Central Office Re-architected as a Datacenter (CORD)

[This is an updated version of the original CORD white-paper which first appeared in June 2015.]

March 14, 2016

Challenges

Network operators face significant challenges supporting ever-increasing bandwidth demands and ever-increasing service expectations. For example, AT&T has seen data traffic increase by 100,000 percent in the last eight years, and plans are now underway to roll out ultra-fast fiber and access to 100 cities across the US [1]. At the same time, introducing a new feature often takes months (waiting for the next vendor product release) and sometimes years (waiting for the standardization process to run its course).

In response to these challenges, network operators are looking for ways to benefit from both the economies of scale (infrastructure constructed from a few commodity building blocks) and the agility (the ability to rapidly deploy and elastically scale services) that commodity cloud providers enjoy today.

Cloud economies and agility are especially needed at the edge of the operator network—in the Telco *Central Office (CO)*—which contains a diverse collection of purpose-built devices, assembled over fifty years, with little coherent or unifying architecture. For example, AT&T currently operates 4700 Central Offices, some of which contain up to 300 unique hardware appliances. This makes them a source of significant CAPEX and OPEX, as well as a barrier to rapid innovation.

This paper describes *CORD*, an architecture for the Telco Central Office that combines *Software-Defined Networking (SDN)*, *Network Functions Virtualization (NFV)*, and

elastic cloud services—all running on commodity hardware—to build cost-effective, agile networks with significantly lower CAPEX/OPEX and to enable rapid service creation and monetization.

Introducing CORD

CORD re-architects the Central Office as a datacenter. The basic approach centers on unifying the following three related but distinct technology trends:

- The first is SDN, which is about separating the network's control and data planes. This makes the control plane programmable, and that can lead to increased innovation. It also allows for simplification of forwarding devices that can be built using merchant silicon, resulting in less expensive white-box switches.
- The second is NFV, which is about moving the data plane from hardware devices to virtual machines. This reduces CAPEX costs (through server consolidation and replacing high-margin devices with commodity hardware) and OPEX costs (through software-based orchestration). It also has the potential to improve operator agility and increase the opportunity for innovation.
- The third is the Cloud, which defines the state-of-the-art in building scalable services—leveraging software-based solutions, micro-service architecture, virtualized commodity platforms, elastic scaling, and service composition, to enable network operators to rapidly innovate.

While it is easy to see that all three threads (SDN, NFV, Cloud) play a role in reducing costs, it is just as important to recognize that all three are also sources of innovative (and revenue generating) services that Telcos can offer subscribers. These include control plane services (e.g., content-centric networking, virtual networks on demand, cloud-network binding), data plane services (e.g., Parental Control, NAT, WAN Acceleration), and global cloud services (e.g., CDN, Storage, Analytics, Internet-of-Things).

In short, the goal of CORD is not only to replace today's purpose-built hardware devices with their more agile software-based counterparts, but also to make the Central Office an integral part of every Telco's larger cloud strategy, enabling them to offer more valuable services. This means CORD's software architecture must be general enough to support a wide-range of services. This includes both access services (e.g., Fiber-to-the-Home) and scalable cloud services (SaaS); services implemented in the data plane (NFV) and services implemented in the control plane (SDN); trusted

operator-provided services and untrusted third-party services; and bundled legacy services and disaggregated greenfield services.

Commodity Hardware

The target hardware for CORD consists of a collection of commodity servers interconnected by a fabric constructed from white-box switches. The illustrative example shown in Figure 1 highlights two salient features of the hardware configuration:

- The switching fabric is organized in a leaf-spine topology to optimize for traffic flowing east-to-west—between the access network that connects customers to the Central Office and the upstream links that connect the Central Office to the operator's backbone. There is no north-south traffic in the conventional sense.
- The racks of GPON OLT MACS commoditize connectivity to the access network. They replace proprietary and closed hardware that connects millions of subscribers to the Internet with an open, software-defined solution. (GPON is the example access technology discussed in this paper, but the same argument applies to other access technologies as well.)

