TABLE OF CONTENTS

1. Introduction 3
2. The Need for SDN 3
3. Overview of SDN 4
4. SDN Trends and Challenges 7
5. SDN Use Cases 9
   5.1 Use Case 1: Data Center 9
   5.2 Use Case 2: Wireless Mobile Edge - Seamless Roaming between 3g/4g + WiFi 10
   5.3 Use case 3: Carrier Network 11
   5.4 Use case 4: Network Application Management 11
6. AMD Smart SDN Solution 11
   6.1 Software Enablement for SDN 11
   6.2 High Compute Power at a Low Power Envelope 11
   6.3 Security Enhancement 12
       6.3.1 DPI - Understanding of Traffic Flow 12
   6.4 I/O Integration 12
   6.5 Other - Software 13
7. AMD SDN Solution using the APU 13
   7.1 AMD APU Details 13
8. Consequences of SDN 15
9. Conclusion 16
Abstract

Legacy heterogeneous networks have become very complex and hard to manage due to upgrade and interoperability challenges, evolving protocols, and management techniques traditionally hard-coded into the underlying hardware platform. SDN addresses these problems by decoupling virtual resources from the physical resources, abstracting control planes and data forwarding planes, and automating network management while enabling centralized orchestration. This paper provides background information on SDN, addresses some of the key technology considerations and requirements associated with it, and provides an overview of AMD’s approach to deploying smart SDN solutions that streamline SDN adoption and enable new levels of intelligence in next-generation networking platforms.

Introduction

The networking and communication industry is at a critical inflection point as it looks to embrace new technologies such as Software Defined Networking (SDN) and Network Functional Virtualization (NFV). While there are incredible advantages to deploying an SDN network, there are challenges as well: SDN and NFV require a revamping of network components and structures and new approaches to writing software for network management and implementing that code in the hardware.

A remarkable point to note is that SDN is not a new concept. Centralized network control and visibility have been around for a number of years, but what’s been missing until recently is a holistic view of networks and technology, with standardized separation of the control and data planes. SDN provides this capability and can efficiently enable data center and service providers to manage network configuration, management, routing, and policy enforcement for their evolving multitenant heterogeneous network.

This paper provides background information on SDN, addresses some of the key technology considerations and requirements associated with it, and provides an overview of AMD’s approach to deploying smart SDN solutions that streamline SDN adoption and enable new levels of intelligence in next-generation networking platforms.

The Need for SDN

Traditional network platforms have both control plane and data plane functionalities in a single physical unit. In this traditional network, routing and switching decisions are made at each individual unit on a dynamic basis. SDN, however, depending on deployment models (described in a later section), moves the control plane to a centralized location and keeps only the data plane in the switches.

SDN aims to solve the following issues found in traditional networks:

- In traditional networks, forwarding decisions are based on predefined rules over which network operators have no control. Thus, all packets going to the same destination are routed along the same path and treated the same way. If there were traffic congestion at any given link along the path, all traffic would suffer from congestion even though an alternate path is available with less traffic. In addition, legacy networks use a spanning tree protocol that limits the use of multilink/bundling, which can increase bandwidth between nodes.

- From the network architecture perspective, the current multitier network architecture (i.e., multiple switches connecting switches for switches) requires many more ports than the actual number of servers or end nodes. When virtual switches (vSwitches) are deployed, it further adds another tier to the network. This multitier architecture increases the complexity in the network structure. SDN architecture can simplify the multitier network architecture by virtualizing layers in the network.
From a network scalability and expansion perspective, a cloud infrastructure-based network with multiple tenants using numerous applications requires logical isolation from each other. However, traditional VLAN technology is unable to provide enough network segments to facilitate this. The future network requires scalable LAN segmentation. SDN networks can provide scalable LAN segmentation to effectively manage cloud infrastructure environments.

