WIKIPEDIA
The Free Encyclopedia

Software-defined networking

Software-defined networking (**SDN**) is an approach to network management that uses abstraction to enable dynamic and programmatically efficient network configuration to create grouping and segmentation while improving network performance and monitoring in a manner more akin to cloud computing than to traditional network management.[1] SDN is meant to improve the static architecture of traditional networks and may be employed to centralize network intelligence in one network component by disassociating the forwarding process of network packets (data plane) from the routing process (control plane).[2] The control plane consists of one or more controllers, which are considered the brains of the SDN network, where the whole intelligence is incorporated. However, centralization has certain drawbacks related to security,[1] scalability and elasticity.[1][3]

SDN was commonly associated with the OpenFlow protocol for remote communication with network plane elements to determine the path of network packets across network switches since OpenFlow's emergence in 2011. However, since 2012, proprietary systems have also used the term.[4][5] These include Cisco Systems' Open Network Environment and Nicira's network virtualization platform.

SD-WAN applies similar technology to a wide area network (WAN).[6]

In addition to WAN use cases, SDN is also increasingly applied in telecom backbones, where it is paired with operational inventory and automation platforms to provide real-time visibility, service assurance, and simplified provisioning across multi-vendor networks.[7][8][9]

History

The history of SDN principles can be traced back to the separation of the control and data planes first used in public switched telephone networks. This provided a manner of simplifying provisioning and management years before the architecture was used in data networks.

The Internet Engineering Task Force (IETF) began considering various ways to decouple the control and data forwarding functions in a proposed interface standard published in 2004 named Forwarding and Control Element Separation (ForCES).[10] The ForCES Working Group also proposed a companion SoftRouter architecture.[11] Additional early standards from the IETF that pursued separating control from data planes include the Linux Netlink as an IP services protocol,[12] and a path computation element (PCE)-based architecture.[13]

These early attempts failed to gain traction. One reason is that many in the Internet community viewed separating control from data to be risky, especially given the potential for failure in the control plane. Another reason 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 these separated architectures traces its roots to the Ethane project at Stanford's computer science department. Ethane's simple switch design led to the creation of OpenFlow,[14] and an API for OpenFlow was first created in 2008.[15] In that same year, NOX, an operating system for networks, was created.[16]

SDN research included emulators such as vSDNEmul,[17] EstiNet,[18] and Mininet.[19]

Work on OpenFlow continued at Stanford, including with the creation of testbeds to evaluate the use of the protocol in a single campus network, as well as across the WAN as a backbone for connecting multiple campuses.[20] In academic settings, there were several research and production networks based on OpenFlow switches from NEC and Hewlett-Packard, as well as those based on Quanta Computer white-boxes starting in about 2009.[21]

Beyond academia, the first deployments were by Nicira in 2010 to control OVS from Onix, codeveloped with NTT and Google. A notable deployment was Google's B4 in 2012.[22][23] Later, Google announced the first OpenFlow/Onix deployments in is datacenters.[24] Another large deployment exists at China Mobile.[25]

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 and removing manual provisioning from service delivery.[26][27]

Concept

SDN architectures decouple network control (control plane) and forwarding (data plane) functions, enabling the network control to become directly programmable and the underlying infrastructure to be abstracted from applications and network services.[28] The OpenFlow protocol is one of the protocols 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.[29]
- 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.

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.[30] 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 may be ill-suited to the dynamic computing and storage needs of today's enterprise data centers, campuses, and carrier environments.[31] 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, connecting from anywhere, at any time. Finally, many enterprise data center managers are deploying a utility computing model, which may 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.

