SDN Use Cases for Service Provider Networks

Central Office Re-Architected as a Data Center

Larry Peterson, Ali Al-Shabibi, Tom Anshutz, Scott Baker, Andy Bavier, Saurav Das, Jonathan Hart, Guru Palukar, and William Snow

CORD is a new design of a telco central office that replaces closed and proprietary hardware with software running on commodity servers, switches, and access devices. It allows network operators to benefit from both the economies of scale (infrastructure constructed from a few commodity building blocks) and agility (the ability to rapidly deploy and elastically scale services) that commodity cloud providers enjoy today.

Abstract

CORD is a new design of a telco central office that replaces closed and proprietary hardware with software running on commodity servers, switches, and access devices. It allows network operators to benefit from both the economies of scale (infrastructure constructed from a few commodity building blocks) and agility (the ability to rapidly deploy and elastically scale services) that commodity cloud providers enjoy today. This article introduces the CORD architecture and describes an open reference implementation of CORD that is available for evaluation.

Challenges

Network operators face significant challenges meeting bandwidth demands and providing enhanced services. 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 United States [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, network operators are looking for ways to benefit from both the economies of scale (infrastructure constructed from a few commodity building blocks) and agility (the ability to rapidly deploy and elastically scale services) that commodity cloud providers enjoy.

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 50 years, with little coherent or unifying architecture. For example, AT&T currently operates 4700 COs, some of which contain up to 300 unique hardware appliances. This makes them a source of significant capital expenditure (CAPEX) and operational expenditure (OPEX), as well as a barrier to rapid innovation.

This article describes CORD, an architecture for the telco CO (or cable head-end) that combines software-defined networking (SDN), network functions virtualization (NFV), and elastic cloud services to build cost-effective, agile access networks. In addition to introducing the architecture, this article also outlines an open reference implementation of CORD that is available for evaluation.

Introducing CORD

CORD re-architects the CO as a data center, and in doing so unifies three related but distinct technology trends.

SDN: Separates the network's control and data planes and makes the control plane programmable. This simplifies network infrastructure and permits inexpensive white-box switches that can be built using merchant silicon.

NFV: Moves the data plane from hardware devices to virtual machines (VMs) running on commodity servers. This replaces high-margin devices with commodity hardware and permits more agile software-based orchestration.

Cloud: Defines best practices in scalable services, leveraging software-based solutions, micro-services, virtualized commodity platforms, elastic scaling, and service composition. This allows network operators to innovate more rapidly.

In unifying these three trends, CORD goes beyond what each technology contributes in isolation:

- It uses SDN not only to simplify the networking infrastructure, but also as a source of innovative services that can be offered to customers.
- It not only supports legacy network functions in VMs, but also disaggregates functionality into finer-grained elements.
- It not only supports conventional SaaS, but also extends the cloud paradigm to include fiber to the home and wireless access as elastic access as a service.

In short, CORD's goal is to not only replace today's purpose-built hardware devices with their more agile software-based counterparts, but also make the CO an integral part of every telco's overall cloud strategy, enabling a rich collection of services, including access for residential, mobile, and enterprise customers.

Commodity Hardware

CORD runs on commodity servers and white-box switches, coupled with disaggregated packaging of media access technologies. These hardware elements are then organized into a rack-able unit, called a POD, that is suitable for deploy-

Larry Peterson, Ali Al-Shabibi,Scott Baker, Andy Bavier, Jonathan Hart, Guru Palukar, and William Snow are with Open Networking Laboratory; Saurav Das is with the Open Networking Foundation; Tom Anshutz is with AT&T.

0163-6804/16/$25.00 © 2016 IEEE

ment in a telco CO.

We have settled on a particular configuration for an initial reference implementation of CORD. It includes:

- **Servers**: Open Compute Project (OCP)-qualified QUANTA STRATOS-S210-X12RS-IU servers, each configured with 128 GB of RAM, 2 × 300 GB HDDs, and a 40GE dual-port NIC.
- **Switches**: OCP-qualified and OpenFlow-enabled Accton 6712 switches, each configured with 32 × 40GE ports, and doubling as both leaf and spine switches in the CORD fabric.
- **I/O Blades**: OCP-contributed *AT&T Open GPON — NFV OLT Line Card*. Celestica-manufactured Optical Line Termination (OLT) "pizza boxes" that include merchant silicon OLT MAC chips from Microsemi. This is a 1u-blade that includes 48 × 2.5 Gb/s gigabit passive optical network (GPON) interfaces and 6 × 40GE uplinks [2].