Network operators need to have the agility to quickly and easily upgrade their network infrastructure, but they’re often impeded by challenges, including real-time debug processes, recovery time requirements, backward compatibility constraints, and more. SDN can enable much shorter development times and easier management of the network through the utilization of common, general purpose hardware, while providing a holistic way of managing and controlling the network. Consequently, network operators can achieve CAPEX and OPEX savings.

Overview of SDN
As defined by the Open Networking Foundation, SDN decouples the network control and forwarding functions, enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services.

Unlike server virtualization, which enables sharing of a single physical resource by many users or entities, virtualizing network resources enables a consolidation of different physical resources by overlaying another or multiple layers of networks on heterogeneous networks, resulting in a homogenous network. Figure 1 describes three requirements that commonly define SDN.

Before delving deeply into SDN technology details, let’s first review the architecture of a typical data center switch. A typical switch consists of line card modules and control card modules. The line modules are used for switching and forwarding and are typically built-in, purpose-built devices such as ASICs. Control modules, built on low-end control processors, handle network control and exception traffic. SDN moves network control from network elements/switches to a centralized network controller (or multiple controllers) using software running on general purpose hardware – this approach is designed to achieve increased control and flexibility. Figure 2 shows the base components of a traditional data center switch vs. an SDN switch and network.

---

**Figure 1. SDN Definition**

- **DECOUPLING**
  - Separation of virtual functions from the physical hardware, so that network functions can be remotely located

- **ABSTRACTION**
  - Abstraction of the control plane and data forwarding plane, which can allow separation of the control plane from the physical platform

- **AUTOMATION AND ORCHESTRATION**
  - Centralized or distributed centralized control is enabled
  - Efficient operation and automated management of networks
OpenFlow is one of the enabling technologies used in an SDN environment, defining the communication interface between a controller (control plane) and forwarding switches (data plane). Supported primarily by the Open Networking Foundation (ONF), OpenFlow removes the entire control plane from the network equipment.

On the other hand, Path Computation Element (PCE) is another SDN technology option and is mostly preferred by closed environments like data centers. It is standardized by the Internet Engineering Task Force (IETF), migrates only the path computation component of networking devices to a centralized role, and is mostly preferred by carrier providers.

The following is a summary of OpenFlow:

- It was initially deployed in the campus network to run experimental protocols and continues to be maintained as an open source standard.
- It provides a standard interface to program switches and routers without using any vendor-specific APIs.
- It allows network managers to access flow tables and update the rules used for switches and routers to direct network traffic based on the entire view of the network. It gives network managers the flexibility to have control over the switching/routing rules with priority control and ACL control, or utilize a custom/new protocol.
- It is independent from the underlying hardware technology, thus enabling SDN.

<table>
<thead>
<tr>
<th>Key Benefit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unifying multi-vendor environments</td>
<td>Most networks are composed of platforms from multiple vendors. Each vendor’s platform utilizes unique and mostly proprietary user interfaces and commands. SDN can put a unified interface to these platforms, allowing centralized management of multivendor environments.</td>
</tr>
<tr>
<td>Reducing complexity</td>
<td>SDN automates the process of updating and configuring multiple platforms in the network.</td>
</tr>
<tr>
<td>Simplifying network update</td>
<td>Using programmability provided by SDN, a new network protocol and policy management framework can be tested and deployed quickly.</td>
</tr>
<tr>
<td>Increasing network reliability and QoS management</td>
<td>SDN controllers provide complete visibility and control over the network – network failure can be detected and managed easily. In addition, it can give better visibility into the traffic and network, providing network-level QoS.</td>
</tr>
</tbody>
</table>

Table 1. SDN Benefits
Figure 3 shows an overall SDN network architecture and well-known solutions available for each layer. Here, all data plane switches are considered as individual switches, whether they are physical switches or virtual switches. A virtual switch is a software switch that can be loaded on an x86 server. As described in the previous sections, the control plane is separated from the data plane in an SDN environment. Situated between the data plane and control plane, the southbound interface includes the service abstraction layer (SAL) – the most popular southbound interface is OpenFlow. Between the control plane and the application layer, the northbound interface is defined. One of the more popular implementations of the northbound interface is the REST (Representational State Transfer) based interface. OpenDaylight is one of the SDN controllers available currently. An open source framework and platform that includes code and architecture, OpenDaylight can help accelerate the adoption of SDN and NFV. The controller spans southbound and northbound interfaces, including the control plane.

