WIKIPEDIA

Software-defined networking

Software-defined networking (**SDN**) technology is an approach to network management that enables dynamic, programmatically efficient network configuration in order to improve network performance and monitoring making it more like cloud computing than traditional network management.[1] SDN is meant to address the fact that the static architecture of traditional networks is decentralized and complex while current networks require more flexibility and easy troubleshooting. SDN attempts to centralize network intelligence in one network component by disassociating the forwarding process of network packets (data plane) from the routing process (control plane). The control plane consists of one or more controllers which are considered as the brain of SDN network where the whole intelligence is incorporated. However, the intelligent centralization has its own drawbacks when it comes to security,[1] scalability and elasticity[1] and this is the main issue of SDN.

SDN was commonly associated with the OpenFlow protocol (for remote communication with network plane elements for the purpose of determining the path of network packets across network switches) since the latter's emergence in 2011. However, since 2012[2][3] OpenFlow for many companies is no longer an exclusive solution, they added proprietary techniques. These include Cisco Systems' Open Network Environment and Nicira's network virtualization platform.

SD-WAN applies similar technology to a wide area network (WAN).[4]

SDN technology is currently available for industrial control applications that require extremely fast failover. One company boasts 100x Faster Failover for Mission-critical processes (fails over in less than 100 µs, compared to 10 ms for traditional networks) along with elimination of certain Cyber Vulnerabilities that are associated with traditional network management switches. [https://selinc.com/solutions/p/software-defined-network/]

Research of SDN continues as many emulators are being developed for research purposes, like vSDNEmul,[5] EstiNet,[6] Mininet[7] etc.

Contents

History

The history of SDN principles can be traced back to the separation of the control and data plane first used in the public switched telephone network as a way to simplify provisioning and management well before this architecture began to be used in data networks.

The Internet Engineering Task Force (IETF) began considering various ways to decouple the control and forwarding functions in a proposed interface standard published in 2004 appropriately named "Forwarding and Control Element Separation" (ForCES).[8] The ForCES Working Group also proposed a companion SoftRouter Architecture.[9] Additional early standards from the IETF that pursued separating control from data include the Linux Netlink as an IP Services Protocol[10] and A Path Computation Element (PCE)-Based Architecture.[11]

These early attempts failed to gain traction for two reasons. One is that many in the Internet community viewed separating control from data to be risky, especially owing to the potential for a failure in the control plane. The second is that vendors were concerned that creating standard application programming interfaces (APIs) between the control and data planes would result in increased competition.

The use of open source software in split control/data plane architectures traces its roots to the Ethane project at Stanford's computer sciences department. Ethane's simple switch design led to the creation of OpenFlow.[12] An API for OpenFlow was first created in 2008.[13] That same year witnessed the creation of NOX—an operating system for networks.[14]

Work on OpenFlow continued at Stanford, including with the creation of testbeds to evaluate use of the protocol in a single campus network, as well as across the WAN as a backbone for connecting multiple campuses.[15] In academic settings there were a few research and production networks based on OpenFlow switches from NEC and Hewlett-Packard; as well as based on Quanta Computer whiteboxes, starting from about 2009.[16]

Beyond academia, the first deployments were by Nicira in 2010 to control OVS from Onix, co-developed with NTT and Google. A notable deployment was Google's B4 deployment in 2012.[17][18] Later Google acknowledged their first OpenFlow with Onix deployments in their Datacenters at the same time.[19] Another known large deployment is at China Mobile.[20]

The Open Networking Foundation was founded in 2011 to promote SDN and OpenFlow.

At the 2014 Interop and Tech Field Day, software-defined networking was demonstrated by Avaya using shortest path bridging (IEEE 802.1aq) and OpenStack as an automated campus, extending automation from the data center to the end device, removing manual provisioning from service delivery.[21][22]

Concept

SDN architectures decouple network control and forwarding functions, enabling network control to become directly programmable and the underlying infrastructure to be abstracted from applications and network services.[23]

The OpenFlow protocol can be used in SDN technologies. The SDN architecture is:

- *Directly programmable*: Network control is directly programmable because it is decoupled from forwarding functions.
- *Agile*: Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.
- *Centrally managed*: Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.
- *Programmatically configured*: SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software.
- *Open standards-based and vendor-neutral*: When implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.

