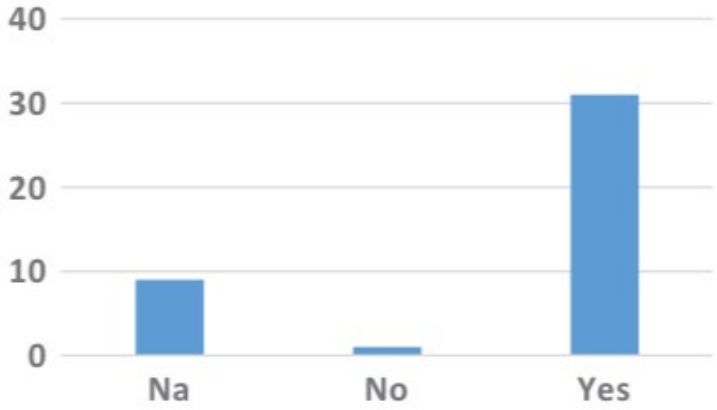

Figure 8 - Openflow usage for in home SDN scenarios

2.5. Conclusions

Looking at the challenges and the complexity of modern home networking environments and their slow evolution over the last two decades, this chapter gives a good overview on how the flexibility and open standardization of a local in home SDN approach can bring the power to address a myriad of compelling use cases. The separation of control from the data plane enables the isolation from the traditional network stack from the legacy infrastructure to provide the home user, or any other third party, with a clean interface to control network operation and exploiting this new programmability of networks. It helps improving and simplifying the network management in a home environment for the average home user in ways which were not feasible without the move towards an SDN approach.

Most of the research work reviewed have focused on this task but produced different architectures and prototypes to prove these concepts and demonstrate the design. Although the majority of these white papers have an SDN approach at their core, they are independent of each other in their approach and implementation and as such most likely incompatible from each other. This mandates the need for further innovation and evolution in this space to come to a unified framework that can combine these different use cases, but for sure the potential to integrate the proposed ideas is great.

Most of these proposed solutions put the user in control to manage their home environment, in most cases with the help of a cloud backend, which amplifies the need for a novel user centric experience. The role of this interface is crucial to make the various functionalities and the information of the underlying home network visible and should allow the user to make changes in an effortless and intuitive way. With the advances in speech recognition and personal assistance concepts these user interactions will be expanded to make further improvements to the user experience.

A last important thing to note is that most of the examples suggest the involvement of a cloud based third party (ISP, or independent 3 party) to assist in the management of the virtualized home network. The biggest challenge here is to find the balance between the home users privacy and security versus the visibility offered to third parties and an economical viable split between local edge functionality versus moving functionality deeper in the network.

3. Network function virtualization

As a recap from previous chapters, SDN technology is a novel approach that facilitates network management and enables programmatically efficient network configuration in order to improve network performance and monitoring. Thanks' to its flexible architecture and open API's the technology can be used in the home to drastically improve the user experience and interaction and address many novel use cases as discussed in previous chapter.

Network Function Virtualization (NFV) on the other hand is an architecture philosophy that utilizes Virtualization technologies to replace hardware proprietary network elements with Virtual Network Functions (VNFs) that can run as software on virtual machines and standard computing hardware. With the appropriate infrastructure, VNFs can be deployed anywhere in the network when needed, creating a network-as-a-service model for network utilization.

SDN and NFV are mutually beneficial but are not dependent on one another. You do not need one to have the other. However, the reality is SDN makes NFV more compelling and visa-versa. SDN contributes network automation that enables policy-based decisions to orchestrate which network traffic goes where, while NFV focuses on the services.

While SDN and NFV don't necessarily require each other to add value to an enterprise, collectively they are joined at the hip. Together SDN and NFV technologies allow better control for the entire network. These technologies can allow companies to increase the efficiency of their infrastructure and make it more functional. SDN and NFV offer the ability to add new services or modify existing services without making extensive changes to the physical network itself.

3.1. Evolution of NFV

NFV replaces network services provided by dedicated hardware with virtualized software. This means that network services, such as routers, firewalls, load balancers, core networking functions like DNS,

DHCP and WAN optimization devices, can be replaced with software running on virtual machines. Virtualized network functions are under the control of a hypervisor, which is the role that SDN fulfils in such a scenario. (See figure)

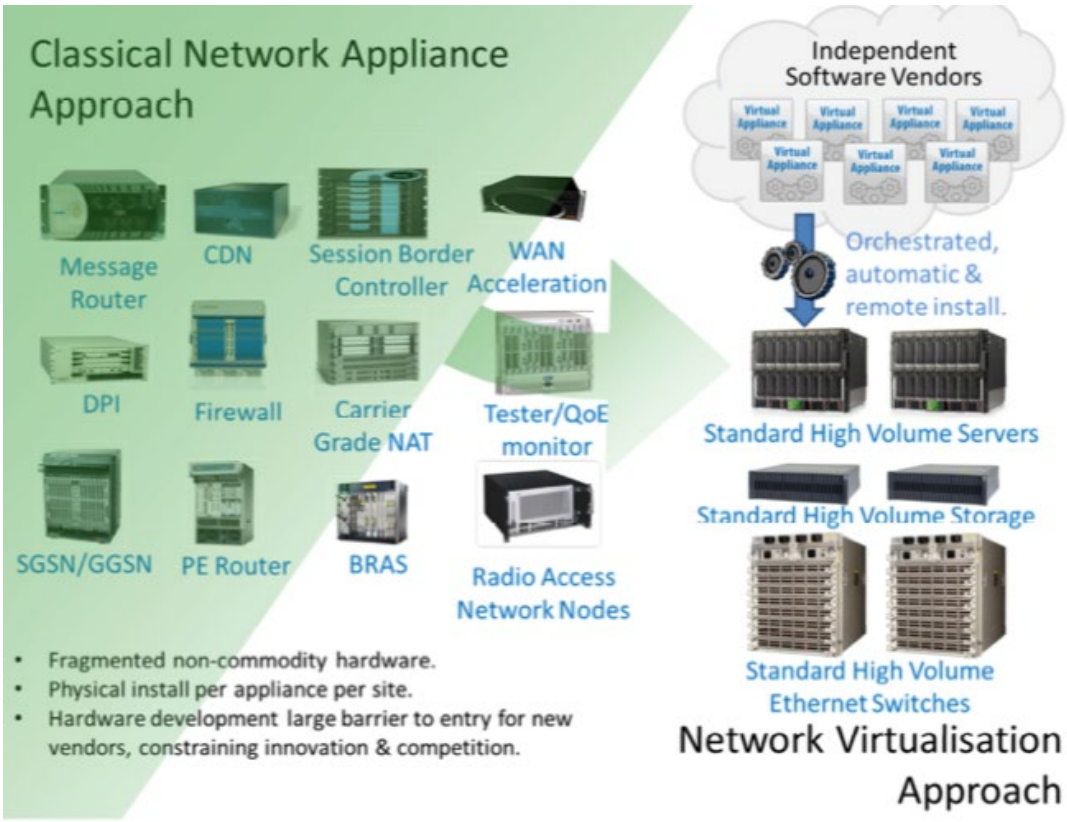

Figure 9 - Network Function Virtualization Approach

NFV services are deployed on commercial off-the-shelf (COTS) hardware platforms, typically running on Intel x86-based hardware and standard switching hardware (White Box). The combination of a base level hardware with this novel type of software creates a virtualized network that is not dependent on specific hardware.

Service providers have been leading the evolution towards NFV as a means to improving time to market and decrease their CapEx and OpEx related with their physical infrastructure. Today, NFV falls under the aegis of ETSI, the European Telecommunications Standards Institute with a close cooperation with the Open Network Foundation (ONF) that brings together opensource initiatives and hardware reference designs.

The goal of ETSI was to take various use cases and put it onto a common platform using virtualisation technologies, initially very hypervisor, virtual machine (VM) centric leveraging both opensource as vendor specific extensions. ETSI has played a pivotal role in the definition of the NFV architecture and numerous specifications.

NFV provides for an open architecture with many flexible options for deploying an NFV solution. The typical architecture of NFV consists of three distinct layers:

- Network functions virtualization infrastructure (NFVi) – the hardware and infrastructure software platform required to run network applications. (Openstack is by far the most popular opensource VIM, aka NFV cloud stack)
- Virtual network functions (VNFs) – software applications that deliver specific network functions, such as routing, security, mobile core, IP multi-media subsystems, video, etc.
- Management, automation and network orchestration (MANO) – the framework for management and orchestration of NFVi and various VNFs.

A simplified architectural diagram is provided hereunder:

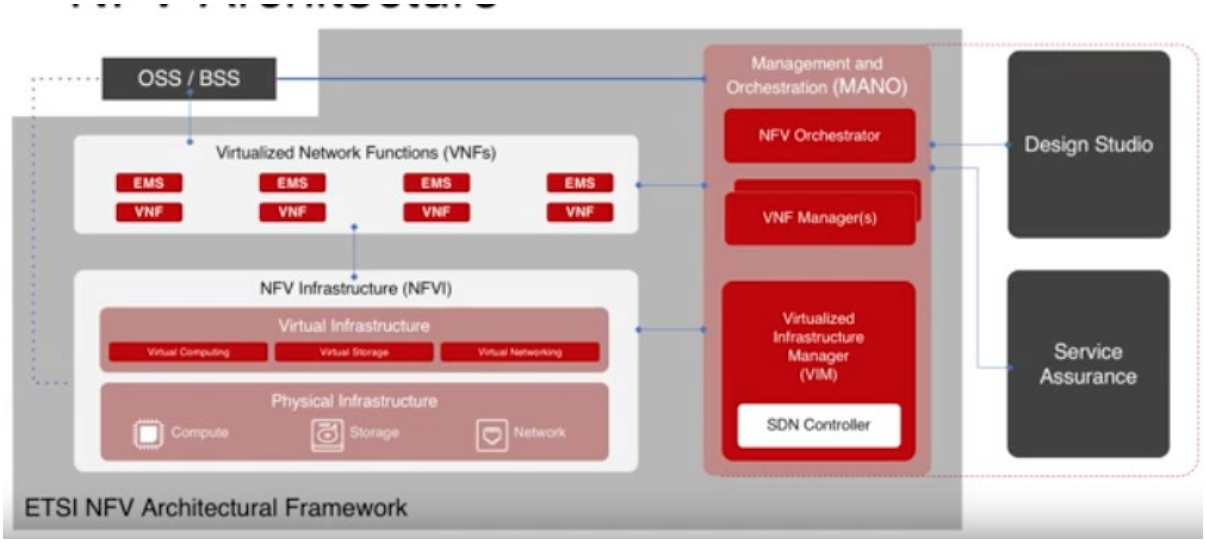

Figure 10 - ETSI NFV Architectural Framework

Open source has played a critical role in this evolution to accelerate the development of the NFV architecture in a way that it tries to complement standards and make them more open and accepted by the community. They have provided reference implementation to accelerate the adoption, validating the specifications while reducing Vendor Locking and creating transparency. Although there exists today an abundance of different open source components, most of the VNF implementations themselves so far have remained highly proprietary. The figure hereunder indicates some of the most relevant open source components used in the architecture.