OpenStack is a commonly used open source application and orchestration layer, providing the ability to manage and control large numbers of compute, storage, and networking resources in data center networks. OpenStack has several major components:

- Compute: Provision and manage large networks of virtual machines (NOVA)
- Storage: Object and block storage for use with servers and applications (Swift, Cinder)
- Network: Defines pluggable, scalable, API-driven network and IP management (Neutron)
- DashBoard: Provides interface to access, provision, and automate cloud-based resources (Horizon)
- Identity Service: Configures centralized policies across users and systems (Keystone)
- Image Service: Enables administrators to create base templates and users to choose from available images or create their own from existing servers (Glance)

In summary, OpenStack is a cloud operating system (OS) that controls large pools of compute, storage, and networking resources throughout a data center.
SDN Trends and Challenges

There are several different SDN deployment scenarios in the industry, although the original SDN concept proposes to have a centralized control plane with only the data plane remaining in the network.

On the controller implementation, a few variations have been considered in the industry. A central or distributed architecture has one or more SDN controllers, and it controls all the switches or subset of switches in the network. Distributed architecture administers a cluster of switches via a dedicated controller, and there are many such clusters with several SDN controllers in the network. This distributed architecture can lower the risk of failure, whereas a single centralized SDN controller failure leads to entire network failure. Using a hierarchical controller is another approach – the Logical xBar\(^4\) may provide better network scalability.

Apart from the controller, the data plane can also become a challenge with the transition to SDN because traditional switching and/or forwarding devices/ASICs will not be able to easily support SDN traffic due to evolving standards. Hence the need to have a hybrid approach. Specifically, a portion of the network (e.g., access network) can be SDN enabled while the other portion (e.g., core network) can remain as a ‘traditional’ network. Thus traditional platforms are located in the intermediate nodes, acting as a big pipe, and SDN enabled platforms are actual switch and routing platforms. With this approach, an SDN network may be enabled immediately without having the need to overhaul the entire network.

For most of the switch platform vendors, SDN is a disruptive technology; hence they may prefer to keep their value added technologies – including part of the control plane – in their platform. This is a control plane hybrid architecture. In this model, the interface between SDN control plane and data plane would still follow the standard (e.g., OpenFlow API), but once the SDN control command reaches the switch platform, the switch would do additional value-added processing with the localized control plane. This localized control plane could potentially provide services to the local traffic without relying on the central controller’s support (may report the locally performed services to the central controller).

Another important challenge that needs to be addressed is the communication latency and load between the SDN controller and the switches. If the network has one centralized controller, then all connection setup requests as well as exception traffic get forwarded to the controller. This increases the load on the controller significantly, and the resulting latency caused from the communication may be unacceptable for latency-sensitive applications. In latency-sensitive environments, further study is needed. DevoFlow\(^6\) is an example of an approach that harnesses local intelligence to scale traffic flow, overcoming performance and latency challenges.

If the SDN network needs to provide higher levels of service including traffic monitoring and application recognition, traffic flows or packets selected for monitoring need to be sent to the controller, as the SDN switching platform is a simple forwarder. Thus, the amount of traffic that needs to be forwarded to the controller becomes significant. Furthermore, if a secure channel is required between switches and controllers as recommended in the OpenFlow standard (TLS secure channel), then the amount of traffic that needs to be encrypted and decrypted can be significant and can draw hardware resources from the switching platforms’ processors and the central controller. Therefore crypto acceleration functionality can be needed to enable SDN.
An SDN network may exhibit latency issues due to centralized control mechanisms. Delay caused by communication between SDN controller and switch could be a significant issue for delay-sensitive applications. As shown in Figure 4, end-to-end delay involved in SDN controller communication has two major components. One is a delay (Delay 2) caused by communication between SDN controller and OpenFlow switch, the other (Delay 1) is an internal delay between control processor and switch device. DevoFlow could be used to solve the Delay 2 issue. Delay 1 can be solved with high-speed, low-cost processors with high-speed interfaces.