The need for a new network architecture

The explosion of mobile devices and content, server virtualization, and the advent of cloud services are among the trends driving the networking industry to re-examine traditional network architectures.[24] Many conventional networks are hierarchical, built with tiers of Ethernet switches arranged in a tree structure. This design made sense when client-server computing was dominant, but such a static architecture is ill-suited to the dynamic computing and storage needs of today's enterprise data centers, campuses, and carrier environments.[25] Some of the key computing trends driving the need for a new network paradigm include:

Changing traffic patterns

Within the enterprise data center, traffic patterns have changed significantly. In contrast to client-server applications where the bulk of the communication occurs between one client and one server, today's applications access different databases and servers, creating a flurry of "east-west" machine-to-machine traffic before returning data to the end user device in the classic "north-south" traffic pattern. At the same time, users are changing network traffic patterns as they push for access to corporate content and applications from any type of device (including their own), connecting from anywhere, at any time. Finally, many enterprise data centers managers are contemplating a utility computing model, which might include a private cloud, public cloud, or some mix of both, resulting in additional traffic across the wide area network.

The "consumerization of IT"

Users are increasingly employing mobile personal devices such as smartphones, tablets, and notebooks to access the corporate network. IT is under pressure to accommodate these personal devices in a fine-grained manner while protecting corporate data and intellectual property and meeting compliance mandates.

The rise of cloud services

Enterprises have enthusiastically embraced both public and private cloud services, resulting in unprecedented growth of these services. Enterprise business units now want the agility to access applications, infrastructure, and other IT resources on demand and à la carte. To add to the complexity, IT's planning for cloud services must be done in an environment of increased security, compliance, and auditing requirements, along with business reorganizations, consolidations, and mergers that can change assumptions overnight.

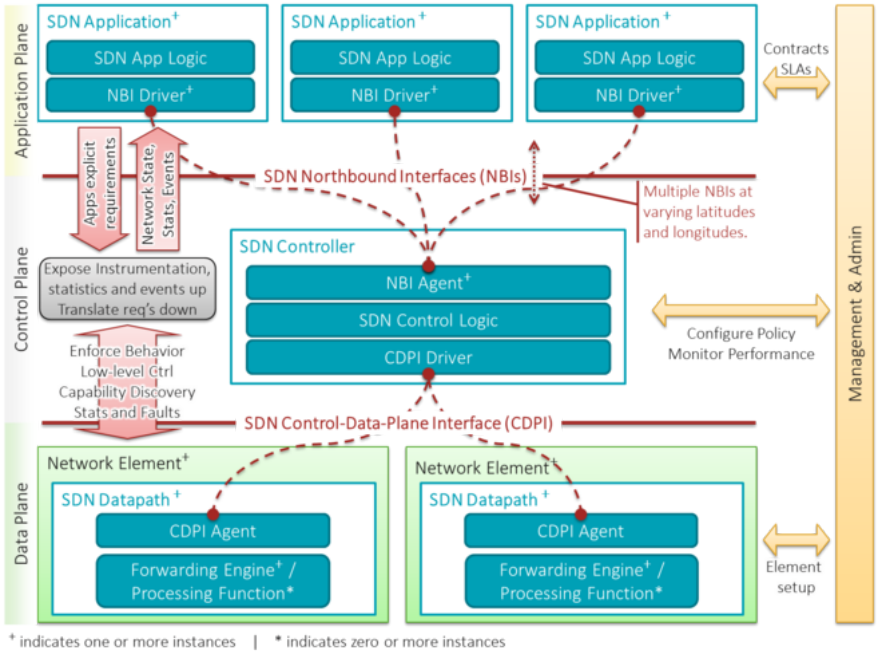
Providing self-service provisioning, whether in a private or public cloud, requires elastic scaling of computing, storage, and network resources, ideally from a common viewpoint and with a common suite of tools.

"Big data" means more bandwidth

Handling today's "big data" or mega datasets requires massive parallel processing on thousands of servers, all of which need direct connections to each other. The rise of mega datasets is fueling a constant demand for additional network capacity in the data center. Operators of hyperscale data center networks face the daunting task of scaling the network to previously unimaginable size, maintaining any-to-any connectivity without going broke.[26]

Architectural components

The following list defines and explains the architectural components:[27]



A high-level overview of the software-defined networking architecture

SDN Application

SDN Applications are programs that explicitly, directly, and programmatically communicate their network requirements and desired network behavior to the SDN Controller via a northbound interface (NBI). In addition they may consume an abstracted view of the network for their internal decision-making purposes. An SDN Application consists of one SDN Application Logic and one or more NBI Drivers. SDN Applications may themselves expose another layer of abstracted network control, thus offering one or more higher-level NBIs through respective NBI agents.