The servers, switches, and OLT blades are assembled into two virtual racks (single physical rack) as illustrated in Fig. 1. The servers and OLT blades are interconnected by a leaf-spine switching fabric, consisting of two spine switches and two leaf (top-of-rack) switches per virtual rack. Also shown in the figure, Leaf-2 is connected to an upstream router, and the OLT blades are connected via GPON to ONTs and home routers.

The servers run Ubuntu LTS 14.04 and include Open vSwitch (OvS). The switches run the open source Atrium software stack [3], which includes Open Network Linux, the Indigo OpenFlow Agent (OF 1.3), and Broadcom's OpenFlow Data Plane Abstraction (OF-DPA), layered on top of Broadcom merchant silicon.

Figure 1 shows just one of many possible hardware configurations. For example, the leaf switches have sufficient capacity to support up to 24 dual-port servers per rack, 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 switches and fits in a partial rack.

Figure 1 also shows two OLT blades, but any access technology can be incorporated into CORD. This includes other residential technologies (e.g., 10GPON, G.Fast, DOCSIS), as well as broadband base units (BBUs) connecting mobile networks and carrier Ethernet connecting enterprises.

Software Building Blocks

With respect to software, CORD's reference implementation exploits four open source projects, as depicted in Fig. 2.

OpenStack [4] is the cluster management suite that provides the core Internet as a service (IaaS) capability, and is responsible for creating and provisioning VMs and virtual networks (VNs).

Docker [5] provides a container-based means to deploy and interconnect services. It also plays a role in deploying CORD itself (e.g., the other management elements are instantiated in Docker containers).

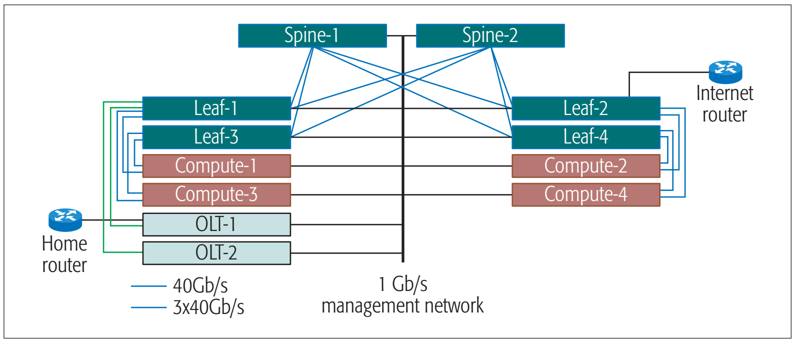

Figure 1. Target hardware POD built from commodity servers, I/O blades, and switches.

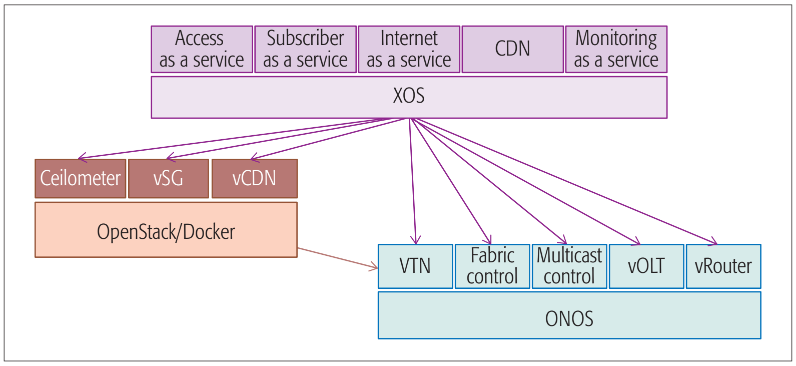

Figure 2. Open source software components in CORD. XOS assembles multi-tenant services, ONOS hosts control applications, and OpenStack/Docker manage compute instances. Specific services and control apps are described in later sections.

ONOS [6] is the network operating system that manages both software switches and the physical switching fabric. It hosts a collection of control applications that implement subscriber services and manage the switching fabric.

XOS [7] 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 VMs or containers).