Figure 1. Target hardware built from commodity servers, I/O Blades, and switches.

We are building a reference implementation of CORD that includes specific choices for all the hardware elements, organized into a rackable unit called a POD. The reference CORD POD consists of:

- Open Compute Project (OCP) qualified QUANTA STRATOS-S210-X12RS-IU servers, each configured with 128GB of RAM, 2x300GB HDDs, and a 40GE dual port NICs.
- OCP-qualified and OpenFlow-enabled Accton 6712 switches, each configured with 32x40GE ports. These switches serve as both leaf and spine switches in the CORD fabric.

- Celestica is the ODM for “OLT pizza box” I/O blades that include the PMC Sierra OLT MAC chip. This is a 1u-blade that includes 48x1Gbps GPON interfaces and 2x40GE uplinks.

The servers run Ubuntu LTS 14.04 and include Open vSwitch (OvS). The switches run the Atrium software stack [2], which includes Open Network Linux, the Indigo OpenFlow Agent (OF 1.3), and the OpenFlow Data Plane Abstraction (OF-DPA), layered on top of Broadcom merchant silicon. Although not included in the initial version, I/O blades for 10G-PON and G.Fast are planned for the near future.

Note that different configurations of the POD hardware are possible. For example, the selected leaf switches have sufficient capacity to support up to 24 dual-port servers and the spine switches have sufficient capacity to support up to 16 racks. At the other end of the spectrum, it is also possible to configure a “micro POD” that includes only leaf/ToR switches and fits in a partial rack.

Software Building Blocks

With respect to software, our reference implementation of CORD exploits four open source projects, as depicted in Figure 2:

- **OpenStack** [3] is the cluster management suite that provides the core IaaS capability, and is responsible for creating and provisioning virtual machines (VMs) and virtual networks (VNs).
- **Docker** [4] provides a container-based means to deploy and interconnect services. It also plays a key role in configuring and deploying CORD itself (e.g., the other elements—XOS, OpenStack, and ONOS—are instantiated in Docker containers on the POD head nodes.
- **ONOS** [5] is the network operating system that manages the underlying white-box switching fabric. It both hosts a collection of control applications that implement services on behalf of Telco subscribers and is responsible for embedding virtual networks in the underlying fabric, which is in turn accessed via OpenStack’s Neutron API.
- **XOS** [6] is a framework for assembling and composing services. It unifies infrastructure services (provided by OpenStack), control plane services (provided by ONOS), and any data plane or cloud services (running in OpenStack-provided virtual machines and Docker-provided containers).

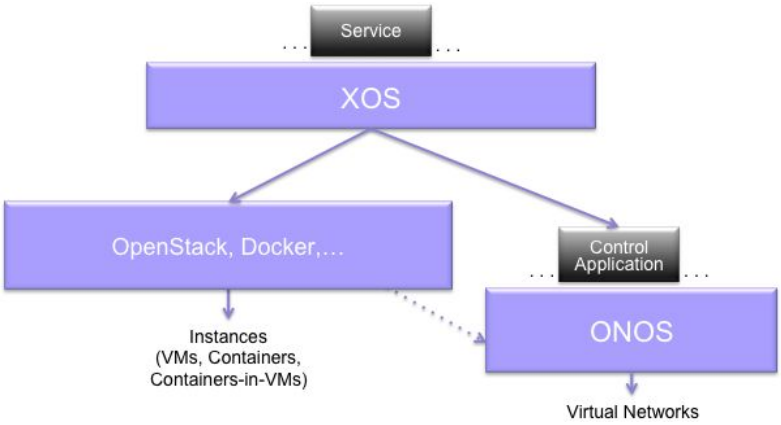

Figure 2. Open source software components used to build CORD.

Note that to allow for the widest possible collection of services, the reference implementation supports services running in virtual machines (KVM), in containers running directly on bare metal (Docker), and in containers nested inside virtual machines (Docker in KVM). To simplify the exposition, we use the term “Virtual Machine (VM)” for all three variants, unless the specific implementation is relevant to the discussion.