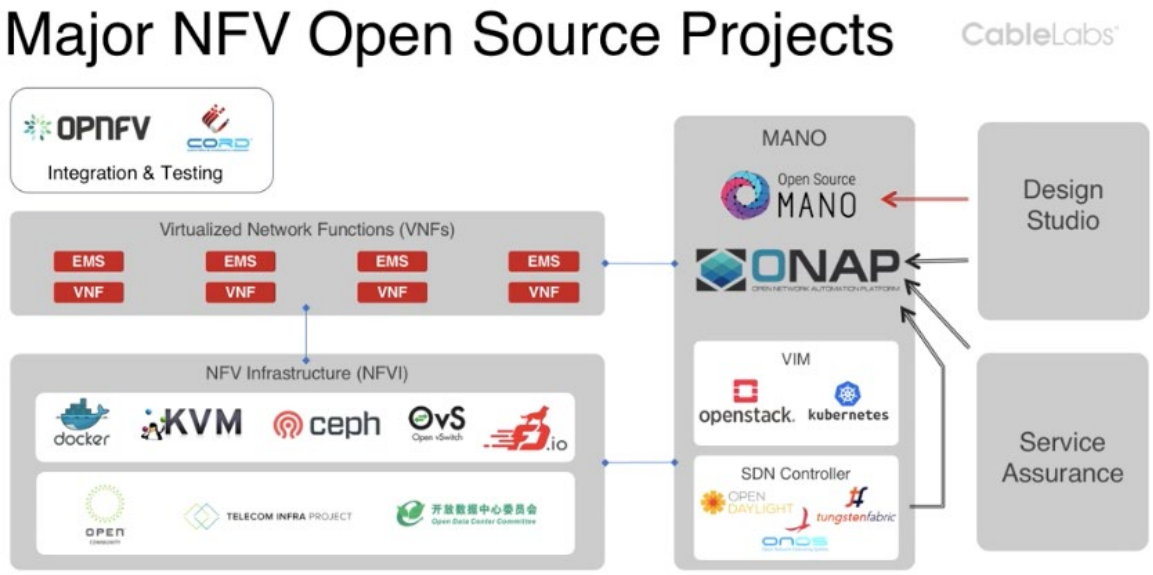

Figure 11 - Major NFV Open Source Projects

Without going into the details of all these components, some of them deserve a closer look to understand their relevance:

NFV orchestration plays a critical role within the broader set of MANO (management and orchestration) capabilities: NFVO tools are responsible for resource orchestration (the orchestration of NFV infrastructure resources across multiple virtualized infrastructure managers) and network service orchestration (the lifecycle management of network services).

Today the two major open source implementations are Open source MANO (OSM) supported by Telefonica and BT and Open Network Automation Platform (ONAP), which grew are out of the merger of open source ECOMP(AT&T) and Open Orchestrator Project (Open-O) (China Mobile). Although they both cover NFV orchestration, ONAP also includes a unified design framework (supporting TOSCA and YANG inspired by Gigaspace commercial Cloudify solution) helping with end-to-end real time service orchestration and automation and provides the components around service assurance, closed loop monitoring and analytics. No real convergence has happened between these two stacks' so far.

Important to note is that in the movement from specialized hardware to more standardized off the shelf equipment like White boxes and enterprise datacenter/switching platforms, effort has been spend in the industry to also create open source hardware blueprints or reference designs.

The two major communities in this space are open compute platform (OCP) and Telecom Infrastructure Platform (TIP).

Open platform for NFV (OPNFV) aims at integrating various NFV related open source projects very close related to the ETSI NFV standards, providing a comprehensive testing framework while providing carrier grade functions. Several competing NFV technologies are provided offering a kind of ala carte menu to select from. (See figure)

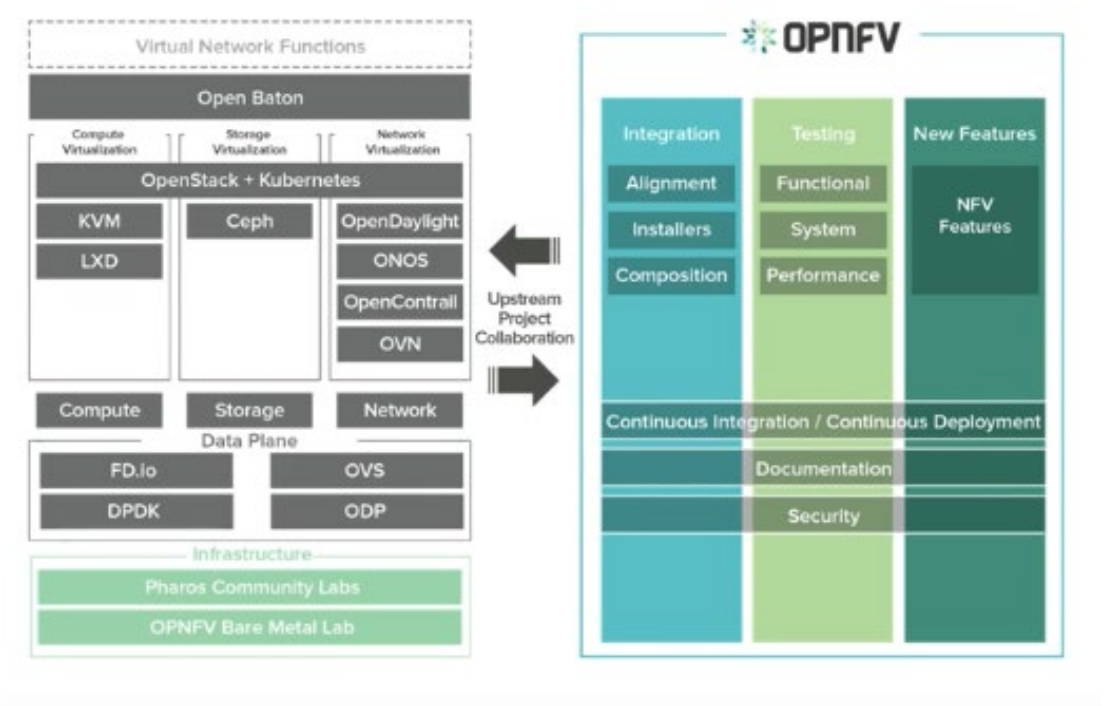

Figure 12 - OPNFV NFV Stack

AT&T has been the driving force behind the Central Office Re-architected as a Data Centre (CORD) program to help the industry on converging the requirements for edge datacenters. Although initially focused on TELCO infrastructure, CORD today offers profiles and specific NFV functions for different industries like R-CORD for residential, M-CORD for mobile edge compute (MEC) and E-CORD for enterprise use cases. The open source activities are guided by the Open Networking Foundation (ONF), which is an operator led consortium. CORD core stack is depicted in this figure.

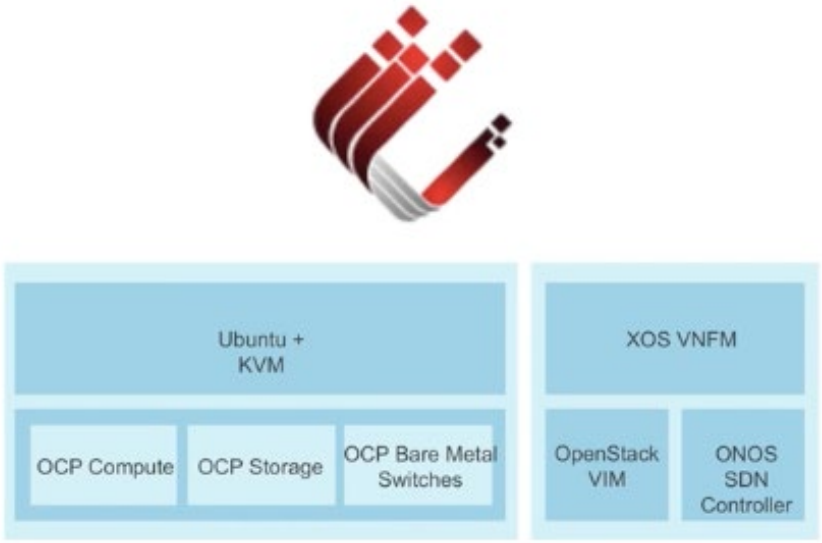

Figure 13 - CORD Technology Stack

3.2. NFV use cases

The amount of use cases that can be addressed by networking virtualization techniques is sheer endless (ETSI, 2017). There does not exits a proper consolidated taxonomy of all these use cases an almost all industries have a set of different problems they want to solve.