SDN Controller

The SDN Controller is a logically centralized entity in charge of (i) translating the requirements from the SDN Application layer down to the SDN Datapaths and (ii) providing the SDN Applications with an abstract view of the network (which may include statistics and events). An SDN Controller consists of one or more NBI Agents, the SDN Control Logic, and the Control to Data-Plane Interface (CDPI) driver. Definition as a logically centralized entity neither prescribes nor precludes implementation details such as the federation of multiple controllers, the hierarchical connection of controllers, communication interfaces between controllers, nor virtualization or slicing of network resources.

SDN Datapath

The SDN Datapath is a logical network device that exposes visibility and uncontested control over its advertised forwarding and data processing capabilities. The logical representation may encompass all or a subset of the physical substrate resources. An SDN Datapath comprises a CDPI agent and a set of one or more traffic forwarding engines and zero or more traffic processing functions. These engines and functions may include simple forwarding between the datapath's external interfaces or internal traffic processing or termination functions. One or more SDN Datapaths may be contained in a single (physical)

network element—an integrated physical combination of communications resources, managed as a unit. An SDN Datapath may also be defined across multiple physical network elements. This logical definition neither prescribes nor precludes implementation details such as the logical to physical mapping, management of shared physical resources, virtualization or slicing of the SDN Datapath, interoperability with non-SDN networking, nor the data processing functionality, which can include OSI layer 4-7 functions.

SDN Control to Data-Plane Interface (CDPI)
The SDN CDPI is the interface defined between an SDN Controller and an SDN Datapath, which provides at least (i) programmatic control of all forwarding operations, (ii) capabilities advertisement, (iii) statistics reporting, and (iv) event notification. One value of SDN lies in the expectation that the CDPI is implemented in an open, vendor-neutral and interoperable way.

SDN Northbound Interfaces (NBI)
SDN NBIs are interfaces between SDN Applications and SDN Controllers and typically provide abstract network views and enable direct expression of network behavior and requirements. This may occur at any level of abstraction (latitude) and across different sets of functionality (longitude). One value of SDN lies in the expectation that these interfaces are implemented in an open, vendor-neutral and interoperable way.

SDN Control Plane

Centralized - Hierarchical - Distributed

The implementation of the SDN control plane can follow a centralized, hierarchical, or decentralized design. Initial SDN control plane proposals focused on a centralized solution, where a single control entity has a global view of the network. While this simplifies the implementation of the control logic, it has scalability limitations as the size and dynamics of the network increase. To overcome these limitations, several approaches have been proposed in the literature that fall into two categories, hierarchical and fully distributed approaches. In hierarchical solutions,[28][29] distributed controllers operate on a partitioned network view, while decisions that require network-wide knowledge are taken by a logically centralized root controller. In distributed approaches,[30][31] controllers operate on their local view or they may exchange synchronization messages to enhance their knowledge. Distributed solutions are more suitable for supporting adaptive SDN applications.

Controller Placement

A key issue when designing a distributed SDN control plane is to decide on the number and placement of control entities. An important parameter to consider while doing so is the propagation delay between the controllers and the network devices,[32] especially in the context of large networks. Other objectives that have been considered involve control path reliability,[33] fault tolerance,[34] and application requirements.[35]

SDN flow forwarding (sdn)

Proactive vs Reactive vs Hybrid[36][37]
OpenFlow uses TCAM tables to route packet sequences (flows). If flows arrive at a switch, a flow table lookup is performed. Depending on the flow table implementation this is done in a software flow table if a vSwitch is used or in an ASIC if it's implemented in hardware. In the case when no matching flow is found, a request to the controller for further instructions is sent. This is handled in one of three different modes. In reactive mode the controller acts after these requests and creates and installs a rule in the flow table for the corresponding

packet if necessary. In proactive mode the controller populates flow table entries for all possible traffic matches possible for this switch in advance. This mode can be compared with typical routing table entries today, where all static entries are installed ahead of time. Following this no request is sent to the controller since all incoming flows will find a matching entry. A major advantage in proactive mode is that all packets are forwarded in line rate (considering all flow table entries in TCAM) and no delay is added. The third mode, hybrid mode, follows the flexibility of a reactive mode for a set of traffic and the low-latency forwarding (proactive mode) for the rest of the traffic.

Applications

SDMN

Software-defined mobile networking (SDMN)[38][39] is an approach to the design of mobile networks where all protocol-specific features are implemented in software, maximizing the use of generic and commodity hardware and software in both the core network and radio access network.[40] It is proposed as an extension of SDN paradigm to incorporate mobile network specific functionalities.[41] Since 3GPP Rel.14, a Control User Plane Separation was introduced in the Mobile Core Network architectures with the PFCP protocol.