To allow for the widest possible collection of services, the reference implementation supports services running in VMs, in containers running directly on bare metal, and in containers nested inside VMs.

ONOS plays two roles in CORD. It both interconnects VMs (this includes implementing VNs and managing flows across the switching fabric) and provides a platform for hosting control programs that implement 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 CO into CORD is a two-step process. The first step is to disaggregate and virtualize the devices, that is, turn each purpose-built hardware device into its software counterpart running on commodity hardware.

The second step is to provide a framework into which the resulting disaggregated elements can be plugged, producing a coherent end-to-end system. This framework defines the unifying abstractions that forge this collection of hard-

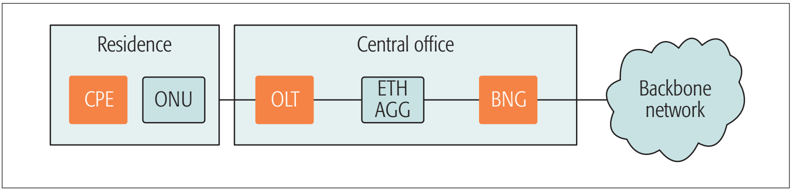

Figure 3. Legacy central office, including three devices (CPE, OLT, BNG) to be virtualized and disaggregated.

ware and software elements into a scalable and agile system. The following two sections describe these two steps to re-architect the CO in greater detail.

Disaggregating Legacy Devices

The first step involves disaggregating the existing hardware devices, transforming each legacy device into its commodity hardware plus software service counterpart. This section walks through the process for the legacy devices highlighted in Fig. 3, which includes optical line termination (OLT), customer premises equipment (CPE), and broadband network gateways (BNGs). We do not virtualize the Ethernet aggregation switch, per se, but the switching fabric in Fig. 1, under the control of ONOS, effectively replaces it.

For the sake of concreteness, this article focuses on the residential use case (i.e., GPON technology and the corresponding OLT device). However, the approach is equally applicable to other access technologies, and CORD is a general service delivery platform that also supports mobile and enterprise customers.

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. Virtualizing OLT is especially challenging because, unlike network appliances that are actually implemented by software running on vendor-branded commodity servers, OLT is implemented primarily in hardware. CPE is 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 CO, making them difficult to evolve in an agile and cost-effective way.

Disaggregating and virtualizing each physical device results in three elements: (1) merchant silicon, including both commodity servers and white-box switches; (2) a control plane function, to which we refer as the SDN element; and (3) a data plane function, to which we refer as the NFV element. Both NFV and SDN elements 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 servers, while SDN elements run on commodity servers but also *control* commodity 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 a 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, as illustrated throughout this section.

Disaggregating/Virtualizing the OLT

OLT terminates the optical link in the CO, with each physical termination point aggregating a set of subscriber connections. The first challenge is to create an I/O blade with the PON OLT medium access control (MAC), and to this end, AT&T has worked with the Open Compute Project to develop an open specification for an Open GPON OLT in the form of a 1RU "pizza box." This box includes the merchant silicon GPON MAC chips under control of a remote control program via OpenFlow.

These boxes 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 bridging). That is, vOLT implements authentication on a per-subscriber basis, establishes and manages VLANs connecting the subscriber's devices to the CO switching fabric, and manages other control plane functions of the OLT.

Disaggregating/Virtualizing the CPE

CPE, including both the GPON-terminating ONT and also a "home router" or "residential gateway," is installed in the customer's premises. They often run a collection of essential functions such as Dynamic Host Configuration Protocol (DHCP) and Network Address Translation (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 article focuses on the residential use case. By extending the capabilities of CPE in the cloud, new value-added 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 CO rather than at the customer's premises. There is still a device in the home (to which we still refer as the CPE), but it can be simplified, with some of the functionality that ran on the original CPE moved into the CO and running in a VM on commodity servers. In other words, the "customer LAN" includes a remote VM that resides in the CO, effectively providing every subscriber with 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 reference implementation uses a container-per-subscriber model, and gives subscribers the ability to select from a small collection of features (e.g., DHCP, NAT, firewall, uplink/downlink speeds, parental filtering). Each feature is then controlled by configuring the container bound to the subscriber, for example, using Linux mechanisms like iptables, dnsmasq, and tc. Experiments show this approach conservatively supports 1000 subscribers per server, with round-trip latency through the CORD POD well under 1 ms.

