Orchestrating the Delivery of End-User Services

by Jonathan Masel, CEO, Inango

The Evolving Network / Virtualization / End-User Services / Services and the Software Integration Challenge / Virtualizing Services / Data-Plane/Control-Plane / Containers / Orchestration and Openflow / Inango Virtual Services

The Evolving Network

Broadband communication technologies have evolved at a tremendous pace. Huge numbers of subscribers across the planet now enjoy connect rates in excess of 100Mbps, many have rates in excess of 1Gbps. And whether the infra-structure is telephone or cable, the push towards higher speeds continues.

This is obviously a huge driver for change and an enormous market opportunity for Service Providers. But it comes with challenges on both the cost and revenue sides.

Looking just at the CPE side of the equation, we can see rising costs, both CapEx and OpEx. Gateways deployed by the SP's are increasing in complexity and cost. This translates to not only higher purchase price for the equipment, but significantly higher OpEx costs in terms of maintenance and upgrades. Home networks are significantly more complex than they were several years ago with many more connected devices (computers, smartphones and IoT devices). Without a clear owner of the home network and all these devices, subscribers typically look to their SP's for support. All this constitutes high costs that SP's face in deploying and then maintaining the equipment that goes into our homes.

On the income side SP's face huge challenges too. Smartphones decimate revenue from fixed line telephones, OTT providers compete with TV or VOD packages. This creates the threat of SP's becoming providers of "dumb pipes" — communications networks of growing throughput rates, yet with little added differentiating value.

Recent years have seen much discussion about two concepts that are often promoted as solutions to the SP problems: virtualization and end-user services. In this paper, we will present various aspects of these technologies/trends, examine some of the underlying assumptions about their use and suggest ways in which they may be in tandem in a practical, yet highly beneficial way.

Virtualization

The Virtual CPE (vCPE) was seen as a way of reducing costs in deploying CPE gateways as well as simplifying their maintenance. In this context, the home gateway becomes a layer-2 device providing physical connectivity and minimal layer-2 networking services. Higher layer protocols that comprise the gateway software stacks (for NAT, DHCP, VoIP, management and much more) can all be moved to the Cloud with resulting reductions in both capital costs and simplified code maintenance.

The vCPE concept has expanded more recently to a uCPE the “universal” CPE (when it includes SD-WAN functions).

For residential and SMB users, though, there have not been huge traction for these technologies. The expected cost savings have not been fully realized as costs for CPE equipment are already quite low and driven by the cost of supporting the physical connections (rather than processors or memory). In fact, the cost of deploying more high-powered services is often found to be greater than any costs saved on the CPE equipment side.

But there are also industry initiatives to use virtualization for parts of the CPE (or other) offering: NFV (Network Function Virtualization) and Service Chaining (see ETSI standards) offer a way of moving some functionality from a CPE device to the cloud without necessarily virtualizing the entire gateway stack (and yes, these efforts can overlap).

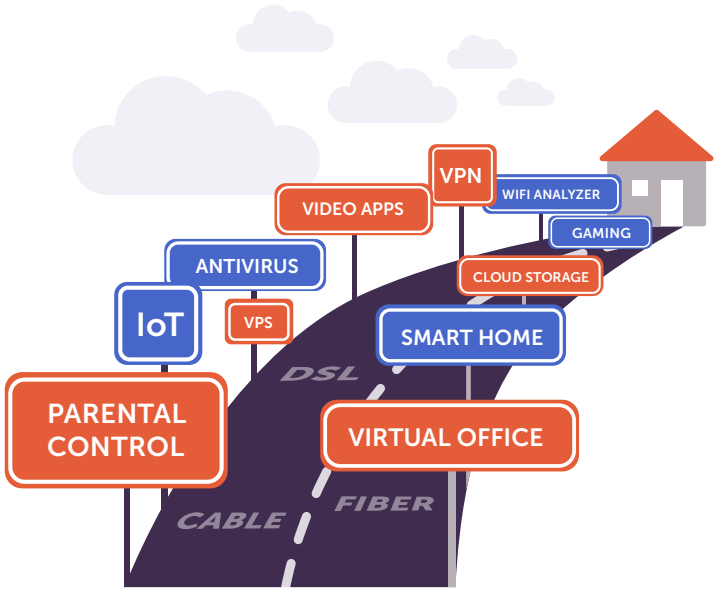
The primary goal of Service Chaining is to allow the off-loading of networking-related functions to the cloud. This may be useful in supporting functions such as intrusion prevention systems (IPS’s) where required processing may exceed the capabilities of the CPE gateway[1]. Or in theory it may be a way of reducing complexity of the gateway stack by offering a more systematic approach to the way in which functionality may be added to the gateway (physically or virtually).