SD-WAN

An SD-WAN is a Wide Area Network (WAN) managed using the principles of software-defined networking.[42] The main driver of SD-WAN is to lower WAN costs using more affordable and commercially available leased lines, as an alternative or partial replacement of more expensive MPLS lines. Control and management is administered separately from the hardware with central controllers allowing for easier configuration and administration.[43]

SD-LAN

An SD-LAN is a Local area network (LAN) built around the principles of software-defined networking, though there are key differences in topology, network security, application visibility and control, management and quality of service.[44] SD-LAN decouples control management, and data planes to enable a policy driven architecture for wired and wireless LANs. SD-LANs are characterized by their use of a cloud management system and wireless connectivity without the presence of a physical controller.[45]

Security using the SDN paradigm

SDN architecture may enable, facilitate or enhance network-related security applications due to the controller's central view of the network, and its capacity to reprogram the data plane at any time. While security of SDN architecture itself remains an open question that has already been studied a couple of times in the research community,[46][47][48][49] the following paragraphs only focus on the security applications made possible or revisited using SDN.

Several research works on SDN have already investigated security applications built upon the SDN controller, with different aims in mind. Distributed Denial of Service (DDoS) detection and mitigation,[50][51] as well as botnet[52] and worm propagation,[53] are some concrete use-cases of such applications: basically, the idea consists in periodically collecting network statistics from the forwarding plane of the network in a standardized manner (e.g. using Openflow), and then apply

classification algorithms on those statistics in order to detect any network anomalies. If an anomaly is detected, the application instructs the controller how to reprogram the data plane in order to mitigate it.

Another kind of security application leverages the SDN controller by implementing some moving target defense (MTD) algorithms. MTD algorithms are typically used to make any attack on a given system or network more difficult than usual by periodically hiding or changing key properties of that system or network. In traditional networks, implementing MTD algorithms is not a trivial task since it is difficult to build a central authority able of determining - for each part of the system to be protected - which key properties are hid or changed. In an SDN network, such tasks become more straightforward thanks to the centrality of the controller. One application can for example periodically assign virtual IPs to hosts within the network, and the mapping virtual IP/real IP is then performed by the controller.[54] Another application can simulate some fake opened/closed/filtered ports on random hosts in the network in order to add significant noise during reconnaissance phase (e.g. scanning) performed by an attacker.[55]

Additional value regarding security in SDN enabled networks can also be gained using FlowVisor[56] and FlowChecker[57] respectively. The former tries to use a single hardware forwarding plane sharing multiple separated logical networks. Following this approach the same hardware resources can be used for production and development purposes as well as separating monitoring, configuration and internet traffic, where each scenario can have its own logical topology which is called slice. In conjunction with this approach FlowChecker[56] realizes the validation of new OpenFlow rules that are deployed by users using their own slice.

SDN controller applications are mostly deployed in large-scale scenarios, which requires comprehensive checks of possible programming errors. A system to do this called NICE was described in 2012.[58] Introducing an overarching security architecture requires a comprehensive and protracted approach to SDN. Since it was introduced, designers are looking at possible ways to secure SDN that do not compromise scalability. One architecture called SN-SECA (SDN+NFV) Security Architecture.[59]

Group Data Delivery Using SDN

Distributed applications that run across datacenters usually replicate data for the purpose of synchronization, fault resiliency, load balancing and getting data closer to users (which reduces latency to users and increases their perceived throughput). Also, many applications, such as Hadoop, replicate data within a datacenter across multiple racks to increase fault tolerance and make data recovery easier. All of these operations require data delivery from one machine or datacenter to multiple machines or datacenters. The process of reliably delivering data from one machine to multiple machines is referred to as Reliable Group Data Delivery (RGDD).

SDN switches can be used for RGDD via installation of rules that allow forwarding to multiple outgoing ports. For example, OpenFlow provides support for Group Tables since version 1.1[60] which makes this possible. Using SDN, a central controller can carefully and intelligently setup forwarding trees for RGDD. Such trees can be built while paying attention to network congestion/load status to improve performance. For example, MCTCP[61] is a scheme for delivery to many nodes inside datacenters that relies on regular and structured topologies of datacenter networks while DCCast[62] and QuickCast[63] are approaches for fast and efficient data and content replication across datacenters over private WANs.