The following table lists the differences between traditional networks and SDN.

<table>
<thead>
<tr>
<th>Traditional Network</th>
<th>SDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Configurable Networks</td>
<td>• Programmable Networks</td>
</tr>
<tr>
<td>• Apps-Aware Networks</td>
<td>• Network-Aware Apps</td>
</tr>
<tr>
<td>• Managed Networks</td>
<td>• Automated Networks</td>
</tr>
<tr>
<td>• Fixed Network Interfaces</td>
<td>• Open Networks</td>
</tr>
<tr>
<td>• Vendor Chassis/Appliances</td>
<td>• Generic Switches &amp; Servers</td>
</tr>
<tr>
<td>• Custom ASICs</td>
<td>• x86, general purpose CPUs</td>
</tr>
</tbody>
</table>

Table 2: Changes from Traditional Networks to SDN
As networks move more into an “open” architecture, security becomes a critical requirement. Providing security in the centralized control is imperative to avoid a single point of failure.

Challenges in SDN are still emerging as the definition of SDN continues to evolve. The scale-out network paradigm is evolving as well. Due to these uncertainties, abstraction mechanisms from different vendors will compete or co-exist. In addition, creation of SDN controllers and switches requires resolution of design challenges in many hardware platforms.

SDN Use Cases

5.1 Use Case 1: Data Center
The data center environment is the most common use case for SDN. In the traditional data center network, there are ToR (Top of Rack), EoR (End of Row), aggregation, and core switches. Multitier networking is a common configuration. To increase data center network manageability, SDN can abstract physical elements and represent them as logical elements using software. It treats all network elements as one large resource across multiple network segments. Therefore it can provide complete visibility of the network and manage policies across network nodes connected to virtual and physical switches.

Figure 5 shows a traditional multitier data center network, and how an SDN controller can manage the entire network from a centralized location.
5.2 Use Case 2: Wireless Mobile Edge - Seamless Roaming between 3G/4G + WiFi

A wireless network is another good use case for deployment of SDN. The growing explosion of handheld devices such as smartphones and 3G/4G-enabled tablets has increased bandwidth consumption per device in hyper-scale, causing spectrum availability and coverage issues in many areas. As mobile spectrum is expensive and limited, one proposed solution to these congestion issues is to use an SDN solution that leverages unlicensed spectrum via Wi-Fi to both offload spectrum and increase spectrum density. This solution is very effective for special events where many people gather in small areas (such as stadiums). The SDN solution also addresses network management challenges, supporting fast, seamless voice, data, and video transition from 3G/4G network to Wi-Fi networks. In this case, as shown in Figure 6, the SDN solution dynamically partitions access points and cell radios based on carriers, usage, identity, and device type to enable optimal usage of spectrum, Wi-Fi and mobile backhaul links to ensure that the maximum number of users can access network resources. The SDN controller also provides a holistic view of the network and dynamically allocates resources based on the status of the network.

In wireless networks, SDN can also be used for separating the control plane from the traditional gateways where there is a control path and forwarding path combined. A centralized SDN controller runs the control plane and manages the gateway platform’s data plane, resulting in a simpler gateway platform architecture. This approach enables dynamic control plane updates and scalability. The ONF Wireless and Mobile Working Group is currently studying this approach.
5.3 Use Case 3: Carrier Network
Carrier networks are mission critical. They are the backbone of the communications infrastructure and are expected to be reliable and fully functional all the time. The costs of building and maintaining these critical networks are tremendous, and it can be extremely difficult and time-intensive to modify them. SDN for carrier networks is especially complex. However, there are possible candidates for SDN use including:

1. Hotspot 2.0/LTE/Wi-Fi/3G converged access: This is the same use as in the wireless mobile edge network use case above, but it would be offered as a carrier network service.