Also note that ONOS plays two distinct roles in CORD. It both interconnects VMs (this includes implementing VNs and managing flows across the switching fabric), and it provides a platform for hosting control programs that implement first-class CORD services. Two examples of the latter are described in the next section (vOLT and vRouter).

Transformation Process

Given this hardware/software foundation, transforming today’s Central Office into CORD can be viewed as a two-step process. The first step is to virtualize the devices, that is, turn each purpose-built hardware device into its software counterpart running on commodity hardware. The key to this step is to disaggregate and refactor the functionality bundled in the legacy devices, deciding what is implemented in the control plane and what is implemented in the data plane.

The second step is to provide a framework that these virtualized software elements—along with any cloud services that the operator wants to run in the Central Office—can be plugged into, producing a coherent end-to-end system. This framework starts with the components just described, but also includes the unifying abstractions that forge this collection of hardware and software elements into a system that is economical, scalable, and agile. The following two sections describe these two steps to re-architect the Central Office in greater detail.

Virtualizing Legacy Devices

The first step in re-architecting the central office as a datacenter involves virtualizing the existing hardware devices, transforming each legacy device into its commodity hardware plus software service counterpart. In the process, functionality is likely disaggregated and re-packaged in new ways.

This section walks through the process for the devices highlighted in red in Figure 3, which includes Optical Line Termination (OLT), Customer Premises Equipment (CPE), and Broadband Network Gateways (BNG). We don't virtualize the Ethernet switch, per se, but it is effectively replaced by the switching fabric in Figure 1, under the control of ONOS.

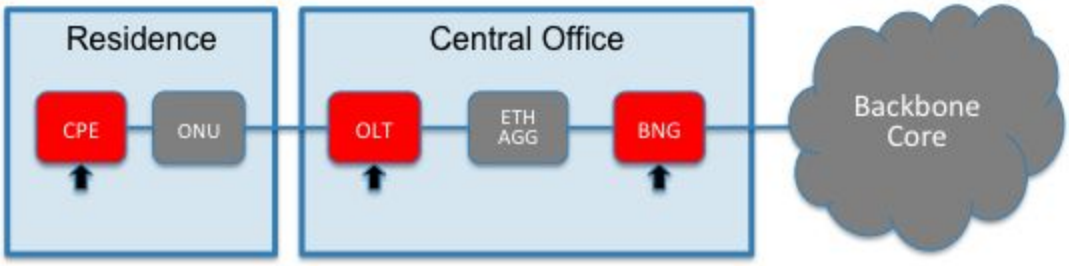

Figure 3. Legacy Central Office, including three physical devices to be virtualized.

This paper focuses on GPON technology and virtualizing the relevant pieces such as OLT. However the underlying principles are equally applicable to other access technologies including 10GPON, copper-based G.Fast networks, cable DOCSIS networks, and BBU mobile networks.

Benefits and Challenges

OLT is a large capital investment, involving racks of closed and proprietary hardware that terminate access for tens of thousands of subscribers per CO. Virtualizing OLT is especially challenging because, unlike many network appliances that are implemented by software running on vendor-branded commodity servers, OLT is implemented primarily in hardware. CPEs are currently distributed to tens of thousands of customer sites per CO, making them a significant operational burden. This is especially true when a service upgrade requires a hardware upgrade. BNGs are expensive and complex routers that have historically aggregated much of the functionality provided by a Central Office, making them difficult to evolve in an agile and cost-effective way.

The challenge is how to systematically transform such a diverse collection of devices into software running on commodity hardware. Our main insight is that there is a simple template for how each physical device is virtualized. It includes a combination of three

elements: (1) merchant silicon, including both commodity servers and white-box switches; (2) a control plane function, which we refer to as the SDN element; and (3) a data plane function, which we refer to as the NFV element. While SDN and NFV are over-loaded terms, for our purposes, both are implemented by software running on commodity servers, where it is considered an NFV element if packet processing is entirely in software, and it is considered an SDN element if that software also controls commodity switches and I/O blades through an open interface like OpenFlow. Said another way, NFV elements run *on* commodity hardware, while SDN elements run on commodity servers but also *control* commodity hardware (switches).