We will re-visit Service Chaining and NFV later.

[1] We have seen companies attempt to move individual modules such as a DHCP Server to the cloud. It is not very clear what the perceived advantages are in this case — perhaps simpler code maintenance, although for code that is so broadly used and highly commoditized this does not always have great value.

End-User Services

Today nearly every Service Provider offers some sort of end-user service on top of Internet connectivity. We will loosely refer to these here as the "end-user services". We can think of several common types of such services: Anti-Virus (and anti-phishing, identity theft and so on), Parental Control services, IoT Protection, Cloud Storage, Wi-Fi Diagnostics and Smart Home applications. And there are many other opportunities for end-user services: Virtual Offices, pseudo-LANs spanning multiple sites, new ways of using standard personal cloud storage accounts, Multi-cast ABR for TV broadcasts and more.



In general, the driving force behind SP's offering these services is their market reach — collectively, they are in regular commercial contact with the entire population of homes (and SMB's) that is connected to the Internet. That is some statement. But there is more — the SP's also have the inbuilt ability to "see" all traffic entering and leaving the CPE premise since they usually control the CPE gateway device[2]. This means that they are uniquely positioned to offer certain types of end-user services. The first of these statements is a quantitative one (unparalleled reach), the second is qualitative (degree of visibility into our networks).

If we have a closer look at a partial list of end-user services above, we can see that they fall into different classes.

[2] To be accurate, this is not completely and universally true. Sometimes SP's only provide a modem and the users purchase off-the-shelf retail gateways or routers. However, it is true that all ingress/egress traffic flows through SP-provided equipment.

Anti-virus software really needs to run on end-point devices (computers, phones, etc) and not the gateway, since so much Internet traffic is encrypted (the gateways can offer some help, but only some). So anti-virus services offered by SP's is usually a commercial bundling ("Buy our Internet connection and get XXX anti-virus software for a good rate) rather than an integration of anti-virus services into the gateway functions.

IoT Protection, on the other hand, is well suited to run in the gateway. All traffic entering the home (or SMB office) can be inspected and checked for threats. If the service is good and robust, no more security software need be integrated in to the IoT devices themselves.

Parental Control services straddle both categories. In many cases, profiles may be defined for each end-point device and enforced at the gateway. Profiles may also be supported for all devices in the home (play stations, networked TV's and so on) where end-point licenses may not be available. And still, for smartphones and tablets, end-point licenses may also be used to provide Parental Control when these devices are outside the home.

Other services such as Smart Home apps and cloud storage are not related to network traffic — they only require connectivity. Integrating a Smart Home app into the gateway may save an additional box in the home (the controller) but runs as a network application and not part of the general CPE traffic.

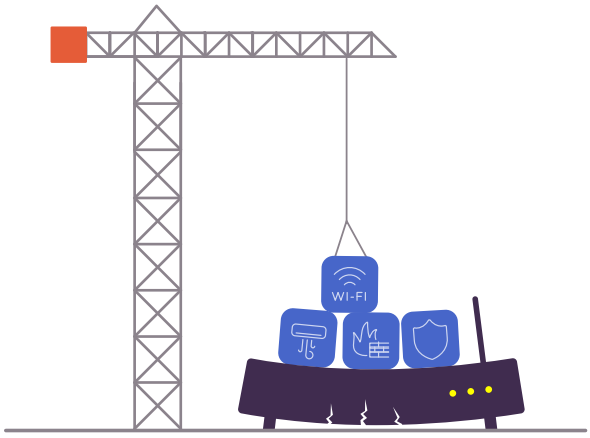
All of these are services that SP's charge their subscribers for use. There are others such as Wi-Fi Diagnostics that are used by the SP's themselves as a support tool, hopefully reducing OpEx costs and improving customer experience — but not something that customers are charged for.

Services and the Software Integration Challenge

What can we say about these services in general?

First, since the fundamental architecture of gateway software offerings has changed little in practice at least two decades, these services are typically integrated into the gateway software stack.