Disaggregating/Virtualizing the BNG

A BNG is one of the more complex and expensive devices in a CO, 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 quality of service (QoS), shaping and policing, virtual private networks (VPNs), generic routing encapsulation (GRE) tunneling, multiprotocol label switching (MPLS) tunneling, 802.1ad termination, and so on.

CORD's virtualized BNG, called a 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, that functionality is provided by another service (e.g., vOLT authenticates subscribers and vSG implements per-subscriber functionality).

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 viewed as a distributed router with line cards and backplanes instantiated by bare-metal switches. The vRouter control program then routes between the attached per-subscriber subnets. It also peers with legacy routers (e.g., advertising BGP routes).

Our approach to vRouter highlights an example of refactoring. Historically, BNG is responsible for authenticating the user as well as subscriber management, but those functions have been unbundled and moved to vOLT and vSG, respectively. This is because subscribers have to be authenticated *before* accessing vSG, which used to reside in the home but has now moved into the CO.

End-to-End Packet Flow

We conclude by sketching a subscriber's packet flow through the CORD, assuming the subscriber already has an account. When the subscriber powers up the home router, an 802.1x authentication packet is sent over GPON to the CO. Upon arrival at a GPON 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 [8]; more information on the latter is given in the next section.

Service Framework

This section focuses on the second step in re-architecting the CO as a data center: 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

Disaggregating hardware devices and replacing them with merchant silicon and software running in VMs is a necessary first step, but it is not sufficient. Just as all the devices in a hardware-based CO must be wired together, 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:

- The elastic scale-out of the resulting virtualized functionality
- The composition of the resulting disaggregated 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 [7]. 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), 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 CO are also managed as scalable services (as outlined below, CORD includes a content distribution network, CDN, as an illustrative example of a conventional cloud service).

Scalable Services, Not Virtual Devices

While the terms vOLT, vSG, and vRouter are used to refer to the virtualized counterparts of

Disaggregating hardware devices and replacing them with merchant silicon and software running in virtual machines is a necessary first step, but it is not sufficient. Just as all the devices in a hardware-based CO must be wired together, their software counterparts must also be managed as a collective.

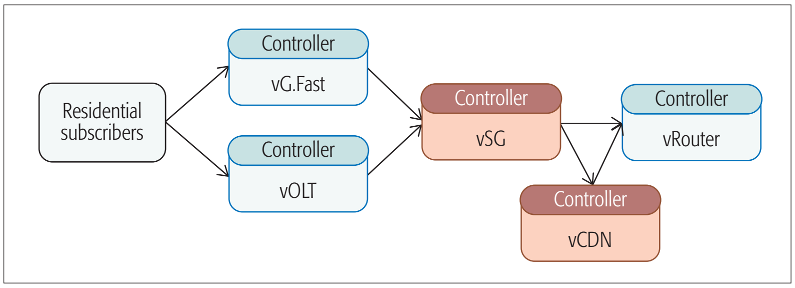

Figure 4. CORD service graph, including two access services: vOLT and vG.Fast.

the three physical devices, they are all 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 acquires a *subscriber VLAN*.
- vSG is a data plane function scaled across a set of containers. It implements *subscriber as a service*, where each tenant acquires a *subscriber bundle*.
- vRouter is a control program running on ONOS. It implements *Internet as a service*, where each tenant acquires a *routable subnet*.

Our reference implementation adds a CDN — itself a scalable cloud service deployed throughout the operator's network, including caches in the CO — to the mix. This gives us 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). Moreover, we have disaggregated the BNG: it is subsumed by a combination of vOLT, vSG, vRouter, and the underlying switching fabric.

This results in the legacy CO depicted in Fig. 3 being re-architected into the service graph shown in Fig. 4. To illustrate CORD's generality as a configurable platform, Fig. 4 also includes a *vG.Fast* service to represent a second 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 read the service graph in Fig. 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.*"

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 Fig. 4 is instantiated in CORD, the subscriber is able to control his or her subscription (e.g., set the parental control feature to disallow access to certain sites) 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.

The service graph shown in Fig. 4 is simplified to focus on the services that provide direct value to end users. There are also a collection of building block services on which the services in Fig. 4 depend, including ONOS (e.g., both vOLT and vRouter are tenants of ONOS) and a monitoring service (e.g., Ceilometer on Fig. 2) that collects and aggregates meters from each hardware and software element in CORD.