The rest of this section shows how this pattern is applied to OLT, CPE, and BNG, resulting in virtual incarnations of each physical device. It is not the goal to preserve a one-to-one mapping between physical and virtual devices, and in fact, the opposite is true. Virtualizing legacy hardware provides an opportunity to disaggregate and refactor their functionality. Examples of such refactoring are presented throughout this section.

Virtualizing the OLT

OLT terminates the optical link in the Central Office, with each physical termination point aggregating a set of subscriber connections. Given the number and cost of OLT devices in a CO, virtualizing the OLT has the potential to yield significant CAPEX and OPEX savings.

The first challenge is to create an I/O Blade with the PON OLT MAC, and to this end, AT&T has worked with the Open Compute Project to develop an open specification for a GPON MAC 1RU “pizza box”. This board includes the essential GPON Media Access Control (MAC) chip under control of a remote control program via OpenFlow.

These I/O blades are then brought under the same SDN-based control paradigm as the white-box based switching fabric. The resulting control program, called *virtual OLT (vOLT)*, runs on top of ONOS, and implements all other functionality normally contained in a legacy OLT chassis (e.g., GPON protocol management, 802.1ad-compliant VLAN bridge). That is, vOLT implements authentication on a per-subscriber basis, establishes and manages VLANs connecting the subscriber’s devices to the Central Office switching fabric, and manages other control plane functions of the OLT.

Virtualizing the CPE

A CPE, sometimes called a “home router” or “residential gateway,” is installed in the customer’s premises. Because of their numbers, they are a significant source of CAPEX and OPEX costs, as well as a barrier to introducing new services. They often run a collection of essential functions (e.g., DHCP, NAT) and optional services (e.g., Firewall,

Parental Control, VoIP) on behalf of residential subscribers. More sophisticated enterprise functions are also common (e.g., WAN Acceleration, IDS), but this paper focuses on residential functions. By extending the capabilities of CPE in the cloud, new value-add services as well as customer care capabilities can be provided where they could not before because of limitations in the hardware.

Our virtualized version of CPE, called *virtual Subscriber Gateway (vSG)*, also runs a bundle of subscriber-selected functions, but it does so on commodity hardware located in the Central Office rather than on the customer's premises. There is still a device in the home (which we still refer to as the CPE), but it can be reduced to a bare-metal switch, with all the functionality that ran on the original CPE moved into CO and running in a VM on commodity servers. In other words, the "customer LAN" includes a remote VM that resides in the central office, effectively providing every subscriber with a direct ingress into the Telco's cloud.

CORD permits a wide range of implementation choices for subscriber bundles, including a full VM, a lightweight container, or a chain of lightweight containers. Our approach is to treat the bundle as a whole (roughly corresponding to a VM image or a container configuration) as the standard representation, and leave the means by which subscribers select the set of functions to be included in their bundle as an implementation choice. For example, our reference implementation gives subscribers the ability to select from a small collection of functions (e.g., DHCP, NAT, firewall, parental filtering), but implements each through the proper configuration of a container (as defined by a corresponding Dockerfile). Experiments show this approach conservatively supports 1000 subscribers per server.

Virtualizing the BNG

A BNG is one of the more complex and expensive devices in a Central Office, providing the means through which subscribers connect to the public Internet. It minimally manages a routable IP address on behalf of each subscriber, and provides that subscriber with some type of network connectivity. In practice, however, the BNG also bundles a large collection of value-added features and functions, including VPNs, GRE tunneling, MPLS tunneling, 802.1ad termination, and so on.

CORD's virtualized BNG, called *virtual Router (vRouter)*, is implemented as an ONOS-hosted control program that manages flows through the switching fabric on behalf of subscribers. No attempt is made to reproduce many of the auxiliary functions historically bundled into a BNG device, although in some cases (e.g., authenticating subscribers), that functionality is provided by another service (e.g., vOLT, in our case).