Hence for simplicity reasons I will give a limited overview of potential use cases in the Service provider space, covering both the Mobile and fixed access telecom and cable industry. (See figure)

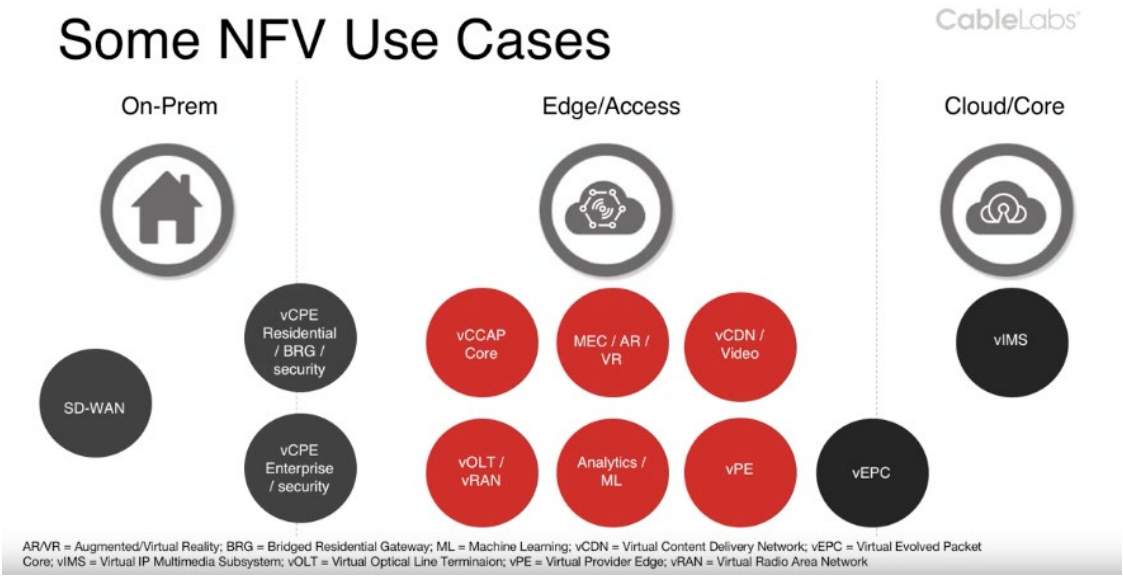

Figure 14 - Some NFV use cases

The majority of use cases for communication Service Providers (CSP's) is centered around the transformation of their current infrastructure towards NFV by replacing existing network components by virtualized ones in order to reduce CAPEX and OPEX and have a more agile infrastructure. This typically includes, standard networking features like DHCP, DNS, Career grade NAT, DDoS protection, Firewall and Deep Packet Inspection (DPI).

Also, the traditional components in the CSP Access/Edge or core network are being virtualized.

In the mobile industry this means virtualization of the Mobile core network, with NFV's like the virtual Evolved packet core (vEPC) and virtual IP Multimedia subsystem (vIMS). On the Edge this means virtualization of the Mobile base station or virtualized Radio Access Node (vRAN).

For Cable operators this centers around the virtualization of their Converged Cable Access Platform (vCCAP), while for Telco this evolution has been focused on their access infrastructure with components like the Optical Line Terminating (OLT) being virtualized (vOLT) and virtual provider edge router (vPE)df. Also virtualization of customer premise equipment (CPE) has received a lot of attention, both for residential deployments as for enterprise and small office/home office (SOHO) bushiness related use cases. Albeit the first attempts where focused on keeping these vCPE devices as simple as possible, which resulted in a failure for residential deployments so-far, the industry doesn't believe anymore that L2-based lightweight CPE will see significant traction, particularly in the enterprise space where L3-based full-function CPEs will be favored.

Current business vCPE have evolved towards a universal CPE (uCPE) device, which provides a better balance and flexibility between functions running at the CPE side versus functions running at the cloud side (ECI). The term uCPE has been coined by AT&T, which recently released their disaggregated network operating system (DANOS) to the Linux Foundation, together with releasing their hardware specifications of an uCPE box towards the open compute platform (OCP)

uCPE allows customer service providers to offer their enterprise and SMB customers enterprise functions as VNFs on a white box server or more commonly on a purpose-built device running at the customer premises. (See figure)

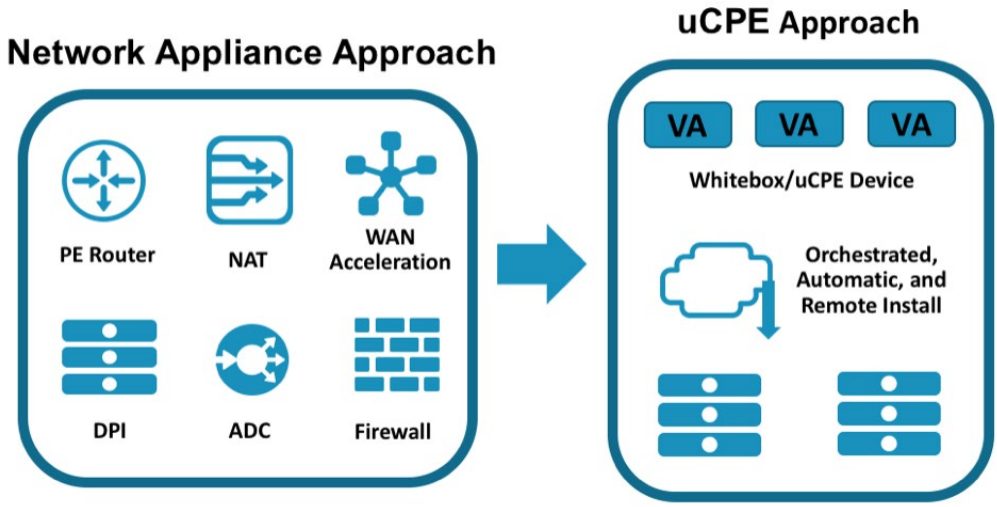

Figure 15 - Universal CPE (uCPE)

The most compelling use cases compared to just virtualizing the basic network functionality for potential cost reasons are those use cases which are innovative and typically are transversal or end to end and provide tangible and clear business advantage.

Key use cases in this area are:

Software defined WAN (SD WAN) (See figure) to reduce cost by using broadband connectivity, simplifies connectivity between remote customer sites and cloud providers and increases security by site to site encryption and micro segmentation of enterprise services.

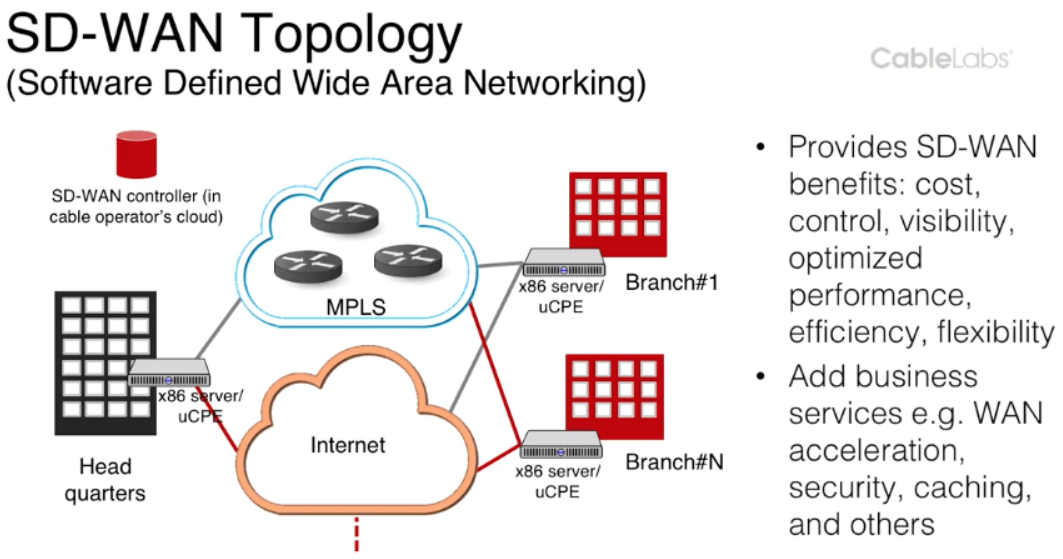

Figure 16 - Software Defined Wide Area Networking

Network Slicing, which allows running multiple logical networks on a common physical infrastructure, hence making the operator infrastructure multi-tenant for different applications. The key benefit of the **network-slicing** concept is that it provides an end-to-end virtual **network** encompassing not just **networking** but compute and storage functions too. This is one of the main use cases in the 5G-network evolution but can be applied for fixed line use cases as well. (See Figure)