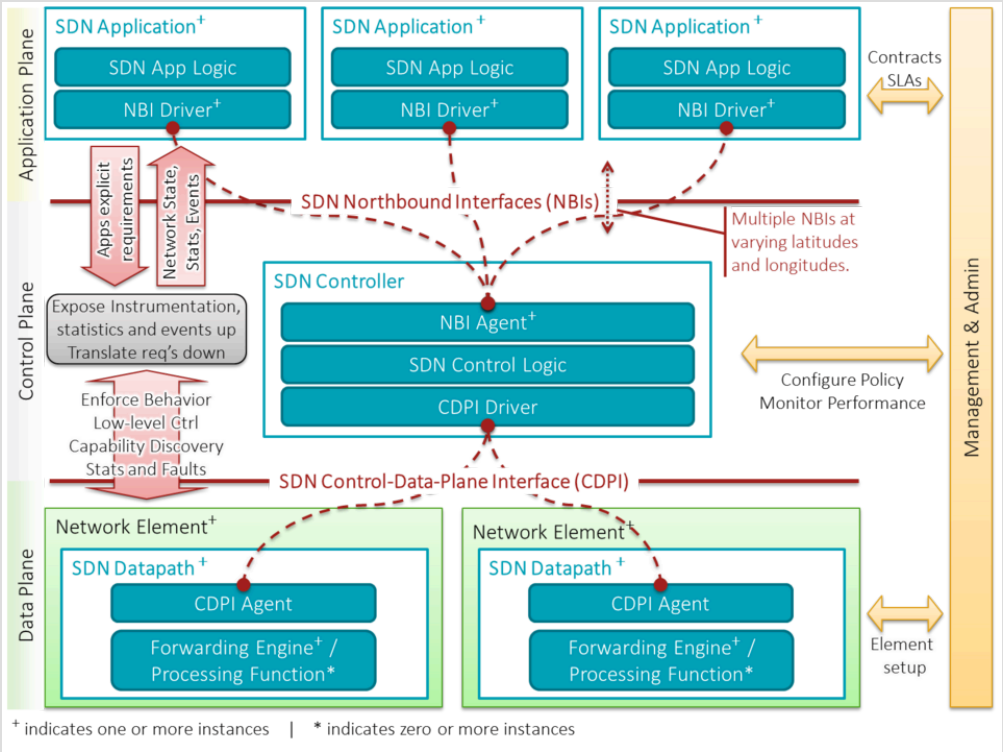
The rise of cloud services
: Enterprises have enthusiastically embraced both public and private cloud services, resulting in unprecedented growth of these services. Many enterprise businesses want the agility to access applications, infrastructure and other IT resources on demand and discretely. IT planning for cloud services must be performed in an environment of increased security, compliance and auditing requirements, along with business reorganizations, consolidations and mergers that can rapidly change assumptions. 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 requires massive parallel processing on thousands of servers, all of which need direct connections to each other. The rise of these large data sets 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 within a limited budget.[32]

Energy use in large data centers
: As Internet of things, cloud computing and SaaS emerged, the need for larger data centers has increased the energy consumption of those facilities. Many researchers have improved SDN's energy efficiency applying existing routing techniques to dynamically adjust the network data plane to save energy.[33] Also techniques to improve control plane energy efficiency are being researched.[34]

Architectural components

The following list defines and explains the SDN architectural components:[35]



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

SDN application

SDN applications are programs that 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 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. The controller's definition as a logically centralized entity neither prescribes nor precludes implementation architectures 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 capabilities. An SDN datapath consists of 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 programmatic control of all forwarding operations, capabilities advertisement, statistics reporting, and 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 and across different sets of functionality.

SDN control plane

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 had 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, hierarchical and distributed approaches have been proposed. In hierarchical solutions,[36][37] 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,[38][39] 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.

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,[40] especially in the context of large networks. Other objectives that have been considered involve control path reliability,[41] fault tolerance,[42] and application requirements.[43]

SDN data plane

In SDN, the data plane is responsible for processing data-carrying packets using a set of rules specified by the control plane. The data plane may be implemented in physical hardware switches or in software, such as Open vSwitch. The memory capacity of hardware switches may limit the number of rules that can be stored whereas software implementations may have higher capacity.[44]

The location of the SDN data plane and agent can be used to classify SDN implementations:

- Hardware switch-based: This approach implements the data plane processing inside a physical device. OpenFlow switches may use TCAM tables to route packet sequences (flows). These switches may use an ASIC for their implementation.
- Software switch-based: Some physical switches may implement SDN support using software on the device to populate flow tables and to act as the SDN agent when communicating with the

controller. Hypervisors may likewise use software implementations to support SDN protocols in the virtual switches used to support their virtual machines.

- Host-based: Rather than deploying the data plane and SDN agent in network infrastructure, host-based SDNs deploy the SDN agent inside the operating system of the communicating endpoints.[45] Such implementations can provide additional context about the application, user, and activity associated with network flows.[46] To achieve the same traffic engineering capabilities of switch-based SDNs, host-based SDNs may require the use of carefully designed VLAN and spanning tree assignments.[47]

Flow table entries may be populated in a proactive, reactive, or hybrid fashion.[48][49] In the proactive mode, the controller populates flow table entries for all possible traffic matches 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 at line rate (considering all flow table entries in TCAM) and no delay is added. In the reactive mode, entries are populated on demand. If a packet arrives without a corresponding match rule in the flow table, the SDN agent sends a request to the controller for further instruction. The controller examines the SDN agent requests and provides instructions, installing a rule in the flow table for the corresponding packet if necessary. The hybrid mode uses the low-latency proactive forwarding mode for a portion of traffic while relying on the flexibility of reactive mode processing for the remaining traffic.