In general, it is more accurate to think of vRouter as providing each subscriber with their own “private virtual router,” where the underlying fabric can be thought of as distributed router with “line cards” and “backplanes” instantiated by bare-metal switches. The vRouter control program then creates an IP network that routes between the attached, per-subscriber subnets. vRouter also peers with legacy routers, for example, advertising BGP routes.

Our approach to vRouter highlights an example of refactoring. Historically, BNG is also responsible for authenticating the user, but that capability has been unbundled and moved to vOLT. This is because subscribers have to be authenticated *before* accessing vSG, which use to reside in the home but has now moved into the Central Office.

End-to-End Packet Flow

We conclude by sketching a subscriber’s packet flow through the CORD, assuming the subscriber already has an account with the Telco. When the subscriber powers up the home router, an 802.1x authentication packet is sent over GPON to the CO. Upon arrival at an I/O blade port, the packet is passed up (through ONOS) to the vOLT control program, which authenticates the subscriber using an account registry like RADIUS. Once authenticated, vOLT assigns VLAN tags to the subscriber and installs the appropriate flow rules in the I/O blade and switching fabric (via ONOS), asks vSG to spin up a container for that subscriber, and binds that container to the VLAN. vSG, in turn, requests a routable IP address from vRouter, which causes vRouter (via ONOS) to install the flow rules in the switching fabric and software switches needed to route packets to/from that subscriber’s container.

Once set up, packets flow from the home router over a VLAN to the subscriber’s container, those packets are processed according to whatever bundle is associated with the subscriber’s account (and configured into the container), and then forwarded on to the Internet using the assigned source IP address.

This description is obviously high-level, glossing over both many low-level details about each component (e.g., how VLAN tags are assigned, precisely what flow rules are installed, the exact composition of functions in each container) and the mechanisms by which the various agents (ONOS, OpenStack, Docker, vOLT, vSG, vRouter) are actually plumbed together. More information on the former is available in a set of engineering design notes [7]; more information on the latter is given in the next section.

Service Framework

This section focuses on the second step in re-architecting the Central Office as a datacenter: orchestrating the software elements resulting from the first step (plus any

additional cloud services that the operator wants to run there) into a functioning and controllable end-to-end system.

Benefits and Challenges

Replacing hardware devices with software running in virtual machines is a necessary first step, but is not by itself sufficient. Just as all the devices in a hardware-based Central Office must be wired together in a meaningful way, their software counterparts must also be managed as a collective. This process is often called *service orchestration*, but if network operators are to enjoy the same agility as cloud providers, the abstractions that underlie the orchestration framework must fully embrace (1) the elastic scale-out of the resulting virtualized functionality, and (2) the composition of the resulting disaggregated (unbundled) functionality. A model that simply "chains" VMs together as though it is operating on their hardware-based counterparts will not achieve either goal.

Our approach is to adopt *Everything-as-a-Service (XaaS)* as a unifying principle [6]. This brings the disparate functionality introduced by virtualizing the hardware devices under a single coherent model. The control functions run as scalable services (these functions run on top of ONOS, a scalable network operating system), the data plane functions run as scalable services (these functions scale across a set of VMs), the commodity infrastructure is itself managed as a service (this service is commonly known by the generic name IaaS), and various other global cloud services running in the Central Office are also managed as scalable services (as outlined below, CORD includes a virtualized CDN as an illustrative example).

Scalable Services, Not Virtual Devices

While the terms vOLT, vSG, and vRouter are used to refer to the virtualized counterpart of the three physical devices, they are all three packaged as services. While we could name these services according to their legacy counterparts, the new architecture no longer requires functionality to be bundled along the same boundaries as before. For this reason, it is more intuitive to think of the virtualization process outlined above as resulting in three generic, multi-tenant services:

- vOLT is a control program running on ONOS. It implements *Access-as-a-Service*, where each tenant corresponds to a *Subscriber VLAN*.
- vSG is a data plane function scaled across a set of containers. It implements *Subscriber-as-a-Service*, where each tenant corresponds to a *Subscriber Bundle*.