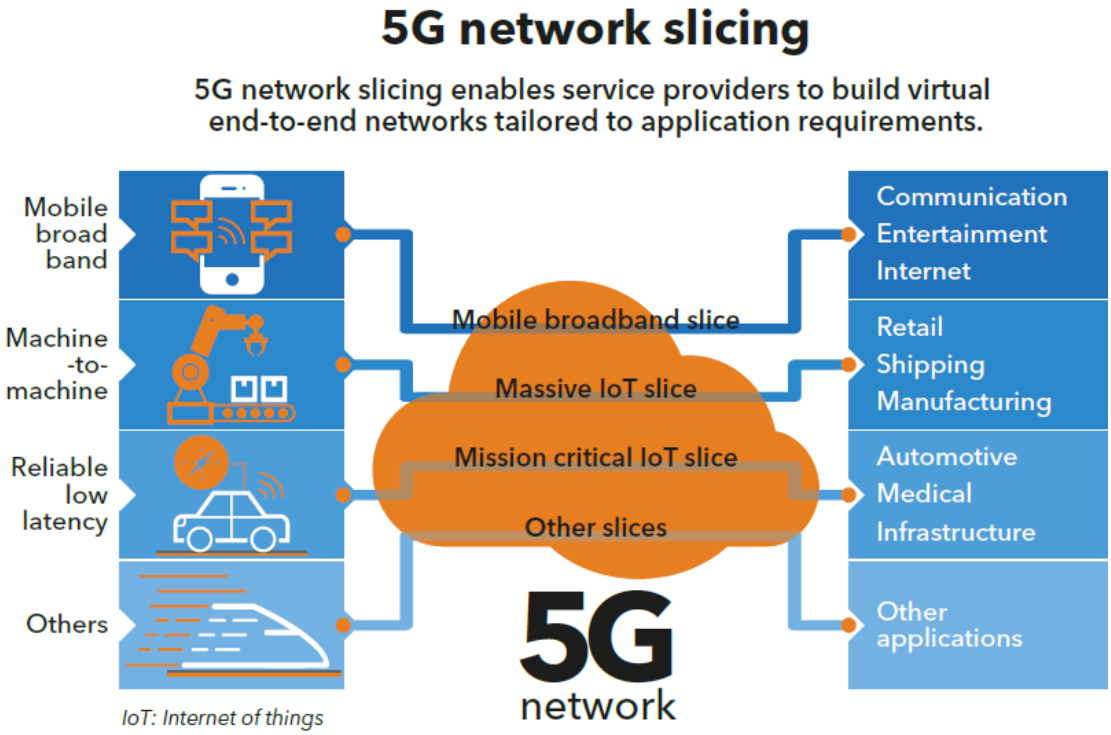

Figure 17 - 5G Network Slicing

3.3. Challenges

This first phase of NFV, starting from 2014 to 2018 has been proving complex and difficult for many operators to deploy at scale. On one hand of the spectrum this process has been hindered due to the fact that it is a transformational process that forces operators to rethink their operational processes and have to go through a deep learning curve. On the other hand, the technical evolution around SDN and NFV has gone so fast that it was hard to keep up with all evolutions and challenges. Some of the problems that operators facing with the technology are the following:

- Lack of insight and information results in a service provider becoming locked into full stack virtualized solutions from a limited set of vendors.
- Lack of interoperability to access a full range of best of breed, trusted VNF's that can be easily and cost-effectively deployed.

- CPU, server cost, rack space and power required to meet the same performance cost of a dedicated custom designed solution, coupled with the operational cost of ownership is for most use cases more expensive.

Since most of the open source NFV projects are quite new, maturity and stability pose a significant barrier for carrier grade deployments. Operationally, VNF's are still multi-sourced virtual appliance, where each of them has to go through a complete lifecycle of sizing, deployment, configuration and upgrades which makes the on boarding process long and complex. Typical length is around 9 months', which negates the original objective around agility. The size of VNF images is quite huge which has to do the so called "lift and Shift" type of development and deployment. Meaning the first generation of VNF's where just repackages traditional network software. The software wasn't architected for virtualized environments.

Due to this lift and shift approach, these implementations where not designed of making use of the power of the cloud with respect to resilience and scaling and typically also lacked good open API's to automate them.

3.4. Future Evolution around NFV

Although the first generation of network virtualization has created a rich ecosystem of system integrators, equipment vendors (white boxes, enterprise infrastructure), software providers, open source frameworks and standards the amount of successful deployments and use cases have been rather limited. The resulting VNF's where rather large monolithic and siloed solutions, with a lack of interoperability running mainly on an Openstack-type of infrastructure. (see figure)

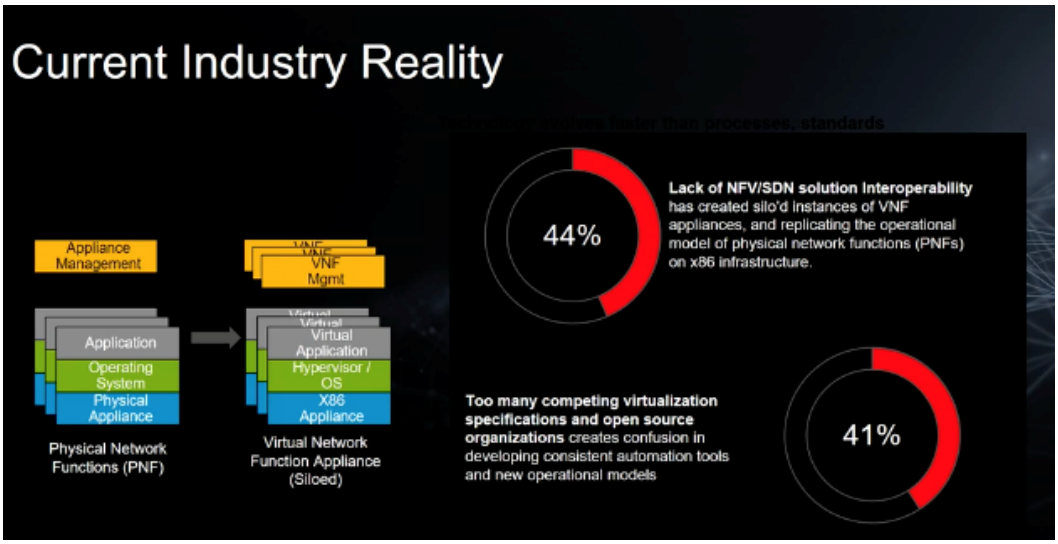

Figure 18 - Current NFV Industry Reality

Comparing to the Web scale company's aka cloud giants like Facebook and Google, one can notice that the challenges faced by CSPs are like those faced by the Web scale companies ten years ago. Web scale companies started by building their businesses on single monolithic applications. Over the time, they developed so-called micro service architectures. By deconstructing an application into smaller components which can be reused for other applications – and restructuring the IT organization into fully accountable micro service teams – companies can create more flexible, scalable and dynamic software development capabilities.

This service-based architecture paradigm in cloud-native distributed applications advocates for "smaller" services, i.e. micro services, to compose complex applications. These micro services are independent, small-scale processes that communicate through pre-defined APIs. Container technologies, including Docker, Kubernetes, DCOS etc., are in fact ideal for cloud-native applications as containers offer a lightweight atomic unit of computing compared to their very server centric VM predecessors.

In the decentralization of the operator's network towards more and more edge infrastructure, one of the main advantages of container based micro-services is their portability. Cloud-native applications are designed to be portable to different deployment environments: for example, in a public, private, or hybrid cloud. They are well suited for edge and customer-premises solutions where compute resources are limited by space and power.

In this respect, the industry has started to move to take advantage of the low overhead of containers to deliver higher performance using cloud-native technologies to build the network functions, so they run in the same network and user space as the applications. Network functions are becoming part of the service topology. Network functions are truly just another service and can be developed and deployed using the same tools as the applications with the same velocity. CISCO has coined the term Cloud-native Networking Functions (CNF) for this new generation of virtualized network functions (CISCO, 2018) (Ericsson, 2018).

Today almost all of the traditional vendors have extended their solutions by integrating Kubernetes has a first-class citizen in their offerings providing a hybrid solution to support legacy VNF on VM's and new cloud native container network functions. They adapted their management plane to support a number of development and deployment pipelines for NFV management and orchestration (MANO).
Major open source frameworks have been adapted as well as OPNFV and carrier grade solutions like SNAP from CableLabs.

The new 5G architecture heavily promotes a cloud native approach at their core and for multi-access edge computing (MEC) (5G_PPP, 2018). The ONAP orchestration layer currently run's on Openstack as on Kubernetes.
Most likely new edge deployment will operate with a much cleaner and smaller stack, removing the legacy Openstack infrastructure to keep deployments cost effective. Over time with improved security around containers like kata containers or gVisor from Google the difference between containers and VM's will likely disappear as abstraction.

In the long run, autonomous networks are the desired future in which every element of the network is automated. High-resolution data would be evaluated to continually optimize the network for current and projected conditions. The result will be programmable mobile networks that leverage telemetry and analytics, automatically scale and self-heal, as well as deliver the highest service assurance.

The following figure presents the different evolutions has described in this chapter.

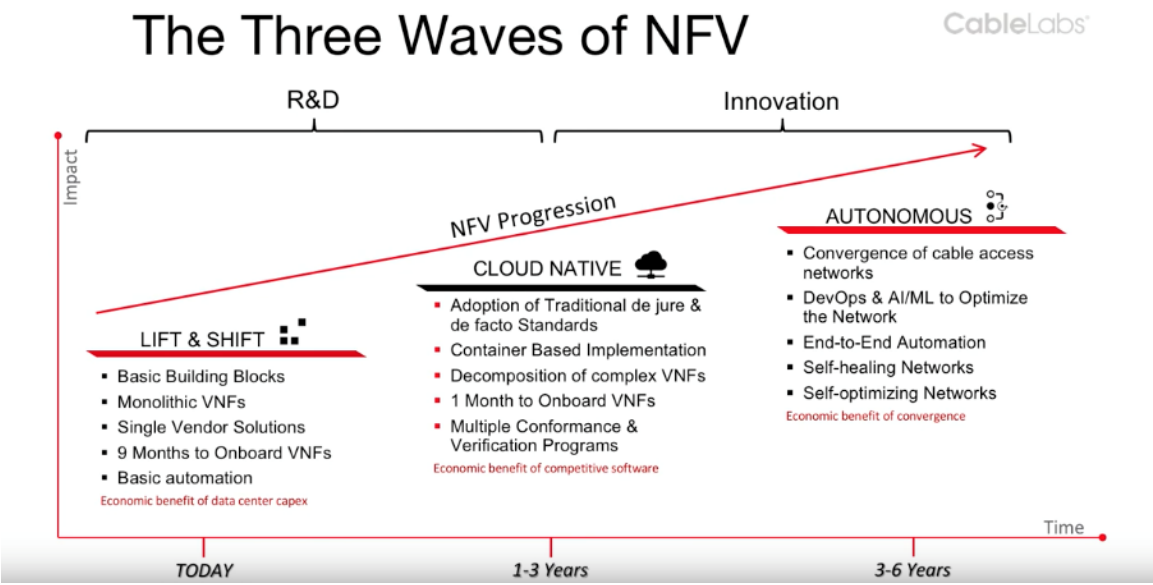

Figure 19 - The three waves of NFV

3.5. Observations and Conclusion

Communication Service Providers are under pressure mainly from the rapid advances that OTT players, web scale companies and new born in the cloud players, are making, thanks to the self-service model and extensive ecosystem's that public cloud environments provide. With increased user demands, the growth of internet of things and the emerging roll-out of 5G cellular, networks are becoming increasingly complex, while the pace of technological advances and innovations are creating a constant evolving architecture.