2. Unique QoE per subscriber class: As SDN promises flow-level management, it can enhance user level service quality. With a full view of the network, the controller can provide the best routing path for the user traffic. Therefore service providers can create premium services and can further enable network monetization.

5.4 Use Case 4: Network Application Management
SDN can enable easy platform upgrades and migration for applications such as firewalls. In this example, an SDN controller can obtain firewall rule information from existing firewall appliances (through CLI) and extract firewall ruleset and route information. Then the SDN controller transfers firewall rulesets to a newer firewall platform, while the SDN controller programs an OpenFlow switch connected to both old and new firewall appliances to redirect the traffic from the old to the new firewall appliance. Route information extracted from the old firewall appliance can be used to program the OpenFlow switch, as the route information would indicate which flows are going through the firewall appliance. Once this process gets established and processes are standardized, full automation can be achieved.

AMD Smart SDN Solution
SDN’s basic tenet is to remove vendor specific dependencies, to reduce complexity, and to improve control, allowing the network to quickly adapt to changes in business needs. Other key SDN requirements, as outlined before, are the disaggregation of control and data planes, and strong compute and packet processing capabilities. AMD has developed the AMD Smart SDN solution to meet SDN’s requirements as described in the following sections.

6.1 Software Enablement for SDN
AMD provides an integration of various components needed to enable SDN, such as ODP, DPDK and OpenStack. This middleware, e.g., DPDK or ODP, enables fast packet I/O for general-purpose CPU platforms, which tend to have a bottleneck in the data path if there is no user space pass-through enablement. This middleware software is a must-have requirement to enable an SDN solution, providing a unified interface to various platforms including AMD x86 and ARM64 platforms.

6.2 High Compute Power at a Low Envelope
An SDN controller has to have strong compute capability to handle large amounts of control traffic coming from many SDN switches – each individual flow needs handling by the central SDN controller. This brings concerns regarding the SDN controller in terms of performance and single point of failure.

There are different architectures proposed in the industry to mitigate the load to the central controller. One example is a distributed centralized controller, which has several SDN controllers, each managing a subsection of the network, with an additional control layer managing these regional controllers. This architecture requires smart, distributed powerful compute capabilities throughout the entire network of SDN controllers.

AMD provides various platforms designed to meet different SDN needs and requirements, all with a common interface. As mentioned above, different nodes, including SDN switch nodes, require different levels of performance and power. AMD provides a wide range of low power to ultra-high performance devices as listed in Figure 7.
6.3 Security Enhancement

OpenFlow recommends TLS secure channels between the OpenFlow controller and OpenFlow switches/agents. However, this will become a mandatory requirement soon for most data center and enterprise applications, as there is a growing need for security. As the amount of control traffic increases, the needs of crypto acceleration or offload increase together. By offloading crypto operation to acceleration engines such as CCP (Crypto Co-processor) on a CPU or GPU in AMD APU (Accelerated Processing Unit) processors, the system level performance can be maintained without compromising compute performance. AMD provides embedded APUs (accelerated processing units) ranging from the high-end AMD Embedded R-Series platform to the low-power AMD Embedded G-Series platform.

6.3.1 DPI - Understanding of Traffic Flow

For an SDN controller to manage the network and associated policies, it requires a good understanding of networking traffic. Centralized or distributed SDN architectures can support a deep understanding of traffic by collecting sets of packets from a traffic flow, and analyzing them. There are two different ways to support this requirement.

- **Option 1**: Based on the assumption of having a big pipe/channel between SDN switches and SDN controller, all of the deep packet inspection or application recognition can be done in the central controller with a powerful DPI engine.