- vRouter is a control program running on ONOS. It implements *Internet-as-a-service*, where each tenant corresponds to a *Routable Subnet*.

If a Content Distribution Network (CDN)—itself a scalable cloud service deployed throughout the operator's network, including caches in the Central Offices—is added to these three new services, we have an example of the three kinds of services outlined in the introduction: a cloud service (CDN), a data plane service (vSG), and two control plane services (vOLT and vRouter).

This results in the legacy Central Office depicted in Figure 3 being re-architected into the service graph shown in Figure 4. To illustrate CORD's generality as a configurable platform, Figure 4 also includes a *vG.Fast* service to represent a second possible access technology.

Each CORD service is multi-tenant and provides some service abstraction, in the same sense that a conventional cloud storage service provides a "Volume" abstraction and a NoSQL DB service provides a "KeyStore" abstraction. As such, we we read the service graph in Figure 4 as: "a subscriber acquires a subscriber VLAN from vOLT, which in turn acquires a subscriber bundle from vSG, which finally acquires a routable subnet from vRouter."

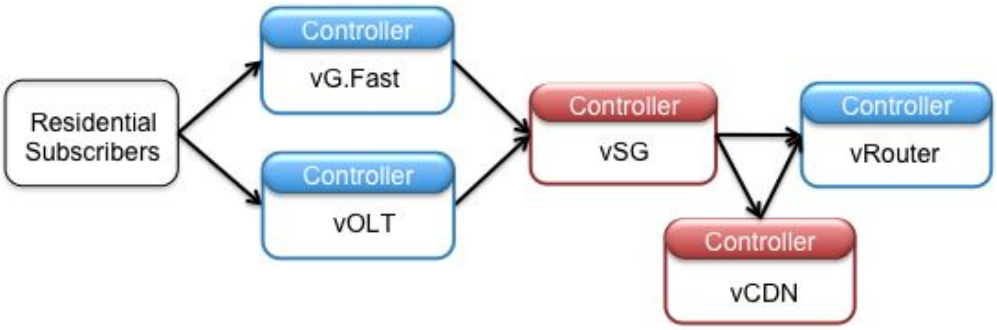

Figure 4. CORD service graph, including two access services: vOLT and vG.Fast.

It is also accurate to say that the subscriber is a tenant of the service graph as a whole. Pragmatically, this means that once the service graph shown in Figure 4 is configured into CORD, the subscriber is able to control his or her subscription (e.g., set the parental control feature to disallow access to Facebook) by invoking operations on the subscriber object, without any awareness of which service implements which feature. The structure imposed by the XOS abstractions maps such a request onto the right set of components.

Note that the service graph shown in Figure 4 is simplified to focus on the services that provide direct value to end users. There are also a collection of building block services

upon which the services in Figure 4 depend. This includes ONOS and OpenStack (e.g., there is a tenant dependency between both vOLT and vRouter and ONOS), as well as a monitoring service that collects and aggregates meters from each hardware and software element in CORD, and delivers them to analytic engines for fault diagnostics and dynamic steering.

Layers of Abstraction

The service graph shown in Figure 4 represents the high-level specification a network operator would provide, but this specification has to be mapped onto the underlying servers, switches, and I/O blades. How this happens is a direct consequence of a nested collection of abstractions that CORD layers on top of the building block components shown in Figure 2. These abstractions impose a structure on the set of services in a way that allows operators to express and enforce policies on them.

Working top down, CORD defines the following abstractions (and corresponding mechanisms that implement them):

- *Service Graph:* Represents a set of dependency relationships among a set of Services (see next). CORD models service composition as a *Tenancy* relationship between a provider service and a tenant (i.e., client) service. Service tenancy is anchored in a *Tenant Principal* (e.g., subscriber) that is bound to one or more *User Accounts*.