Relationship to NFV

NFV Network Function Virtualization is a concept that complements SDN. Thus, NFV is not dependent on SDN or SDN concepts. NFV disunites software from hardware to enable flexible network deployment and dynamic operation. NFV deployments typically use commodity servers to run network services software versions that previously were hardware-based. These software-based services that run in an NFV environment are called Virtual Network Functions (VNF).[64] SDN-NFV hybrid program was provided for high efficiency, elastic and scalable capabilities NFV aimed at accelerating service innovation and provisioning using standard IT virtualization technologies.[64][65] SDN provides the agility of controlling the generic forwarding devices such as the routers and switches by using SDN controllers. On the other hand, NFV agility is provided for the network applications by using virtualized servers. It is entirely possible to implement a virtualized network function (VNF) as a standalone entity using existing networking and orchestration paradigms. However, there are inherent benefits in leveraging SDN concepts to implement and manage an NFV infrastructure, particularly when looking at the management and orchestration of VNFs, and that's why multivendor platforms are being defined that incorporate SDN and NFV in concerted ecosystems.[66]

Relationship to DPI

DPI Deep Packet Inspection provides network with application-awareness, while SDN provides applications with network-awareness.[67] Although SDN will radically change the generic network architectures, it should cope with working with traditional network architectures to offer high interoperability. The new SDN based network architecture should consider all the capabilities that are currently provided in separate devices or software other than the main forwarding devices (routers and switches) such as the DPI, security appliances [68]

See also

- Active networking
- Frenetic (programming language)
- IEEE 802.1aq
- Intel Data Plane Development Kit (DPDK)
- List of SDN controller software
- Network functions virtualization
- ONOS
- OpenDaylight Project
- SD-WAN
- Software-defined data center
- Software-defined mobile network
- Software-defined protection

References

1. Benzekki, Kamal; El Fergougui, Abdeslam; Elbelrhiti Elalaoui, Abdelbaki (2016). "Software-defined networking (SDN): A survey". *Security and Communication Networks*. **9** (18): 5803–5833. doi:10.1002/sec.1737 (https://doi.org/10.1002%2Fsec.1737).
2. "Software-defined networking is not OpenFlow, companies proclaim" (http://searchsdn.techtarget.com/news/2240158633/Software-defined-networking-is-not-OpenFlow-companies-proclaim). *searchsdn.techtarget.com*.

3. "InCNTRE's OpenFlow SDN testing lab works toward certified SDN product" (http://searchsdn.techtarget.com/guides/Guide-OpenFlow-SDN-not-the-only-show-in-town-for-vendors).
4. "Predicting SD-WAN Adoption" (http://blogs.gartner.com/andrew-lerner/2015/12/15/predicting-sd-wan-adoption/). gartner.com. 2015-12-15. Retrieved 2016-06-27.
5. Farias, Fernando N. N.; Junior, Antônio de O.; da Costa, Leonardo B.; Pinheiro, Billy A.; Abelém, Antônio J. G. (2019-08-28). "vSDNEmul: A Software-Defined Network Emulator Based on Container Virtualization". arXiv:1908.10980 (https://arxiv.org/abs/1908.10980) [cs.NI (https://arxiv.org/archive/cs.NI)].
6. Wang, S.; Chou, C.; Yang, C. (September 2013). "EstiNet openflow network simulator and emulator". *IEEE Communications Magazine*. **51** (9): 110–117. doi:10.1109/MCOM.2013.6588659 (https://doi.org/10.1109%2FMCOM.2013.6588659). ISSN 1558-1896 (https://www.worldcat.org/issn/1558-1896).
7. Oliveira, R. L. S. de; Schweitzer, C. M.; Shinoda, A. A.; Ligia Rodrigues Prete (June 2014). "Using Mininet for emulation and prototyping Software-Defined Networks" (https://ieeexplore.ieee.org/document/6860404). *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*: 1–6. doi:10.1109/ColComCon.2014.6860404 (https://doi.org/10.1109%2FColComCon.2014.6860404). ISBN 978-1-4799-4340-1.
8. L. Yang (Intel Corp.), R. Dantu (Univ. of North Texas), T. Anderson (Intel Corp.) & R. Gopal (Nokia.) (April 2004). "Forwarding and Control Element Separation (ForCES) Framework" (https://tools.ietf.org/html/rfc3746).
9. T. V. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo (Nov 2004). "The SoftRouter Architecture" (http://conferences.sigcomm.org/hotnets/2004/HotNets-III%20Proceedings/lakshman.pdf) (PDF).
10. J. Salim (Znyx Networks), H. Khosravi (Intel), A. Kleen (Suse), and A. Kuznetsov (INR/Swsoft) (July 2003). "Linux Netlink as an IP Services Protocol" (https://tools.ietf.org/html/rfc3549).
11. A. Farrel (Old Dog Consulting), J. Vasseur (Cisco Systems, Inc.), and J. Ash (AT&T) (August 2006). "A Path Computation Element (PCE)-Based Architecture" (https://tools.ietf.org/html/rfc4655).
12. Martìn Casado, Michael J. Freedman, Justin Pettit, Jianying Luo, and Nick McKeown (Stanford University) (August 2007). "Ethane: Taking Control of the Enterprise" (http://cs.brown.edu/courses/csci2950-u/s14/papers/Casado07Ethane.pdf) (PDF).
13. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. (April 2008). "OpenFlow: Enabling Innovation in Campus Networks" (http://ccr.sigcomm.org/online/files/p69-v38n2n-mckeown.pdf) (PDF).
14. N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. (July 2008). "NOX: Towards an Operating System for Networks" (https://benpfaff.org/papers/nox.pdf) (PDF).
15. "GENI. Campus OpenFlow topology" (http://groups.geni.net/geni/wiki/OpenFlow/CampusTopology). 2011.
16. Kuang-Ching “KC” Wang (Oct 3, 2011). "Software Defined Networking and OpenFlow for Universities: Motivation, Strategy, and Uses" (https://www.internet2.edu/presentations/fall11/20111003-wang-openflow.pdf) (PDF).
17. Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jonathan Zolla, Urs Hölzle, Stephen Stuart and Amin Vahdat (Google) (August 12–16, 2013). "B4: Experience with a Globally-Deployed Software Defined WAN" (http://cseweb.ucsd.edu/~vahdat/papers/b4-sigcomm13.pdf) (PDF).
18. brent salisbury (May 14, 2013). "Inside Google's Software-Defined Network" (https://www.networkcomputing.com/networking/inside-googles-software-defined-network/512240144).
19. Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, Amin Vahdat (2015). "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network" (https://research.google.com/pubs/pub43837.html).