With an industry that evolves as quickly as the telecom industry, it is easy for hype to overcome progress. Network virtualization has been at the foreground of operator's strategy to improve their margins, agility to innovate and propel them in their journey towards digital transformation.

Nevertheless, the first wave around network virtualization has not enabled them to live up to the promises that the technology potentially could provide.

They have been initially centered around a monolithic approach, aligned with the first advances of server or machine virtualization technology platforms like Openstack and a type of lift and shift approach of traditional basic network technology stacks into network virtualized functions. Due to the fragmentation and immaturity of open-source initiatives and the reliance on traditional vendors who have a conflict of interest with the CSP transformation agenda to provide them with truly open systems. NFV 1.0 has not allowed telco's to scale or bring them the cost savings they need. While this transformation eventually results in a more efficient network with significantly lower TCO, typically the intermediate evolution stages will experience TCO increases resulting from transformation costs and manual processes.

Interoperability issues between different VNF solution providers have also played a key role in the slow progress of the industry.

With the advances of container technology, and the understanding that the new generation of network function's will have to adhere to more "cloud native" principles in order to horizontally scale out and provide the necessary resilience and high-availability in case of failure, the next wave of cloud-native network functions are embracing container orchestration frameworks like Kubernetes to be able to run more lightweight network functions in any public, private or hybrid cloud environment This will becomes even more critical in edge computing, where resource utilization is key.

The initial slow progress around network virtualization and the technical and organizational difficulties have given operators a more realistic view on their timelines to evolve their infrastructure. Advance in 5G and automation technology will be key enablers for this next phase.

Despite all these difficulties the industry has learned valuable lessons to improve, while on the other hand a rich ecosystem of partners, equipment vendor's, standards and open source initiatives has matured to a point that gives more confidence towards the future. The industry is converging around a number of key tenants to achieve this progress.

A more pragmatic approach around standardization is necessary to achieve interoperability and the need for consolidation of the many different standardization bodies.

In this respect a better balance between ad-hoc standardization as witnessed in the public cloud (Ex Kubernetes and Docker) and standardization driven by standardization bodies need to be found, while standardization bodies should evolve towards more open standardization where the existence of an open source reference implementation will be key to drive progress and validate the design. The adagio of just enough standards should prevail instead of the very fragmented approach, which is typical in the early stage of technology evolution and disruption. The bottom line is that we need to accept that "the only constant is change." Innovation in software can bring many good things, but we need to learn how we can eliminate the silos, guard against new ones forming, create better interoperability, and simplify operational complexity.

Open source software—and now even open source hardware—will be critical components. Open source is all about public debate, and unlike previous transitions that were driven more by standards bodies driving consensus, the next generation of communications technology will be driven more by open source organizations like OPNFV, OpenStack, OpenDaylight, DPDK, FD.io, ONOS and ONAP as the underpinnings of 5G will be virtual and very cloud-centric. Open source should drive standards and not the other way around.

Collaboration will become the preferred mechanism for driving change. No single vendor can deliver the full stack, and proprietary technologies will not keep pace with these future needs. This transformation will be delivered in virtualized (not physical) technologies, open source and multivendor, relying on significant integration work across many in the industry to be successful. In this respect we see already a shift in the value chain with new type of ecosystem player's, taking up the role of open source NFV/SDN-trained large system integrators. Examples here are companies like AMDOCS providing a fully integrated ONAP platform and even an AWS based development platform to allow quick development and players like RADISYS providing a first turnkey reference solution for M-CORD. Provided with additional services like a build, operate and transfer (BOT) bushiness model it help's operators to focus on their use cases instead of spending their cycles on integrating, maintaining and evolving their infrastructure.

Improving automation and using a DevOps mentality to orchestrate their infrastructure in an end-to-end fashion will be key. The distinction between virtualized network functions and normal application

components will gradually disappear, and hence leveraging the evolutions in the cloud native automation DevOps domain will be key.

4. Virtualization on residential CPE

4.1. The first wave residendial vCPE

While SDN and network virtualization have a rather strong business rational for enterprise and SOHO use cases and see evolution to uCPE, the residential application of these technologies has not been very successful. Most of the focus initially has been on cloud-based vCPE scenarios rather than running VNF's on premise or in a hybrid scenario, with the main objective of trying to achieve cost reduction by simplifying these CPE devices to mainly layer-2 based network devices and moving most of the layer-3 networking like NAT, DHCP, firewall and routing inside the access network.

Depending on the amount of flexibility required, two scenarios have been envisaged (Hermesh, Shulman, & Ray, 2016).

Using software defined forwarding at the CPE side, allowing an SDN controller to control and manage the network topology at the CPE side, see figure.

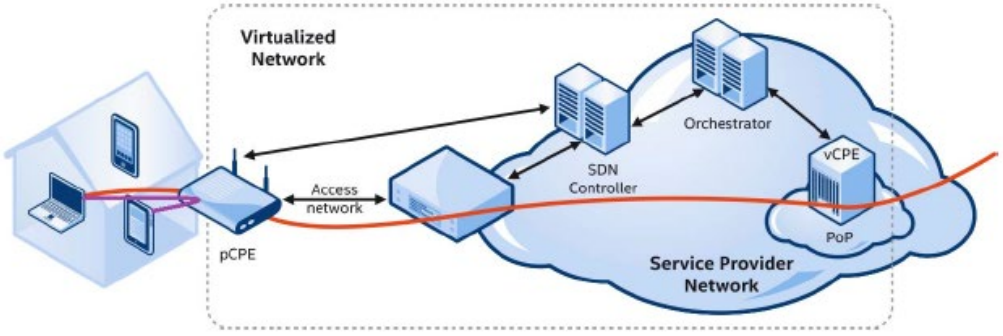

Figure 20 - vCPE with SDN capabilities on CPE side

An alternative to a fully virtualized network and using SDN capabilities at the CPE side is using tunnels as a bridge between SDN domains. Tunnels are logical point to point connections on top of an underlying network. Popular tunneling techniques include Layer 2 General Routing Encapsulation (L2GRE) over IP and Virtual Extensible LAN (VXLAN) over UDP. (See figure)

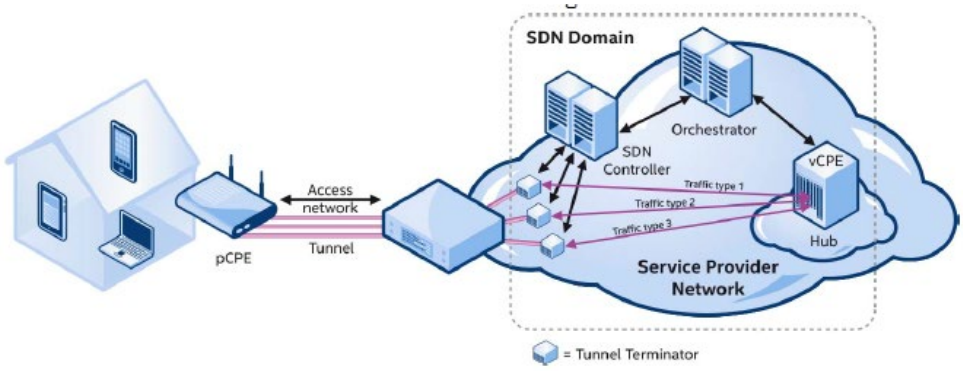

Figure 21 - vCPE with tunneling on CPE side

Since in the later scenario some GW functions remained locally and the complexity and operational expenses of the CPE remained, most of the initial marketing around residential vCPE focused on the first scenario or a hybrid where tunnels where configured by an SDN controller at the CPE side to allow a gradual transition from the existing architecture to a fully compliant SDN architecture.

Despite all hype and effort, the excessive attention towards a thin physical CPE device has never lived up to his promise. This is mainly due to a couple of reasons.

As described in the previous chapter on the NFV evolution, the rather slow progress and the transformational complexity has hindered the progress of vCPE deployment for residential use cases.

For most of the basic network functions, despite the flexibility a cloud model provides, the cost model cannot compete with similar function's executed locally on the CPE, since most of these functions require modest CPU performance and where the energy consumption is subsidized by the consumer.

The lack of compelling business use cases where consumers are willing to pay a premium for the service offered.

The closed nature and fragmentation of current CPE vendor landscape, which does not allow third parties to easily develop functionalities on a CPE device and deploy and evolve them.

The amount and cost of bandwidth required of sending a lot of the traffic towards the backend infrastructure and the rather asymmetrical nature of current access network, leading to trombone effects that becomes a bottleneck.

Latency issues with the approach which provides a sub optimal user experience due to round trip delays in the network.

The limited support of hardware offloading and acceleration techniques of typical low-cost System on Chip (SoC) designs to support integration of software based switches like OVS.

The lack of standards and API's inside the current CPE devices to become truly open and the non-existence of an execution environment that allows easy portability of different software agents on those devices which have a rather monolithic firmware approach.

The very VM centric approach of the first generation VNF's designs, making them unsuitable for deployment on these very constrained devices.

4.2. Current residential CPE landscape

While the first wave of residential vCPE was heavily focused on running NFV's in the datacenter, new innovative players have shifted gear and have started to focus on how to leverage existing residential gateways to integrate their applications. These applications typically address the shortcomings in different domains of the home networking stack to provide compelling services around security, in home WiFi coverage and configuration, IoT use cases and enhanced monitoring capabilities.

Today these applications are built in a siloed fashion, leveraging their own backend/cloud solutions for control and management and relying heavily on analytics and machine learning.

Most of these solutions today are using a mixture of public cloud infrastructure, mainly AWS or Google and open source components to allow them some cloud agnosticism for their core infrastructure.