Layers of Abstraction

The service graph shown in Fig. 4 represents the high-level specification a network operator provides, a so-called *service control plane*, where this specification is mapped onto the underlying servers, switches, and I/O blades. This mapping is the consequence of a set of nested abstractions that CORD layers on top of the building block components shown in Fig. 2. Working top down, CORD defines the following abstractions.

Service Graph: Represents dependency relationships among a set of *services* (see next). CORD models service composition as a *tenancy* relationship between a provider service and a tenant service. Service tenancy is anchored in a *tenant principal* (e.g., subscriber).

Service: Represents an elastically scalable, multi-tenant program, including an interface to instantiate, control, and scale functionality. CORD models a service as a *service controller* that exports a multi-tenant interface and a slice (see next) that contains an elastically scalable set of *service instances*.

Slice: Represents a POD-wide resource container in which services execute, including a specification of how those resources are embedded in the underlying infrastructure. CORD models a slice as a set of *service instances* (VMs or containers) and a set of VNs. Instances are provided by the underlying IaaS components (OpenStack and Docker), while VNs are implemented by ONOS (see next).

Virtual Network: Represents a communication interconnect 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 support service composition.

A pair of ONOS-hosted control applications implements CORD VNs. The first, called *VTN*, installs flow rules in the OvS running on each server to implement a service composition overlay, using VxLAN tunnels and a custom OvS pipeline. The second, called *Fabric Control*, implements aggregate flows between servers across the layer 2/3 (L2/L3) leaf-spine switching fabric (i.e., the hardware underlay), and works with other ONOS applications to interface with downstream access devices and upstream metro routers.

Looking at Fig. 4 through the lens of NFV, each service in CORD corresponds to a virtualized network function (VNF) in the NFV architecture [9]. How a sequence of such VNFs (a service chain) maps onto a sequence of compute instances (VMs or containers) depends on three things:

- Whether the service is implemented on the network control plane or in the network data plane
- How each service maps its tenants onto one or more instances
- The type of VN interconnect instances

As a consequence, a linear chain of instances 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.

Security Architecture

The XOS abstractions effectively layer a service control plane on top of a collection of micro-services, and in doing so, provide explicit support for multiple domains of trust. This makes it possible to mediate trust (verify a chain of trust through a sequence of components) and apply the principle of least privilege (support a fine-grain separation of privilege):

- XOS minimizes the trusted code base by running as many management services as possible in isolated IaaS-provided slices rather than on cluster head nodes.
- XOS mediates trust by requiring all service-to-service and operator-to-service control operations to pass through a logically centralized XOS control point, where the security policy is enforced.
- XOS supports least privilege on the control plane by providing a role-based access control mechanism that associates fine-grained privileges with an extensible set of roles.
- XOS supports least privilege on the data plane by allowing services to control the network(s) through which their instances are accessed, rather than interconnect services using publicly routable Internet addresses.
- XOS makes it possible to verify an end-to-end chain of trust by modeling all services as multi-tenant, with tenants corresponding to either authenticated users or other services.

Concluding Remarks

CORD is a revolutionary effort to transform legacy central offices in the telco network. In the re-architected CO, 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 CO that enjoys both the CAPEX and OPEX benefits of commodity infrastructure, and the agility of modern cloud providers.

The reference implementation of CORD is both sufficiently complete to support field trials, and general enough to be applicable to many telco and cable applications, including specific implementations that address verticals in this large market. For example, in addition to the residential use case described in this article (R-CORD), there is a CORD implementation targeted at mobile users (M-CORD) and another targeted at enterprise users (E-CORD). Specifications and software for the reference implementations are available at http://opencord.org/.

The reference implementation of CORD is both sufficiently complete to support field trials, and general enough to be applicable to many telco and cable applications, including specific implementations that address verticals in this large market.

Acknowledgments

Many people have contributed to CORD, including Charles Chan, Flavio Castro, Matteo Scandolo, Masayoshi Kabayashi, Hyunsun Moon, Zack Williams, Don Newton, David Bainbridge, Zsolt Haraszti, Srikanth Vavilapalli, Marc Fiuczynski, and Sapan Bhatia.