- *Service:* Represents an elastically scalable, multi-tenant program, including the means to instantiate, control, and scale functionality. CORD models a service as a *Service Controller* that exports a multi-tenant interface and an elastically scalable set of *Service Instances* that are collectively instantiated in a *Slice* (see next). XOS includes mechanisms to assemble a CORD-ready service from both greenfield and legacy components.

- *Slice:* Represents a system-wide resource container in which Services execute, including the means to specify how those resources are embedded in the underlying infrastructure. CORD models a slice as a set of *Virtual Machines (VMs)* and a set of *Virtual Networks (VNs)*. VMs are implemented by the underlying IaaS components (OpenStack and Docker), while VNs are implemented by ONOS (see next).

- *Virtual Network:* Represents a communication interconnection among a set of instances. CORD supports several VN types, including *Private* (connects instances within a Slice), *Access_Direct* (used by a tenant service to access a provider service by directly addressing each instance in the provider service),

and *Access_Indirect* (used by a tenant service to access a provider service by addressing the service as a whole). The latter two types support service composition.

The mechanism underlying CORD's support for Virtual Networks is implemented by a pair of control applications that run on top of ONOS. The first, called VTN, installs flow rules in the OvS running on each server to implement direct or indirect addressing. The second, called Segment Routing, implements aggregate flows between servers across the switching fabric.

Looking at Figure 4 through the lens of NFV, each tenant abstraction in CORD corresponds to a Virtualized Network Function (VNF) in the NFV architecture [8]. How a sequence of such VNFs (a service chain) map onto a sequence of VMs depends on three things: (1) whether the service is implemented on the network control plane or in the network data plane, (2) how each service maps its tenants onto one or more service instances, and (3) what type of virtual network interconnects service instances.

Said another way, a linear chain of instances corresponding to single subscriber—an implementation of service chaining assumed by many service orchestrators—is just one of many possible outcomes of service composition in CORD. Our experience with a wide collection of services that span the full NFV, SDN, and Cloud space is that a more general model of service composition is required, and this experience informs CORD's design.

Future Plans

CORD is a revolutionary effort to transform legacy Central Offices in the Telco network. In the new Central Office re-architected as a datacenter, closed and proprietary hardware is replaced with software running on commodity servers and switches. This software, in turn, is managed and orchestrated as a collection of scalable services. In doing so, CORD's goal is to demonstrate the feasibility of a Central Office that enjoys both the CAPEX and OPEX benefits of commodity infrastructure, and the agility of modern cloud providers.

We are building a reference implementation of CORD, with plans to deploy it in a residential field trial at AT&T. The reference implementation has been informed by the specific requirements of the field trial, it is designed to be a general platform that can be configured for a wide range of deployment scenarios. For example, a configuration targeted at Mobile users (M-CORD) and another targeted at Enterprise users (E-CORD) are already in the works.

Open source software plays a critical role in CORD, which leverages: OpenStack and Docker to provision the virtual virtual compute instances, ONOS to manage the switching fabric and host control applications, and XOS to assemble and compose services. Atrium and the Open Compute Project are also valuable parts of CORD.

References

[1] Delivering a Software-based Network Infrastructure. Krish Prabhu. AT&T Labs (October 2015).

[2] Atrium: A Complete SDN Distribution from ONF. http://onosproject.org/wp-content/uploads/2015/06/PoC_atrium.pdf (2016).

[3] OpenStack: Open Source Cloud Computing Software. https://www.openstack.org/ (2016).

[4] Docker: Build, Ship, Run Any App, Anywhere. https://www.docker.com/ (2016).

[5] ONOS: Towards an Open, Distributed SDN OS. *HotSDN 2014.* P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, G. Parulkar (August 2014).

[6] XOS: An Extensible Cloud Operating System. *ACM BigSystems 2015*. L. Peterson, S. Baker, A. Bavier S. Bhatia J. Nelson, M. Wawrzoniak, M. De Leeneer, and J. Hartman (June 2015).

[7] CORD: Re-inventing Central Offices for Efficiency and Agility. http://opencord.org (2016).

[8] Network Functions Virtualization—An Introductory White Paper. *SDN and OpenFlow World Congress* (October 2012).