Some of the major players in this domain are Cujo (Security) and Plume (WiFi and security). They both recently open sourced their CPE agent and are both leveraging SDN network programmability, more specifically Plume is using OVS (Openflow) in their agent which is called OpenSync, while Cujo is using their own internal developed linux kernel module called NFLua which uses similar concepts as in the early days of active networking.

Plume's Opensync architecture is depicted in following figure.

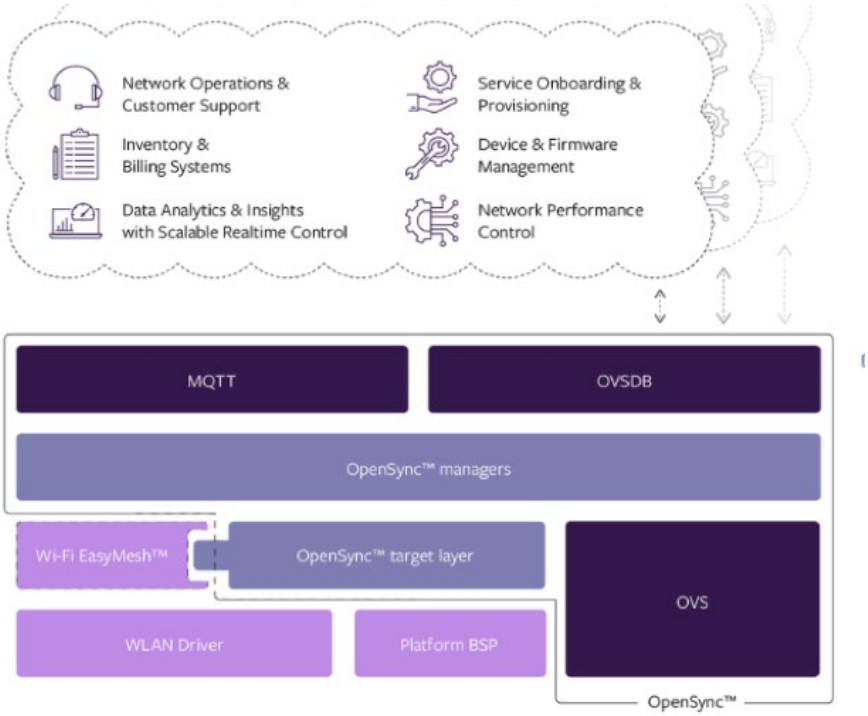

Figure 22 - Plume's OpenSync Architecture

For cloud connectivity, they are using OVSDB, which is a part of OVS, for control and management, while using MQTT for ingesting monitoring events in their cloud infrastructure.

The OpenSync framework on the gateway is centered around OVS, while they have created an abstraction layer, called the target layer to allow interaction and integration with the legacy networking daemons like Wireless, DHCP, NAT etc.

Cujo's agent architecture is depicted in following figure.

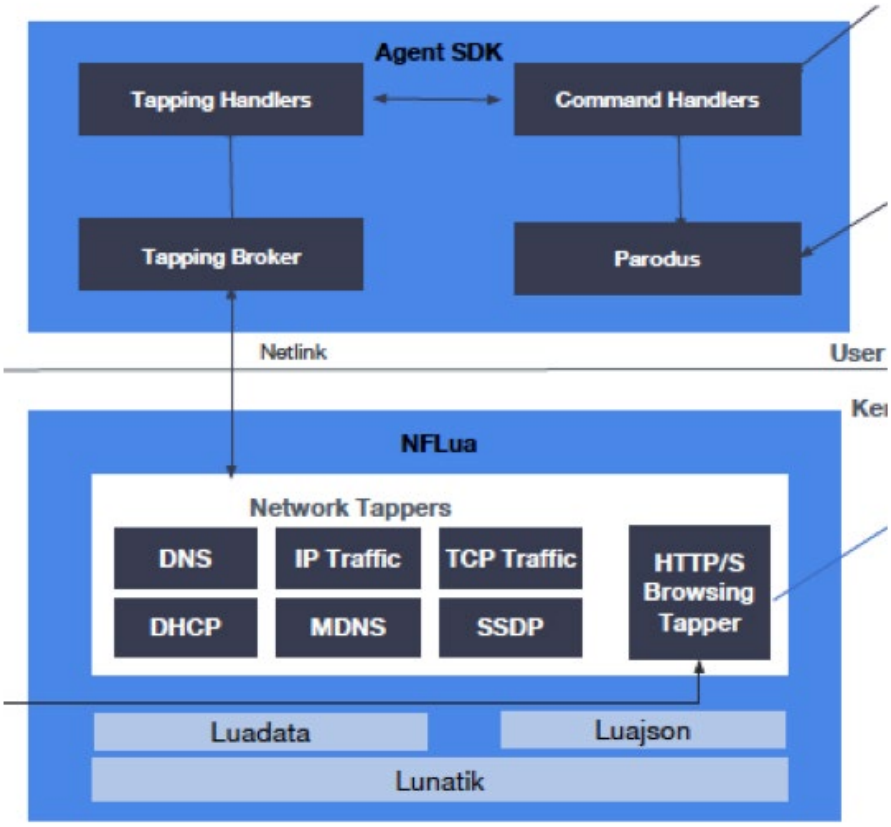

Figure 23 - Cujo CPE Agent

For cloud connectivity they are using XMIDT, that has been recently open sourced by Comcast, which has been built from the ground up in a cloud native way for scalability reasons.

Since their core use case focusses on security and requires quite some programable flexibility inside the data networking plane they opted to write an in-house performant and low footprint engine that could cope with the very constrained nature of these type of embedded devices with respect to memory and CPU budget. This engine which resides in the Linux kernel uses the LUA programming language, which is a data centric scripting language which has been very popular for low footprint embedded devices and gives them the flexibility they require to evolve and adapt their system towards new requirements.

Although these embedded agents are not following the exact definition as used for their cloud oriented NFV counterparts, the fact that they are using software defined networking techniques are putting these

types of agent's in the same category. Thanks to this new evolution, the industry has started to finally embrace technologies like OVS for residential gateways, and SoC manufacturers and traditional gateway OEM's are forced to start optimizing the integration of OVS to make use of hardware offloading and acceleration capabilities in their existing designs.

Most likely some convergence will start to happen around these two approached, centered around extending OVS and Openflow around residential use cases.

Today these agents are integrated as third party application's inside the legacy monolithically firmware of traditional CPE's, and hence are following the same slow feature evolution and deployment cycle as the rest of CPE functionality.

4.3. Containerization and Edge compute

Next generation networks, driven by mobile, enterprise and internet of things evolutions are pushing the boundaries of computing more and more to the edge of the network in close proximity of the user, to cope with increased network utilization and to be able to guarantee performance and quality of service (QoS).

Due to the heavy footprint of current NFV platforms, the industry is shifting towards more lightweight solutions, that are easier to manage and operate at scale.

As an example, 5G mobile networks will utilize Mobile or Multi-Access Edge Computing (MEC) platforms at the edge of the network, in mobile access devices (E.g. base stations) with a key objective of having an IT service environment with cloud computing capabilities in close proximity to the mobile subscribers.

Other edge devices including enterprise edge servers, residential wireless routers and even IoT gateways that connect IoT devices to the internet.

As edge devices are located close to the users, services running at the edge provide higher forwarding performance (high throughput, lower latency) than running services remotely, and therefore save the utilization of the WAN infrastructure, which can correspond to savings of million dollars per year.

Due to this evolution, NFV's are starting to move to lightweight virtualization technologies, like containers to avoid the additional hardware requirements and overhead which is typically associated with using hypervisors and VM's.

One of the more recent research proposals in this field is published by the university of Glasgow, presenting Glasgow Network functions (GNF), that brings together NFV and edge computing by using generic lightweight Linux containers to host VNF's in a distributed, heterogeneous edge infrastructure (Cziva & Pezaros, 2017). The overall architecture is depicted in following figure.

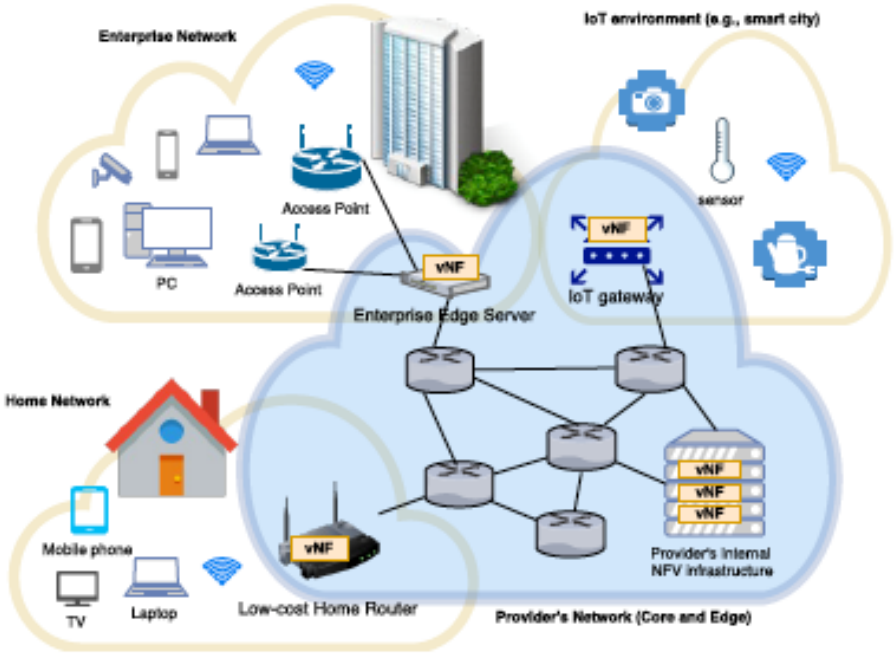

Figure 24 - Glasgow Network Function High Level Architecture

In the past years different container technologies have emerged as a key technology for application isolation, packaging and delivery technology which are using core Linux kernel evolutions such as Control groups (Cgroups) for resource management and Namespaces for process isolation. Linux Containers (LXC) is a prominent example of such a technology, while today Docker has become the defacto standard for containerization, driven by its success of cloud deployments. Compared to LXC, the success of Docker is driven by its focus to deliver the developer with an easy user experience to package software inside a container, delivering a workflow that can easily be automated, and having a vast eco system of tools to be able to share images for different environments.