Applications

SDMN

Software-defined mobile networking (SDMN)[50][51] 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.[52] It is proposed as an extension of SDN paradigm to incorporate mobile network specific functionalities.[53] 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 WAN managed using the principles of software-defined networking.[54] 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.[55]

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.[56] 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.[57]

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 the security of SDN architecture itself remains an open question that has already been studied a couple of times in the research community,[58][59][60][61] 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,[62][63] as well as botnet[64] and worm propagation,[65] 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 hidden 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.[66] 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.[67]

Additional value regarding security in SDN enabled networks can also be gained using FlowVisor[68] and FlowChecker[69] 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[68] 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.[70] 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.[71]

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[72] 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[73] is a scheme for delivery to many nodes inside datacenters that relies on regular and structured topologies of datacenter networks while DCCast[74] and QuickCast[75] are approaches for fast and efficient data and content replication across datacenters over private WANs.

Relationship to NFV

Network Function Virtualization, or NFV for short, is a concept that complements SDN. Thus, NFV is not dependent on SDN or SDN concepts. NFV separates 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).[76] 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.[76][77] 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.[78]

Relationship to DPI

Deep packet inspection (DPI) provides network with application-awareness, while SDN provides applications with network-awareness.[79] 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 [80]

Quality of Experience (QoE) estimation using SDN

When using an SDN based model for transmitting multimedia traffic, an important aspect to take account is the QoE estimation. To estimate the QoE, first we have to be able to classify the traffic and then, it's recommended that the system can solve critical problems on its own by analyzing the traffic.[81][82]

See also

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

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. Montazerolghaem, Ahmadreza (2020-07-13). "Software-defined load-balanced data center: design, implementation and performance analysis" (http://link.springer.com/10.1007/s10586-020-03134-x). *Cluster Computing*. **24** (2): 591–610. doi:10.1007/s10586-020-03134-x (https://doi.org/10.1007%2Fs10586-020-03134-x). ISSN 1386-7857 (https://search.worldcat.org/issn/1386-7857). S2CID 220490312 (https://api.semanticscholar.org/CorpusID:220490312).
3. Montazerolghaem, Ahmadreza (2021). "Software-defined Internet of Multimedia Things: Energy-efficient and Load-balanced Resource Management" (https://hal.science/hal-03281413). *IEEE Internet of Things Journal*. **9** (3): 2432–2442. doi:10.1109/JIOT.2021.3095237 (https://doi.org/10.1109%2FJIOT.2021.3095237). ISSN 2327-4662 (https://search.worldcat.org/issn/2327-4662). S2CID 237801052 (https://api.semanticscholar.org/CorpusID:237801052).
4. "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*.

5. "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). 10 February 2016.
6. "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.
7. Mijumbi, R.; Serrat, J.; Gorricho, J. L.; Bouten, N.; De Turck, F.; Boutaba, R. (2016). "Network Function Virtualization: State-of-the-Art and Research Challenges". *IEEE Communications Surveys & Tutorials*. **18** (1): 236–262. arXiv:1509.07675 (https://arxiv.org/abs/1509.07675). doi:10.1109/COMST.2015.2477041 (https://doi.org/10.1109%2FCOMST.2015.2477041).
8. Li, Y.; Chen, M. (2015). "Software-Defined Network Function Virtualization: A Survey" (https://doi.org/10.1109%2FACCESS.2015.2499271). *IEEE Access*. **3**: 2542–2553. doi:10.1109/ACCESS.2015.2499271 (https://doi.org/10.1109%2FACCESS.2015.2499271).
9. Scott-Hayward, S.; Natarajan, S.; Sezer, S. (2016). "A Survey of Security in Software Defined Networks". *IEEE Communications Surveys & Tutorials*. **18** (1): 623–654. doi:10.1109/COMST.2015.2453114 (https://doi.org/10.1109%2FCOMST.2015.2453114).
10. 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://www.rfc-editor.org/rfc/rfc3746). Internet Engineering Task Force. doi:10.17487/RFC3746 (https://doi.org/10.17487%2FRFC3746). RFC 3746 (https://datatracker.ietf.org/doc/html/rfc3746).
11. 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).
12. 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). doi:10.17487/RFC3549 (https://doi.org/10.17487%2FRFC3549). `{{cite journal}}`: Cite journal requires `|journal=` (help)
13. 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). doi:10.17487/RFC4655 (https://doi.org/10.17487%2FRFC4655). `{{cite journal}}`: Cite journal requires `|journal=` (help)
14. Martìn Casado, Michael J. Freedman, Justin Pettit, Jianying Luo, and Nick McKeown (Stanford University) (August 2007). "Ethane: Taking Control of the Enterprise" (https://cs.brown.edu/courses/csci2950-u/s14/papers/Casado07Ethane.pdf) (PDF).
15. 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).
16. 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).
17. 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)].
18. Wang, S.; Chou, C.; Yang, C. (September 2013). "EstiNet openflow network simulator and emulator". *IEEE Communications Magazine*. **51** (9): 110–117. Bibcode:2013IComM..51i.110W (https://ui.adsabs.harvard.edu/abs/2013IComM..51i.110W). doi:10.1109/MCOM.2013.6588659 (https://doi.org/10.1109%2FMCOM.2013.6588659). ISSN 1558-1896 (https://search.worldcat.org/issn/1558-1896). S2CID 14375937 (https://api.semanticscholar.org/CorpusID:14375937).