This has fundamental implications as to how services can be rolled out. Larger SP's usually have multiple suppliers of gateways, each with its own version upgrade schedule, variations and capabilities. Integrating more and more services into the gateway stack becomes an ever-increasing software nightmare: the more services that are added, the harder it is to add new ones, free memory is scarce, demarking areas of responsibility when bugs arise is a challenge, how to allocate resources between more competing services, how to keep offerings more-or-less consistent across a heterogenous offering of gateways and so on. Integrating a new service into an SDK will take several man-years of effort up-front and a considerable on-going expense in code maintenance.



As a side point, it should be noted that these are software-only issues. The same problems largely exist for vCPE platforms as well as for traditional, “physical” gateways.

Companies such as Intel already report a growth of one hundred-fold in gateway software sizes over the past years. Surely this is unsustainable.

Apart from the up-front and on-going costs, the system has built-in resistance to change. Getting a service integrated is hard enough — getting it out of an SDK or replacing it is just as difficult. This is excellent for the 3rd party providers of the software services, but bad news for SP’s and their subscribers alike. Products leapfrog each other in terms of functionality, quality and price, companies get bought or fold up — the industry is far more dynamic than such a monolithic architecture can support.

And underlying all of this is a worrying thought: Service Providers are responsible for providing Internet connectivity along with the basic gateway network functions at speeds that they sell. This is a contractual obligation. The more 3rd-party services that are integrated in the gateway, the greater is the risk that a bug in one service may prevent the gateway from providing its basic service. Despite considerable efforts — and costs — in QA and system testing, it is not possible to guarantee that this will not happen.

Virtualizing Services

What other approaches are Considered?

Network Function Virtualization (NFV) speaks of moving specific modules of functionality to the cloud. A related standard/technology is Service Chaining — or Service Function Chaining (SFC). One of the prominent pushes towards SFC standardization is ETSI, who has been working on this since 2012. Basically, network functionality can be portioned into modules and chained together in different ways to form different types of networking applications. There are schemes for distributing the processing over multiple servers, balancing loads between them and more.

There is a clear potential overlap between NFV/SFC and the issue raised earlier of integrating end-user services on gateways. If the issue of software integration is indeed unsustainable in the long run, perhaps we can apply some of the techniques of SFC to alleviate the issue. Do not try to virtualize the entire gateway (with a vCPE) but off-load some functions to the cloud and chain them together in a more modular, manageable way.

This may well be the way of the future, but we are not quite there yet.

Most discussion of services in SFC is around network services — I.e. services related to network traffic. These may be standard functions (such as DHCP, packet inspection, firewalls, etc) or system functions like IoT Protection, Parental Control and others. It is less clear that this model is useful for applications (such as Wi-Fi Monitoring, Smart Home apps, cloud storage and so on). The strong focus on networking functions is natural since NFV and SFC sprouted from SDN — which is a way of virtualizing networks and network switches (as opposed to gateway functionality). But it does mean that at least today the standards are not really addressing end-user services.

There is a hidden issue of hardware acceleration. Virtually all CPE gateway architectures support hardware acceleration for fast-path (or data plane) functions. Details will vary from one SoC to another, but the critical paths of packet forwarding are nearly always implemented in hardware. Any system of virtualization will need to preserve this capability — or system throughput will be compromised.

In any event, beyond the issue of SFC itself, is the orchestration of the system — the entity that can "join all the dots", configure packet flows using a consistent language, maintain concepts of end-users authentication and their desired preferences, keep logs and statistics, generate events and reports, control licensing and so on. These are functions that can be shared among all end-user services, but a comprehensive, robust framework is necessary.

Data-Plane/Control-Plane

So what is the way forward? Can CPE gateways benefit from virtualization technology and somehow reduce complexity of the software integration issues while improving the gateway's robustness in case any of the services malfunctions?

We offer an approach that does address these issues — in a very practical and beneficial way.

Before introducing the solution, let us discuss a few points in a little more detail.

The first is the division between control and data planes (or fast-path and slow-path) of any software module. The distinction sounds quite obvious (what you do often is data plane, what is done seldom is control), but looking at the dividing line between the control and data plane carefully can lead to some rather surprising results.

Take a service for IoT Protection, for example. A service looking for threats needs to examine any new connection to an IoT device. But it does not need to see all packets, just the first 10-15 packets of the new session. So, if I am doing a firmware upgrade to a device, the service can verify from the first N packets that it is a genuine firmware upgrade but there is no need (or point) in scanning the firmware itself. Same for intrusion inspection (just check the first packets in a session), same for parental control (just check which type of URL I am going to). And so on and so on.