20. " "MPLS-TP OpenFlow Protocol Extensions for SPTN" becomes a formal ONF standard by unanimous approval" (https://www.telecomtv.com/content/tracker/mpls-tp-openflow-protocol-extensions-for-sptn-becomes-a-formal-onf-standard-by-unanimous-approval-27467/). June 27, 2017.
21. Camille Campbell (February 6, 2014). "Avaya Debuts Networking Innovations at 'Tech Field Day' " (https://www.avaya.com/blogs/archives/2014/02/avaya-debuts-networking-innovations-at-tech-field-day.html).
22. Elizabeth Miller Coyne (September 23, 2016). "Huawei Exec: SDN's Become a 'Completely Meaningless Term' " (http://www.lightreading.com/carrier-sdn/sdn-architectures/huawei-exec-sdns-become-a-completely-meaningless-term/d/d-id/726364).
23. "Software-Defined Networking (SDN) Definition" (http://www.opennetworking.org/sdn-resources/sdn-definition). *Opennetworking.org*. Retrieved 26 October 2014.
24. "White Papers" (http://www.opennetworking.org/sdn-resources/sdn-library/whitepapers). *Opennetworking.org*. Retrieved 26 October 2014.
25. Montazerolghaem, Ahmadreza.; Yaghmaee, M. H.; Leon-Garcia, A. (2017). "OpenSIP: Toward Software-Defined SIP Networking". *IEEE Transactions on Network and Service Management*. **PP** (99): 184–199. arXiv:1709.01320 (https://arxiv.org/abs/1709.01320). Bibcode:2017arXiv170901320M (https://ui.adsabs.harvard.edu/abs/2017arXiv170901320M). doi:10.1109/tnsm.2017.2741258 (https://doi.org/10.1109%2Ftnsm.2017.2741258). ISSN 1932-4537 (https://www.worldcat.org/issn/1932-4537).
26. Vicentini, Cleverton; Santin, Altair; Viegas, Eduardo; Abreu, Vilmar (January 2019). "SDN-based and multitenant-aware resource provisioning mechanism for cloud-based big data streaming". *Journal of Network and Computer Applications*. **126**: 133–149. doi:10.1016/j.jnca.2018.11.005 (https://doi.org/10.1016%2Fj.jnca.2018.11.005).
27. "SDN Architecture Overview" (http://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/SDN-architecture-overview-1.0.pdf) (PDF). *Opennetworking.org*. Retrieved 22 November 2014.
28. S.H. Yeganeh, Y. Ganjali, "Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications," proceedings of HotSDN, Helsinki, Finland, 2012.
29. R. Ahmed, R. Boutaba, "Design considerations for managing wide area software defined networks," Communications Magazine, IEEE, vol. 52, no. 7, pp. 116–123, July 2014.
30. T. Koponen et al, "Onix: A Distributed Control Platform for Large scale Production Networks," proceedings USENIX, ser. OSDI'10, Vancouver, Canada, 2010.
31. D. Tuncer, M. Charalambides, S. Clayman, G. Pavlou, "Adaptive Resource Management and Control in Software Defined Networks," Network and Service Management, IEEE Transactions on, vol. 12, no. 1, pp. 18–33, March 2015.
32. B. Heller, R. Sherwood, and N. McKeown, "The Controller Placement Problem," proceedings of HotSDN'12, 2012.
33. Y.N. Hu, W.D. Wang, X.Y. Gong, X.R. Que, S.D. Cheng, "On the placement of controllers in software-defined networks," Journal of China Universities of Posts and Telecommunications, vol. 19, Supplement 2, no. 0, pp. 92 – 171, 2012.
34. F.J. Ros, P.M. Ruiz, "Five nines of southbound reliability in software defined networks," proceedings of HotSDN'14, 2014.
35. D. Tuncer, M. Charalambides, S. Clayman, G. Pavlou, "On the Placement of Management and Control Functionality in Software Defined Networks," proceedings of 2nd IEEE International Workshop on Management of SDN and NFV Systems (ManSDN/NFV), Barcelona, Spain, November 2015.
36. "OpenFlow: Proactive vs Reactive" (http://networkstatic.net/openflow-proactive-vs-reactive-flows/). *NetworkStatic.net*. 2013-01-15. Retrieved 2014-07-01.
37. "Reactive, Proactive, Predictive: SDN Models | F5 DevCentral" (https://devcentral.f5.com/articles/reactive-proactive-predictive-sdn-models). *Devcentral.f5.com*. 2012-10-11. Retrieved 2016-06-30.