Recently, a more performant and lightweight version of Docker has become available in open-source by a company called Balena, which provides a docker runtime that can easily be integrated in constrained devices like home gateways and STB, with a footprint of under 30MBytes. Most CPE vendors start to offer support for both LXC as Docker as first-class citizens on their platforms, as a way to improve the integration effort for third party software and services.

In line with the cost/performance curve, the next generation of the broadband GWs will integrate a Modem, a router, best in class Wi-Fi, IOT radios (802.15), Gigabit Ethernet interfaces, multi core CPU (e.g. Quad core ARM), 512+ MB of RAM, 512 MB+ of Flash etc.; all for a familiar price point that is amenable to MSOs. Such integrated Gateways are capable of doing computing at the edge / customer premise

From an operator perspective, the Edge means the base station and their core network infrastructure, but extending their reach towards the physical edge, and applying these lightweight virtualization techniques on their customer premises equipment, which could give them end to end control and flexibility around their efforts on network virtualization, and allow third parties to innovate with new IoT use cases, directly on CPE devices.

4.4. Software Life Cycle Management and Orchestration

While container technology provides a technology agnostic execution environment that is capable of running all types software and even network functions, written in a multitude of different programming languages, the scale of moving these type of computation towards the edge poses additional challenges on how to automate and operate these environments, as the amount and number of edge platforms start to increase. Certainly, when CPE devices are coming in the mix, bumping this challenge towards millions of these platforms.

CPE have been traditionally managed with technologies dating from original concepts as SNMP or TR69 for their configuration and Firmware life cycle management, which are costly and difficult to scale.

Current NFV platforms have the same type of challenge with respect to this type of software life cycle management or orchestration, even if they have moved to container orchestration frameworks like Kubernetes, and have shifted towards standardized frameworks like TOSCA to provide them with infrastructure as code blueprints to deploy applications and network functions. However, current NFV standards are falling short to provide orchestration that can cope with hundred thousand or even millions of disparate platforms.

Driven by the slow progress and frustration inside the NFV community, recently a group of NFV specialist from different leading vendors and operators have coined the term LeanNFV and released a white paper in order to pinpoint current design constrained that are hindering innovation and progress at the Open Network Summit 2019 in San Jose.

Their biggest complaint is the tight coupling of NFVs with the traditional software infrastructural components like Openstack or Kubernetes or SDN controllers, which increase complexity and complicates integration of VNF's from different vendors.

They propose the addition of a type of Key Value store with publish/subscribe semantics which serves as a universal point of integration and that provides a sort of central repository to enable VNFs to exchange state and discover each other.

This type of plugin integration enables NFVs to be plugged into any computational infrastructure, being it Openstack, Kubernetes or maybe in the future just a Docker enabled CPE devices. (see figure)

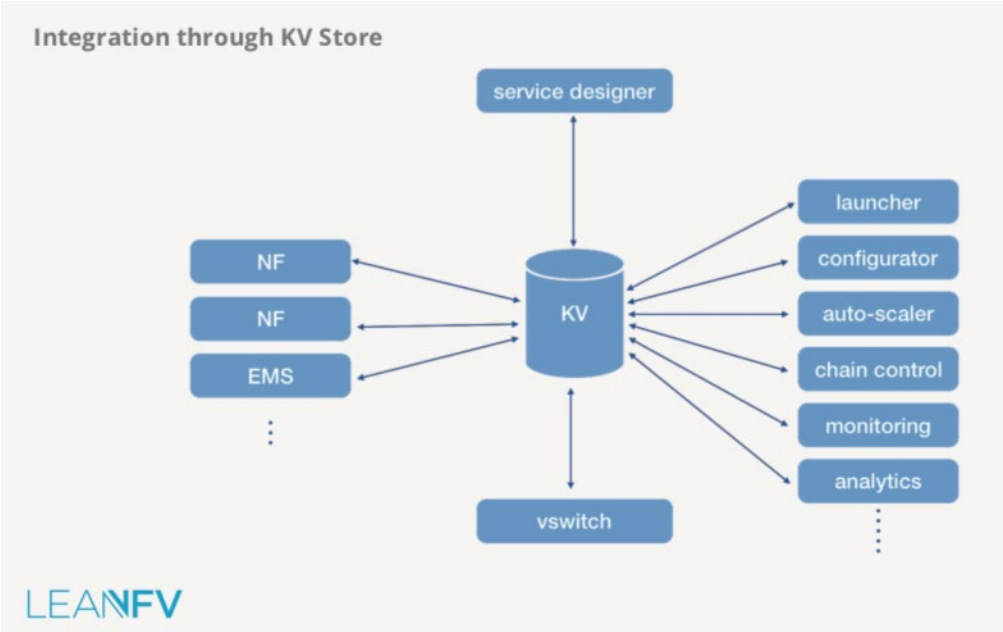

Figure 25 - LeanNFV Key Value store integration

The LeanNFV movement is off course still very immature and has certainly not gained wide acceptance yet in the industry. Nevertheless it addresses one of the core problems of the current NFV architecture, which will only become more relevant when NFV's are starting to migrate more and more to the extreme edge, E.g. CPE devices which will only run a limited environment for containers and where there is a necessity to decouple the life cycle management from the NFV implementation itself. As such, further API definition will serve as a guidance for NFV implementations to become truly cloud native while improving integration and addressing scalability concerns.

5. Conclusion, The future of residential CPE

"**The future** is already **here**. It's just not evenly distributed yet" is a famous quote from William Gibson, which perfectly captures the current state of technology around the further decentralization of cloud computing towards the edge and the softwarization that is happening due to network virtualization and software defined networking.

Despite the tremendous focus of the industry on the redesign of their core and access network, the acceleration that 5G will bring around MEC initiatives and the maturity around the white box ecosystem for enterprise and SOHO use cases with the introduction of universal CPE platforms, this decentralization is not a linear process.

In the shadow of all these evolutions new and compelling services and applications are getting directly integrated onto existing or next generation residential CPE equipment to enhance the home experience, being it network centric features or smart home, IoT centric evolutions. Albeit still vertically focused, resulting in siloed solutions, most of these providers have not waited for the operator's edge to materialize, and are working in a more top down approach using the public cloud as their backbone to deliver their end to end solutions, while heavily relaying on big data and analytics to augment their products while creating deep insight in user behavior and the home context.

In the current classical residential CPE architecture like home gateways network functions are deeply embedded inside the firmware in a monolithic fashion, while Soc's are deeply integrated and fully optimized in function of cost and forwarding capabilities, resulting in long integration and deployment cycles for new features.

With respect to network programmability there is no solutions fits all, hence today's evolving CPE market will have space for different shades of CPE.

The discriminating factor here is avoiding fundamental cost impact and the technology choice will be most likely made depending on the balance between Store and Forwarding needs of different use cases. This means that for devices like setop boxes where the cost impact is relatively small, DPDK will naturally have a bigger benefit for host originating traffic and storage driven applications, while for gateways which are heavily focused on forwarding use cases a better integration of OpenVswitch's data plane with SoC assisted hardware offloading will strike a better balance.

In order to be successful this process will be a gradual evolution from the existing legacy CPE architecture towards a more flexible and future safe platform rather than a revolution or big bang, which will have to strike a balance between offering compelling features and in home experiences and bringing them to market as fast as possible while leaving room for innovation and tackle the obstacles and technology gaps that still exists.

The main areas for improvements are better security and isolation, ultra-scalable orchestration and life cycle management for these new containerized components (pure application or network centric), improved network programmability of the SDN data patch in line with the hardware constraints of these type of devices (Bianchi & Bonola), a decentralizations and federation of the SDN controller path (Bhowmik & Tariq, 2015) and multi tenancy to allow different solutions to cooperate with each other and evolve to features like Software defined LAN and network slicing to offer true end to end QoS guarantees.

Due to the early stage of this evolution, this initial stage will be characterized with fragmentation in a lot of different initiatives, which is a typical cycle before the market will consolidate and mature standardization will set in. Today we already see different open-source initiatives from different vendors like Plume's Opensync and Broadband Forum USP providing different frameworks and others will follow.

To avoid the same pitfalls as witnessed in the SDN and network virtualization area which resulted in years of slow progress and severe fragmentation of the technology landscape the next figure will showcase a home gateway architecture that could drive an incremental approach and accelerate this evolution in drastic way by leveraging a natural evolution from the current legacy black box approach towards more of a "grey box". The figure also includes the basic building blocks to innovate on the network virtualization and SDN front.