As previously noted, most gateways can provide hardware acceleration for data path flows. Going back to our example of IoT Protection, we need to have the first 10-15 packets go through the processor (to the service) and once the session is declared fit, accelerate it through the available hardware blocks.

It is a little more complicated than defining the IoT service as a service function and chaining it to others — the flow of packets for the first 10-15 packets in a session is different from the remaining packets even though conceptually they are following the same functional chain.

Containers

The same issue, by the way, arises with containers. Many companies like to vaunt their code's modularity and flexibility simply because it is all containerized. We like containers and use them extensively, but a simplistic container approach is not enough: it will not address the data-plane/control-plane issue. Virtualization libraries (such as libvirt) may be used to give a container control over a network interface, but this will disable all hardware accelerations that may otherwise happen on these packets[3].

So how can we manage all the dynamic connections between various modules of code and yet still preserve the ability to accelerate parts when possible? Does this mean we are destined to continue coding all of the logic in the gateway code and compiling it all together along with the associated drawbacks as previously discussed?

Clearly, we don't think so. And the way we manage all of this is via OpenFlow – the protocol that underlies much of network virtualization in SDN and related technologies.

[3] It actually goes deeper than that. You cannot simply containerize a service if it needs to call device driver routines (to read Wi-Fi parameters, for example) or anything that actually "touches" the underlying gateway system.

Orchestration and OpenFlow

By playing with OpenFlow rules, we can achieve our desired results. We can to a very fine level of resolution direct packets that comprise a service's control plane to that service and allow the data plane packets to stay fully accelerated. For example, we can send DNS requests to a domain-based Parental Control service, or the first packets of a new session to an IoT Protection service. In fact, using standard OpenFlow rules we find that we can filter out virtually all control plane traffic for most services[4].

This allows us to achieve our primary goals of separating services from the SDK. Once packets are directed to a service via OpenFlow rules, it does not need to be compiled into the SDK — it can even run on a different server. The logic that defines the separation of control plane and data plane is embedded in the OpenFlow rules, not written in code binding the service with the SDK. Now, supporting a new service means defining the OpenFlow rules that are appropriate for it, swapping services in and out is a similar process. No more protracted software integration tasks are needed. It also gives us good isolation — if a server is running on a different server, it cannot interfere with the gateway's basic functions[5].

But what if we need to run the service locally, on the gateway, whether due to privacy considerations, severe latency constraints, or if the service needs to function when Internet connectivity is lost?

In these cases, we can run the service either in a virtual machine (VM) or in an LXC container on the gateway. Some Intel processors, for example, have built-in hardware support for VM's — we can exploit this by creating VM's for local services, connect them to the SDK via OpenFlow (exactly as if they were running on a remote server) and we get the best of both worlds: loose coupling, meaning no software integration beyond defining OpenFlow rules and isolation of basic gateway functionality from the 3rd party service on one hand and local execution on the other.

Inango Virtual Services

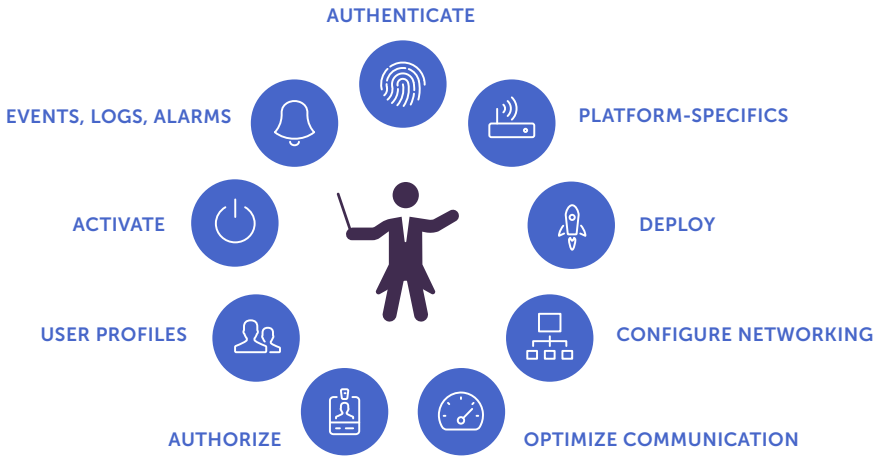
In order to manage all of this we need a piece of software that will manage (the commonly used term is "orchestrate") the deployment of 3rd party services in this way. In other words, a program that loads services (on a server or in a VM on the gateway[6]) and sets up the OpenFlow rules connecting it to the SDK. Then this orchestrator would need to

---

[4] There are some exceptions. Most notably, for services that require preserving state between packets. We have solutions for this too — a combined OpenFlow/P4 system — but that goes beyond the scope of this paper.