38. Pentikousis, Kostas; Wang, Yan; Hu, Weihua (2013). "Mobileflow: Toward software-defined mobile networks". *IEEE Communications Magazine*. **51** (7): 44–53. doi:10.1109/MCOM.2013.6553677 (https://doi.org/10.1109%2FMCOM.2013.6553677).
39. Liyanage, Madhusanka (2015). *Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture* (http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118900286.html). UK: John Wiley. pp. 1–438. ISBN 978-1-118-90028-4.
40. Jose Costa-Requena, Jesús Llorente Santos, Vicent Ferrer Guasch, Kimmo Ahokas, Gopika Premsankar, Sakari Luukkainen, Ijaz Ahmed, Madhusanka Liyanage, Mika Ylianttila, Oscar López Pérez, Mikel Uriarte Itzazelaia, Edgardo Montes de Oca, *SDN and NFV Integration in Generalized Mobile Network Architecture (http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=7194059&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel7%2F7179207%2F7194024%2F07194059.pdf%3Farnumber%3D7194059)* , in Proc. of European Conference on Networks and Communications (EUCNC), Paris, France. June 2015.
41. Madhusanka Liyanage, Mika Ylianttila, Andrei Gurtov, Securing the Control Channel of Software-Defined Mobile Networks (http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6918981&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6918981) , in Proc. of IEEE 15th International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM), Sydney, Australia. June 2014.
42. Haranas, Mark (8 October 2016). "16 Hot Networking Products Putting The Sizzle In SD-WAN" (http://www.crn.com/slide-shows/networking/300082325/16-hot-networking-products-putting-the-sizzle-in-sd-wan.htm?itc=hp_slideshow). CRN. Retrieved 1 November 2016.
43. "SD-WAN: What it is and why you'll use it one day" (http://www.networkworld.com/article/3031279/internet/sd-wan-what-it-is-and-why-you-ll-use-it-one-day.html). networkworld.com. 2016-02-10. Retrieved 2016-06-27.
44. Serries, William (12 September 2016). "SD-LAN et SD-WAN : Deux Approches Différentes pour le Software Defined Networking" (http://www.zdnet.fr/actualites/sd-lan-et-sd-wan-deux-approches-differentes-pour-le-software-defined-networking-39841794.htm). ZDNet. Retrieved 1 November 2016.
45. Kerravala, Zeus (13 September 2016). "Aerohive Introduces the Software-defined LAN" (http://www.networkworld.com/article/3119598/lan-wan/aerohive-introduces-the-software-defined-lan.html). Network World. Retrieved 1 November 2016.
46. Kreutz, Diego; Ramos, Fernando; Verissimo, Paulo (2013). "Towards secure and dependable software-defined networks". *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. pp. 50–60.
47. Scott-Hayward, Sandra; O'Callaghan, Gemma; Sezer, Sakir (2013). "SDN security: A survey". *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*. pp. 1–7.
48. Benton, Kevin; Camp, L Jean; Small, Chris (2013). "Openflow vulnerability assessment". *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. pp. 151–152.
49. Abdou, AbdelRahman; van Oorschot, Paul; Wan, Tao (May 2018). "A Framework and Comparative Analysis of Control Plane Security of SDN and Conventional Networks". *IEEE Communications Surveys and Tutorials*. to appear. arXiv:1703.06992 (https://arxiv.org/abs/1703.06992). Bibcode:2017arXiv170306992A (https://ui.adsabs.harvard.edu/abs/2017arXiv170306992A).
50. Giotis, K; Argyropoulos, Christos; Androulidakis, Georgios; Kalogeras, Dimitrios; Maglaris, Vasilis (2014). "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments" (https://zenodo.org/record/3415467). *Computer Networks*. **62**: 122–136. doi:10.1016/j.bjp.2013.10.014 (https://doi.org/10.1016%2Fj.bjp.2013.10.014).
51. Braga, Rodrigo; Mota, Edjard; Passito, Alexandre (2010). "Lightweight DDoS flooding attack detection using NOX/OpenFlow". *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*. pp. 408–415.
52. Feamster, Nick (2010). "Outsourcing home network security". *Proceedings of the 2010 ACM SIGCOMM workshop on Home networks*. pp. 37–42.