19. 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". *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*. pp. 1–6. doi:10.1109/ColComCon.2014.6860404 (https://doi.org/10.1109%2FColComCon.2014.6860404). ISBN 978-1-4799-4340-1. S2CID 17915639 (https://api.semanticscholar.org/CorpusID:17915639).
20. "GENI. Campus OpenFlow topology" (http://groups.geni.net/geni/wiki/OpenFlow/CampusTopology). 2011.
21. Kuang-Ching "KC" Wang (Oct 3, 2011). "Software Defined Networking and OpenFlow for Universities: Motivation, Strategy, and Uses" (https://web.archive.org/web/20180103133512/https://www.internet2.edu/presentations/fall11/20111003-wang-openflow.pdf) (PDF). Archived from the original (https://www.internet2.edu/presentations/fall11/20111003-wang-openflow.pdf) (PDF) on 2018-01-03.
22. 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). {{cite web}}: |author= has generic name (help)
23. brent salisbury (May 14, 2013). "Inside Google's Software-Defined Network" (https://www.networkcomputing.com/networking/inside-googles-software-defined-network/512240144). *Network Computing*.
24. 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).
25. " "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.
26. 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).
27. 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). *Light Reading*.
28. "Software-Defined Networking (SDN) Definition" (http://www.opennetworking.org/sdn-resources/sdn-definition). *Opennetworking.org*. Retrieved 26 October 2014.
29. Montazerolghaem, Ahmadreza; Yaghmaee, Mohammad Hossein; Leon-Garcia, Alberto (September 2020). "Green Cloud Multimedia Networking: NFV/SDN Based Energy-Efficient Resource Allocation" (https://hal.science/hal-02519007). *IEEE Transactions on Green Communications and Networking*. **4** (3): 873–889. Bibcode:2020ITGCN...4..873M (https://ui.adsabs.harvard.edu/abs/2020ITGCN...4..873M). doi:10.1109/TGCN.2020.2982821 (https://doi.org/10.1109%2FTGCN.2020.2982821). ISSN 2473-2400 (https://search.worldcat.org/issn/2473-2400). S2CID 216188024 (https://api.semanticscholar.org/CorpusID:216188024).
30. "White Papers" (http://www.opennetworking.org/sdn-resources/sdn-library/whitepapers). *Opennetworking.org*. Retrieved 26 October 2014.