The architecture is basically divided in three different areas that have distinct different life cycles in terms of design, development and go to market. (See figure)

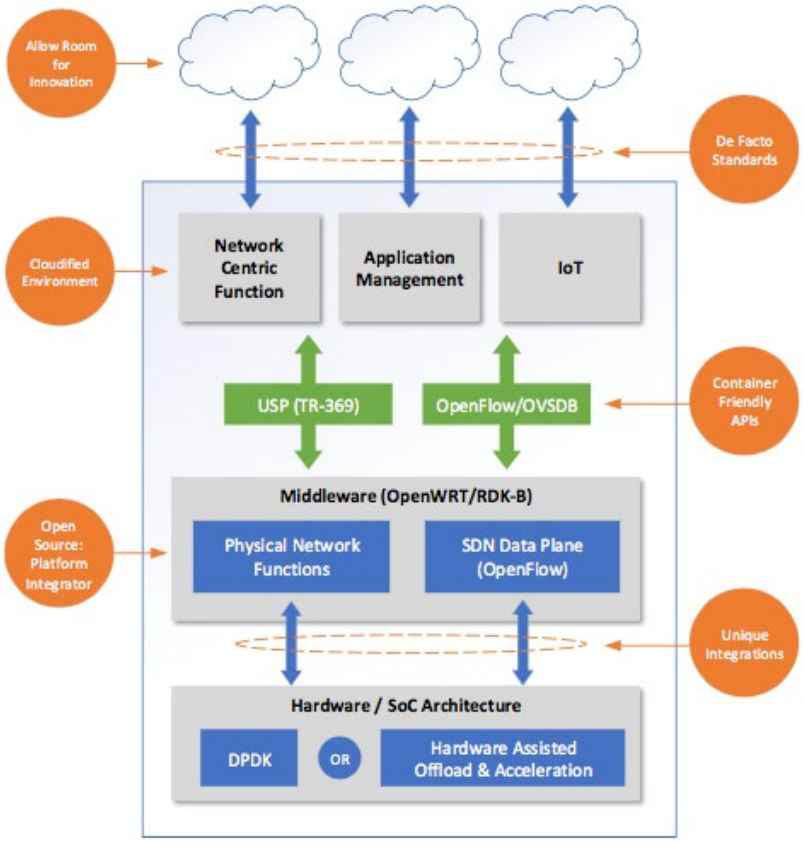

Figure 26 - Residential CPE architecture Layers

At the lowest layer this cycle is closely linked to the hardware evolution, which is bound to a very slow evolution of ASIC or SoC development which usually takes a couple of years to be developed, while the resulting products then have a life time in the field of approximately 4 to 7 years. Since this layer includes the "data path" it has a clear need to contain "data path" programmability capabilities as expressed by SDN or to be exploited by network functions.

While not the perfect solution, OVS seems to be to most likely and promising candidate here thanks to its mature and vibrant ecosystem and since it offers both a standardized northbound API (Openflow or OVSDB), while it also supports the low level interfaces that allow SoC vendors to offload the data path further down in their ASIC's. Please note that OVS is mainly limited towards level 2 forwarding, hence the industry will have to make an effort to extend OVS with more flexibility in this area while preserving the overall cost and performance of the total SoC cost. Ideally this should evolve to an industry specific domain programming language for the networking data path in line with the hardware constraint of these devices, being it something like P4 or a similar but optimized embedded version of it.

To allow faster experimentation of novel new approaches, the industry and research community should leverage ecosystems like off the shelf OpenWRT boxes or even trying to leverage the more Do it Yourselves (DIY) ecosystems like the once we see maturing around the Raspberry Pi which deliver hardware that can easily be extended.

The middle layer contains the current more traditional networking stacks that we find in the current generation of residential gateways. Although there is quite some fragmentation with different existing solutions and vendors, today we see a consolidation of two major ecosystems emerging. On one hand OpenWRT has played a dominant role in this space to deliver a community based opensource ecosystems of classical Linux based network components, while Comcast has a more syndicated and controlled ecosystem around RDK both for routers as for set-top boxes.

Typically, the life cycle at this level is characterized by lengthy feature development and integration cycles of approximately 6 to 9 months, which require rigorous testing and qualification cycles for radios like WiFi. Operators are typically very conservative in deploying these new releases in the field, and this problem is aggravated due to the rather old school remote management solutions like TR69 or SNMP, which are costly to scale and don't provide a lot of feedback and monitoring to seamlessly upgrade the installed base without introducing considerable risk.

Due to the rather monolithically design and implementation of this layer, this typical lengthy cycle cannot be fundamentally improved. Nevertheless, this layer today contains a lot of in home and access networking features which are mature and have been added over a decade of deployments.

Rather than trying to redesign these basic functions or even try to move them away towards the cloud or datacenter like in the original vCPE effort, this functionality should be leveraged in combination with the new upcoming SDN functionality since, as indicated in most of the SDN in the home research, most use case will require a combination of the two approaches. The control plane management of WiFi is a good example of that but basic NAT, DHCP and DNS are similar examples.

Today this middleware layer also includes support for life cycle management of different containerized execution environment, which will serve as the foundation of the next layer.

In terms of northbound API's to manage these build-in network functions or allow eventing for monitoring purposes they should be chosen in such a way that minimal intrusion or rework is necessary in this layer. One of the most promising API standards as candidate is the newly released broadband Forum specification called user service platform (USP), which addresses the shortcomings of the current TR69 standard with respect to multitenancy, provides more modern container friendly API's and leverage the existing TR69 data model.

At the highest layer, the experience to develop or integrate new features should be as frictionless as possible, mimicking the self-service experience that made development in cloud environments so popular. Hence the use of containerization at this layer to isolate these components from each other is a necessary step to take. Besides isolation which providers a better security model, this containerization also allows developers to use whatever technology or programming language they are most familiar while providing excellent portability decoupled from the Linux environment they are running on. A key challenge here is to provide the proper software life cycle management or orchestration of these components, which requires an ultra-scalable implementation of this management layer.

Today, most of the third parties are using their own design or implementation for this cloud communication layer. Examples are either in house developed implementation like Comcast Xmidt, existing more enterprise or datacenter originated solutions like OVSDB itself or some are using public cloud provider offerings like AWS IoT. The later could be a good inspiration as an ultra-scalable cost-effective solution.

Also, Broadband Forum has realized that there is not a solution that fits all, and hence has decoupled their USP implementation from the underlying communication protocol and design.

In order to not jeopardize the needed agility and innovation in this layer, vendors should have the flexibility to select their own implementation at this moment to avoid premature standardization. Nevertheless, the industry should start to follow closely the efforts from the LeanNFV movement in order to come to consensus on a set of APIs for this cloud communication that could both entail management and control and synchronizing of state for these components between the device and the cloud.

The key aspect of this layer is that it allows secure end to end ownership for third party solutions decoupled from the traditionally slower life cycle of the monolithic firmware in such a way that every of the 3 layers can evolve at their own specific pace.

Abbreviations

AA	Active Applications
API	Application Programming Interface
CapEx	Capital Expenditure
Cgroups	Control Groups
CNF	Cloud-native Network Functions
COAP	Constrained Application Protocol
CORD	Central Office Re-architected as a Datacenter
COTS	Commercial off-the-shelf
CPE	Customer Premise Equipment
CPU	Central Processing Unit
CSP	Communication Service Provider
DANOS	Disaggregated Network Operating System
DPDK	Data Plane Development Kit
EE	Execution Environment
ETSI	European Telecommunication Standards Institute
FPGA	Field-Programmable Gate Array
GNF	Glasgow Network Function
GPU	Graphics Processing Unit
ISP	Internet Service Provider
LXC	Linux Containers
L2GRE	Layer 2 General Routing Encapsulation
MANO	Management, Automation and Network Orchestration
MEC	Multi-access Edge Compute
NAT	Network Address Translation
NIC	Network Interface Card
NFV	Network Function Virtualization
OCP	Open Compute Platform
ONAP	Open Networking Automation Platform
ONF	Open Network Foundation
OLT	Optical Line Termination
OpEx	Operational Expenditure
OTT	Over The Top
OPNFV	Open Platform for NFV
OSM	Open Source MANO
QoS	Quality of Service
QoE	Quality of Experience
SDN	Software Defined Networking
SDWAN	Software Defined Wide Area Network
SOC	System On Chip
TCO	Total Cost of Ownership
TIP	Telecom Infrastructure Platform
USP	User Services platform
vCCAP	Virtual Converged Cable Access platform
vCPE	Virtual CPE

vEPC	Virtual Evolved Packet Core
vIMS	Virtual IP Multimedia Subsystem
VM	Virtual Machine
VPP	Vector Packeting Processing
vRAN	Virtual Radio Access Node
VXLAN	Virtual Extensible Local Area Network
SCTE	Society of Cable Telecommunications Engineers

Bibliography & References

5G_PPP. (2018). From Webscale to Telco, the Cloud Native Journey.

Alshnta, M. A., Mohd, F. A., & Al-Haiqi, A. (2018). SDN in the home: A survey of home network solutions using Software Defined Networking. Malysia.

Bhowmik, S., & Tariq, M. A. (2015). Distributed Control Plane for Software-defined Networks: A Case Study Using Event-based Middleware. Stuttgart.

Bianchi, G., & Bonola, M. (n.d.). Data Plane Programmability the next step in SDN. Roma.

Boussard, M. (2018). Future Spaces: Reinventing the Home Network for Better Security and Automation in the IoT Era. Nozay.

Calvert, K. (n.d.). Reflections on Network Architecture: an Active Networking Perspective. Kentucky.

CISCO. (2018). Cloud-Native Network Functions (CNFs) White Paper.

Cziva, R., & Pezaros, D. (2017). Container network functions: bringing NFV to the network edge. Glasgow: IEEE Communications Magazine.

ECI. (n.d.). The definitive guide to vCPE, Tempering expectations and makinf NFV a reality.

Ericsson. (2018). NFV Transformation Journey eBrief.

ETSI. (2017). NFV USe Cases.

Feamster, N., Rexford, J., & Zegura, E. (n.d.). The Road to SDN: An Intellectual History of Programmable Networks. princeton.

Hermesh, B., Shulman, S., & Ray, G. (2016). HEAD IN THE CLOUD, FEET ON THE GROUND: VIRTUALIZATION OF THE RESIDENTIAL GATEWAY.

Miettinen, M., Marchal, S., & Hafeez, I. (2016). IOT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT. Darmstadt.

Mortier, R., & Rodden, T. (2012). Control and Understanding: Owning Your Home Network. Nottinham.

Santamaria, C. j. (2017). Management of a heterogeneous distributed architecture with the SDN. Reims.

Sapien, M. (2016). Evaluating the trade-offs in applying NFV to enterprise services delivery. Ovum.

Yiakoumis, Y., Kok-Kiong, Y., Katti, S., & McKeown, N. (n.d.). Slicing Home Networks. Stanford.