53. Jin, Ruofan & Wang, Bing (2013). "Malware detection for mobile devices using software-defined networking". *Research and Educational Experiment Workshop (GREE), 2013 Second GENI*. 81-88.
54. Jafarian, Jafar Haadi; Al-Shaer, Ehab; Duan, Qi (2012). "Openflow random host mutation: transparent moving target defense using software defined networking". *Proceedings of the first workshop on Hot topics in software defined networks*. pp. 127–132.
55. Kampanakis, Panos; Perros, Harry; Beyene, Tsegereda. *SDN-based solutions for Moving Target Defense network protection* (http://www4.ncsu.edu/~hp/Panos.pdf) (PDF). Retrieved 23 July 2014.
56. Sherwood, Rob; Gibb, Glen; Yap, Kok-Kiong; Appenzeller, Guido; Casado, Martin; McKeown, Nick; Parulkar, Guru (2009). "Flowvisor: A network virtualization layer". *OpenFlow Switch Consortium, Tech. Rep*.
57. Al-Shaer, Ehab & Al-Haj, Saeed (2010). "FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures". *Proceedings of the 3rd ACM workshop on Assurable and usable security configuration*. pp. 37–44.
58. Canini, Marco; Venzano, Daniele; Peresini, Peter; Kostic, Dejan; Rexford, Jennifer; et al. (2012). *A NICE Way to Test OpenFlow Applications*. NSDI. pp. 127–140.
59. Bernardo and Chua (2015). *Introduction and Analysis of SDN and NFV Security Architecture (SA-SECA)*. 29th IEEE AINA 2015. pp. 796–801.
60. B. Pfaf; et al. (February 28, 2011). "OpenFlow Switch Specification" (http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf) (PDF). Retrieved July 8, 2017.
61. T. Zhu; et al. (October 18, 2016). "MCTCP: Congestion-aware and robust multicast TCP in Software-Defined networks" (https://ieeexplore.ieee.org/document/7590433/). *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*. IEEE. pp. 1–10. doi:10.1109/IWQoS.2016.7590433 (https://doi.org/10.1109%2FIWQoS.2016.7590433). ISBN 978-1-5090-2634-0. Retrieved July 3, 2017.
62. M. Noormohammadpour; et al. (July 10, 2017). "DCCast: Efficient Point to Multipoint Transfers Across Datacenters" (https://www.researchgate.net/publication/316921061). USENIX. Retrieved July 3, 2017.
63. M. Noormohammadpour; et al. (2018). *QuickCast: Fast and Efficient Inter-Datacenter Transfers using Forwarding Tree Cohorts* (https://www.researchgate.net/publication/322243498). arXiv:1801.00837 (https://arxiv.org/abs/1801.00837). Bibcode:2018arXiv180100837N (https://ui.adsabs.harvard.edu/abs/2018arXiv180100837N). doi:10.31219/osf.io/uzr24 (https://doi.org/10.31219%2Fosf.io%2Fuzr24). Retrieved January 23, 2018.
64. William, Stalling (2016). "Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud". *Pearson Education*.
65. Rowayda, A. Sadek (May 2018). "An Agile Internet of Things (IoT) based Software Defined Network (SDN) Architecture". *Egyptian Computer Science Journal*. **42** (2): 13–29.
66. Platform to Multivendor Virtual and Physical Infrastructure (http://www.cisco.com/go/esp)
67. Graham, Finnie (December 2012). "The Role Of DPI In An SDN World". *White Paper*.
68. Series, Y. (May 2015). "Global Information Infrastructure, Internet Protocol Aspects And NextGeneration Networks". *ITU-T Y.2770 Series, Supplement on DPI Use Cases and Application Scenarios*.