References

[1] K. Prabhu, "Delivering a Software-Based Network Infrastructure," AT&T Labs, Oct. 2015.
[2] T. Anschutz and R. Clark, "AT&T Open GPON – NFV OLT Line Card Specification," Open Compute Project, http://www.opencompute.org/wiki/Telcos, Mar. 2016.
[3] "Atrium: A Complete SDN Distribution from ONF," https://github.com/onfsdn/atrium-docs/wiki, visited June 2016.
[4] "OpenStack: Open Source Cloud Computing Software," https://www.openstack.org/, visited June 2016.
[5] "Docker: Build, Ship, Run Any App, Anywhere," https://www.docker.com/, visited June 2016.
[6] P. Berde *et al.*, "ONOS: Towards an Open, Distributed SDN OS," *HotSDN 2014*, Aug. 2014.
[7] L. Peterson *et al.*, "XOS: An Extensible Cloud Operating System," *ACM BigSystems 2015*, June 2015.
[8] "CORD: Re-inventing Central Offices for Efficiency and Agility," http://opencord.org, visited June 2016.
[9] "Network Functions Virtualization – An Introductory White Paper," *SDN and OpenFlow World Congress*, Oct. 2012.

Biographies

LARRY PETERSON (llp@cs.princeton.edu) is chief architect at the Open Networking Lab and the Robert E. Kahn Professor of Computer Science, Emeritus, at Princeton University. He is a recipient of the IEEE Kobayashi Computer and Communication Award and ACM SIGCOMM Award, and a member of the National Academy of Engineering. He received his Ph.D. from Purdue University in 1985.

ALI AL-SHABIBI is a member of technical staff at the Open Networking Lab. Before joining ON.Lab, he was a postdoctoral researcher at Stanford University. He received his Ph.D. from the University of Heidelberg, Germany, in 2011 after performing his doctoral research at CERN.

TOM ANSCHUTZ is a Distinguished Member of Technical Staff at AT&T. He has been instrumental in the development of AT&T's SDN and NFV architecture, called Domain 2.0. He also has experience in product management and standards development, where he was named a Distinguished Fellow of the Broadband Forum. He has been granted 50 patents and earned an M.S.E.E. at Georgia Tech in 1986.

SCOTT BAKER is a member of technical staff at the Open Networking Lab. Previously, he was involved in the PlanetLab and GENI projects, where he contributed the Raven Provisioning Service. He received his Ph.D. from the University of Arizona in 2005.

ANDY BAVIER is a member of technical staff at the Open Networking Lab and an associate research scholar at Princeton University. He has been a long-time contributor to the PlanetLab and GENI projects, and was awarded the SIGCOMM Test-of-Time Award as an author of an early PlanetLab paper. He received his Ph.D. from Princeton University in 2004.

SAURAV DAS is a principal system architect at the Open Networking Foundation. He did pioneering research on SDN in wide area networks and demonstrated a converged IP/MPLS/optical WAN architecture based on SDN and OpenFlow. He also worked on controller scalability at Big Switch Networks and optical systems at Enablence. He holds an M.S. in optical sciences from the University of Arizona and a Ph.D. in electrical engineering from Stanford University.

JONATHAN HART is a member of technical staff at the Open Networking Lab. Originally from New Zealand, he completed his Bachelor's degree in network engineering from Victoria University of Wellington in 2011. He has been involved with SDN projects in both academia and industry over the past several years, and is a core developer on the ONOS project.

GURU PARULKAR is a consulting professor of electrical engineering at Stanford University and executive director of the Open Networking Lab. Previously, he co-founded and served as CTO of Growth Networks, Tenaya Networks, and Sceos, and was an early investor in Nicra. He was also a program director at the National Science Foundation, where he received NSF's Program Management Excellence award. He received his Ph.D. from the University of Delaware in 1987.

William Snow is VP for engineering at the Open Networking Lab. Previously, he served as VP of engineering and operations for enterprise security start-ups, including Cymtec Systems, Agari Data, the 41st Parameter, and Identity Engines. He also served as director of engineering at Cisco Systems, where he was responsible for all routing and high availability features of the CRS-1, and as VP of engineering at Nortel Networks. He received a B.S. in electrical engineering from Cornell University, and M.S. degrees in both electrical and computer engineering and engineering management from Stanford University.