- **Option 2**: A small DPI engine can be implemented in the distributed SDN switches. These switches perform a basic deep packet inspection, then report the results or send only streams of important traffic. As we have seen, the latter case requires cheaper and simpler implementation to meet the basic SDN tenet. Low-cost and low-power AMD APU solutions can be used for DPI applications through the utilization of the APU’s onboard GPU, which is optimized for highly parallel programmable applications.

6.4 I/O Integration

The main processor for SDN requires high-speed I/O interfaces, i.e., embedded network interfaces such as 1G, 10GE, and PCIe. This can lower system cost and ease system design complexity.
6.5 Other - Software
As alluded to in the previous section, SDN-enabled systems will be based on commoditized switches and need to support only basic system level control functions. However, future SDN-enabled platforms may require a sub-set of intelligent control and data plane instructions as listed in Table 3. Meanwhile, Most of the intelligence resides in the central SDN controller.

Further complicating development of SDN solutions are the evolving standards. Throughout the industry, there are different approaches to enabling network virtualization (VXLAN, NVGRE, etc.), and these standards evolve as they move to the next phases. To meet the requirements of these evolving standards – and any emerging network overlaying protocols – platforms must be able to provide flexibility and ease of programmability. As an example, the transition from the OpenFlow 1.0 spec to the OpenFlow revision 1.3 significantly increased complexity, as it aimed to support many types of networking functions and protocols. AMD platforms can provide a scalable alternative to solve emerging complexities with SDN protocols such as OpenFlow and, in general, the dynamic nature of SDN overlays. AMD is working diligently with our ecosystem partners to enable these software features, all of which will be optimized for AMD platforms.

### AMD SDN Solution Using the APU

#### 7.1 AMD APU Details
AMD provides an integration of various components needed to enable SDN, such as ODP, DPDK and Open-Stack. This middleware, e.g., DPDK or ODP, enables fast packet I/O for general-purpose CPU platforms, which tend to have a bottleneck in the data path if there is no user space pass-through enablement. This middleware software is a must-have requirement to enable an SDN solution, providing a unified interface to various platforms including AMD x86 and ARM64 platforms.

- General purpose, programmable scalar (CPU), and vector processing cores (GPU)
- High-performance bus
- Common, low-latency memory model (HSA)

### Table 3. SDN Intelligent Feature Requirements

<table>
<thead>
<tr>
<th>Key Benefit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Plane</td>
<td>Support for the continuation of traditional control plane which does not require a central controller – Hybrid approach</td>
</tr>
<tr>
<td>Control/Service Plane</td>
<td>Value-added features such as • Ethernet OAM/BFD, • Security – Firewall and SSL/TLS, • DPI • Service and application awareness • QoS management</td>
</tr>
<tr>
<td></td>
<td>Dynamic policy update/programmability</td>
</tr>
</tbody>
</table>
HSA (Heterogeneous System Architecture) is implemented in some AMD APU platforms, and is critical to maximizing throughput. With HSA, the CPU hands over only the pointers of the data blocks to the GPU. The GPU takes the pointers and processes the data block in the specific memory location and hands them back to the CPU. HSA ensures cache coherency between the CPU and the GPU. Figure 8 depicts an overview of the APU architecture.

![APU High Level Architecture](image)

**Figure 8. APU High Level Architecture**

GPUs are extremely efficient and effective for parallel processing applications, and they can also be used for crypto operations, DPI (deep packet inspection), classification, compression and other applications. In the case of crypto operations, the CPU doesn’t have to get involved in the data plane crypto operation directly. With this architecture, the system level performance can be maintained even when the amount of traffic needing encryption or decryption increases. This is one of the key features of the AMD Smart SDN solution, which can selectively accelerate or offload CPU compute-intensive operations to the GPU. Here are a few additional functions that can be accelerated or offloaded to the GPU:

- DPI: Implement PCRE based RegEx engine
- Security (such as IPSec) operations: RSA, crypto operation
- Compression operation for distributed storage applications
Figure 9 shows AMD’s SDN use case details. Multi-core AMD Embedded R-Series and G-Series processing platforms can be deployed in central SDN controllers where high single thread compute power is used, as well as SDN switch controllers which may need a subset of intelligence to lower the central controller’s load and localize network traffic.