[5] Yes, we can think of counter-examples — if OpenFlow rules are not set up properly. But rules are kept in a specific place and can easily be removed. The risk involved is of an order of magnitude less that software bugs, memory corruptions and the like.

maintain a concept of users, authorize them to use the service, perhaps checking licensing or payment if necessary, set up that user's preferences for the service and activate it. And then, do things like optimizing packet flows, setting up tunnels between communicating entities, keeping detailed statistics, logs and generating reports. And then continually monitor the system, flagging any significant events of failures.



This (and more) is what is done with Inango's Virtual Service Launcher (VS Launcher for short) — it is a Service Orchestrator that enables Service Providers to offer their subscribers (residential or SMB) services via the gateways in exactly the way we have described in this paper. It allows for the fully automated deployment of services, their configuration and activation in the network.

The VS Launcher runs at a Service Provider's data center and manages the enablement of end-user services via the on-premise gateways but without the services needing to be tightly integrated with the gateway SDK[7].

We have also augmented the VS Launcher to include what we call a "Docker Launcher", to facilitate the support of 3rd party services completely unchanged. The Docker Launcher runs services in a docker and provides them with a Remote System Call library — the ability to seamlessly execute any system call remotely on the gateway. This is useful for services that need to directly interact with the gateway and its device drivers. A common example is Wi-Fi Diagnostics — the service needs to interface with the driver (via ioctls, sockets, read/writes, etc) to extract state information, execute specific commands and so on.

[6] We support full virtual machines, using Qemu, or LXC containers. Qemu provides maximum isolation and protection from bugs in the service, albeit at a higher price in memory footprint.

[7] As mentioned previously in this paper, the services themselves may be deployed on a server or in a VM on the gateway SoC. The choice between the two may be made when the service is deployed – or changed dynamically later.

The service itself then runs as-is on the VS Launcher — without even "knowing" that it is not physically running within the gateway.

What types of services do we find useful in such a system? There are the regular examples we have discussed above, such as IoT Protection, Parental Control, Wi-Fi Diagnostics and Smart Home apps. But opportunities are much broader — a Service Orchestrator gives the Service Provider an opportunity to offer far more services. It is akin to having a dedicated edge computer for each subscriber, part of the on-premise network yet requiring no resources there. We can offer SMB's a complete Virtual Office — cloud storage and cloud computers (VPS) all integrated and part of the office network. Or a pseudoLAN[8] and VPN service where off-the-shelf equipment can be used to provide connectivity for remote workers or full layer-2 connectivity for multi-sites without dedicated (and expensive) equipment or additional licenses.

For residential subscribers we can offer DLNA access to their own cloud storage (so home videos stored on Dropbox can be seen on a networked TV without having any client installed), We can offer gaming servers on the home network, so home owners can be saved the expense and hassle of buying their own servers and video cards. And in future versions we will be able to offer true DPI or video apps such as M-ABR.

Essentially, this amounts to providing the capabilities of edge computing for each subscriber. The sky really is the limit. All for the price of a single one-time integration of a tiny client (<1MB in flash) that is added to the gateway[9].

Once the VS Client (or equivalent) is in the gateway you will be able to deploy services in a fully automated way, without adding a single bit of code to the gateway for any specific individual service.

Implementation of the VS Launcher is based on an ONOS SDN controller that is enriched with several plug-ins and a control bus via which multiple micro-services are added. It includes support for large networks, with clustering, redundancy and load balancing. And there are integrated mechanisms for expedited handling of large volumes of user data as well (ONOS is written in Java and not designed for speed — for data-hungry services we found this to be an issue).

---

[8] pseudoLAN is what we call a service of ours that combines several offices in to a seamless, single layer-2 network. It is like an MPLS pseudo-wire service (e.g. for SD-WAN), but runs on standard gateways with no dedicated hardware, is completely agnostic to the network topology (star or full mesh) and can span multiple provider networks transparently.

[9] We can also work with gateways that are equipped with an OpenSync client instead of our VS Client. It will not work with every service (there is no Remote System Call capability) but for traffic-related services there will be no loss of functionality.