**Consequences of SDN**

SDN introduces a new approach to network resource utilization and management, and each networking vendor in the market is looking into their own way to build SDN solutions. One key action that needs to be taken to enable SDN is to open up the intelligence of switches and routers to enable the abstraction of proprietary vendor technologies. Many equipment vendors that have been traditionally reluctant to open up their proprietary technologies have slowly opened up their APIs in response to the growing adoption of SDN. Mega data center players (Amazon, Google, Facebook and the like) are implementing technologies that will allow them to enable greater flexibility and lower costs. Amazon and Google are building their own networking (white box) switches so that they don’t have to rely on the platforms produced by OEM vendors. Facebook is driving the Open Compute Platform (OCP) to develop specifications for open architecture switches that will be manufactured by low-cost original device manufacturers. The open architecture approach from Facebook is creating an ecosystem where standard, high volume commodity platforms could be used to minimize CAPEX and OPEX costs.

Clearly the white box and open architecture approaches are creating major concerns for networking equipment manufacturers who prefer to add their own value-added technologies into hardware and software. SDN disrupts these existing technology and business models, but is creating more openness and enabling new and different business models.

SDN will drive toward a more software-centric architecture and implementation. Thus, it becomes difficult to provide platform differentiators. With SDN, the need for less expensive and easy-to-access hardware becomes paramount, and platform-specific, value-added services are deprioritized.
Conclusion

SDN aims to minimize the complexity of heterogeneous networks, protocols and management techniques, thereby achieving CAPEX and OPEX savings. This paper addresses the definition of SDN, trends and challenges associated with SDN, requirements, potential SDN use cases, and how AMD focuses on deploying smart SDN solutions that facilitate the seamless adoption of SDN while enabling greater intelligence in next generation networking platforms.

AMD provides differentiated technologies and enablement solutions to meet the requirements of SDN, including:

- A wide selection of low power and high performance processing platforms
- Integration of the middleware - DPDK or ODP. This middleware is required to enable fast packet I/O for general purpose CPU platforms.

This software enablement can provide a unified interface to various platforms including AMD x86 and ARM64 platforms

- GPU-enabled security enhancements. SSL/TLS is an optional feature for the communication between controllers and switches. In an APU, the on-board GPU can be used for crypto acceleration and offload
- Other GPU acceleration and offload features including DPI, compression, etc.
- Integrated I/O and various interfaces
- Open source integration, including OpenDaylight and OpenStack integration.

AMD’s Smart APU solution provides an intelligent processing platform capable of offloading and accelerating SDN protocols and functions making it ideally suited to handle the performance requirements of next generation networks.

www.amd.com/embedded

Keywords – Software Defined Network, OpenFlow, Network virtualization, Centralized Control, Acceleration, GPU compute.

DISCLAIMER

4) McCauley, James, et al. “Extending SDN to Large-Scale Networks.” ONS 2013
7) Palmer Wiretap Ventures, “WAN & Carrier Network,” ONS Summit 2012, Santa Clara, CA
8) As of December 2014
10) For details, please contact your AMD representative.
11) Jack Clark, “Amazon, Facebook, Google give Cisco’s switches the COLD shoulder,” The Register, http://www.theregister.co.uk/2013/11/18/cisco_cloud_problem/?page=1, 18th November 2013

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of non-infringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD’s products are as set forth in a signed agreement between the parties or in AMD’s Standard Terms and Conditions of Sale.

AMD, the AMD logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

© 2015 Advanced Micro Devices, Inc. All rights reserved. PID# 156640-A