31. 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://search.worldcat.org/issn/1932-4537). S2CID 3873601 (https://api.semanticscholar.org/CorpusID:3873601).
32. 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). S2CID 57941895 (https://api.semanticscholar.org/CorpusID:57941895).
33. Assefa, Beakal Gizachew; Özkasap, Öznur (June 2020). "RESDN: A Novel Metric and Method for Energy Efficient Routing in Software Defined Networks" (https://doi.org/10.1109/TNSM.2020.2973621). *IEEE Transactions on Network and Service Management*. **17** (2): 736–749. arXiv:1905.12219 (https://arxiv.org/abs/1905.12219). Bibcode:2020ITNSM..17..736A (https://ui.adsabs.harvard.edu/abs/2020ITNSM..17..736A). doi:10.1109/TNSM.2020.2973621 (https://doi.org/10.1109%2FTNSM.2020.2973621). S2CID 199442001 (https://api.semanticscholar.org/CorpusID:199442001).
34. Oliveira, Tadeu F.; Xavier-de-Souza, Samuel; Silveira, Luiz F. (May 2021). "Improving Energy Efficiency on SDN Control-Plane Using Multi-Core Controllers" (https://doi.org/10.3390%2Fen14113161). *Energies*. **14** (11): 3161. doi:10.3390/en14113161 (https://doi.org/10.3390%2Fen14113161).
35. "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.
36. Yeganeh, S.H.; Ganjali, Y. "Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications" (https://doi.org/10.1145%2F2342441.2342446). doi:10.1145/2342441.2342446 (https://doi.org/10.1145%2F2342441.2342446). S2CID 193153 (https://api.semanticscholar.org/CorpusID:193153). {{cite journal}}: Cite journal requires |journal= (help)
37. Ahmed, R.; Boutaba, R. (2014). "Design considerations for managing wide area software defined networks". *IEEE Communications Magazine*. **52** (7): 116–123. Bibcode:2014IComM..52g.116A (https://ui.adsabs.harvard.edu/abs/2014IComM..52g.116A). doi:10.1109/MCOM.2014.6852092 (https://doi.org/10.1109%2FMCOM.2014.6852092). S2CID 7912785 (https://api.semanticscholar.org/CorpusID:7912785).
38. Koponen, T. (2010). "Onix: A Distributed Control Platform for Large scale Production Networks" (https://www.usenix.org/legacy/events/osdi10/tech/full_papers/Koponen.pdf) (PDF). *Proceedings USENIX, Ser. OSDI'10*. Vancouver, Canada.
39. Tuncer, Daphne; Charalambides, Marinos; Clayman, Stuart; Pavlou, George (March 2015). "Adaptive Resource Management and Control in Software Defined Networks" (https://discovery.ucl.ac.uk/id/eprint/1478892/). *IEEE Transactions on Network and Service Management*. **12** (1): 18–33. Bibcode:2015ITNSM..12...18T (https://ui.adsabs.harvard.edu/abs/2015ITNSM..12...18T). doi:10.1109/TNSM.2015.2402752 (https://doi.org/10.1109%2FTNSM.2015.2402752). hdl:10044/1/63600 (https://hdl.handle.net/10044%2F1%2F63600). S2CID 9215618 (https://api.semanticscholar.org/CorpusID:9215618).
40. Heller, B.; Sherwood, R.; McKeown, N. (2012). "The Controller Placement Problem". *Proceedings of the first workshop on Hot topics in software defined networks - HotSDN '12*. p. 7. doi:10.1145/2342441.2342444 (https://doi.org/10.1145%2F2342441.2342444). ISBN 9781450314770. S2CID 1770114 (https://api.semanticscholar.org/CorpusID:1770114).

41. Hu, Yan-nan; Wang, Wen-Dong; Gong, Xiang-Yang; Que, Xi-Rong; Cheng, Shi-Duan (2012). "On the placement of controllers in software-defined networks". *The Journal of China Universities of Posts and Telecommunications*. **19**: 92–171. doi:10.1016/S1005-8885(11)60438-X (https://doi.org/10.1016%2FS1005-8885%2811%2960438-X).
42. Ros, Francisco Javier; Ruiz, Pedro Miguel (2014). "Five nines of southbound reliability in software-defined networks". *Proceedings of the third workshop on Hot topics in software defined networking*. pp. 31–36. doi:10.1145/2620728.2620752 (https://doi.org/10.1145%2F2620728.2620752). ISBN 9781450329897. S2CID 17088018 (https://api.semanticscholar.org/CorpusID:17088018).
43. Tuncer, Daphne; Charalambides, Marinos; Clayman, Stuart; Pavlou, George (2015). "On the Placement of Management and Control Functionality in Software Defined Networks". *2015 11th International Conference on Network and Service Management (CNSM)* (https://discovery.ucl.ac.uk/id/eprint/1502968/). pp. 360–365. doi:10.1109/CNSM.2015.7367383 (https://doi.org/10.1109%2FCNSM.2015.7367383). ISBN 978-3-9018-8277-7. S2CID 6977724 (https://api.semanticscholar.org/CorpusID:6977724).
44. Wang, An; Guo, Yang; Hao, Fang; Lakshman, T.; Chen, Songqing (2 December 2014). "Scotch: Elastically Scaling up SDN Control-Plane using vSwitch based Overlay" (http://conferences2.sigcomm.org/co-next/2014/CoNEXT_papers/p403.pdf) (PDF). *ACM CoNEXT*.
45. Taylor, Curtis; MacFarland, Douglas; Smestad, Doran; Shue, Craig (10 April 2014). "Contextual, flow-based access control with scalable host-based SDN techniques" (https://par.nsf.gov/servlets/purl/10055769). *IEEE INFOCOM 2016 - the 35th Annual IEEE International Conference on Computer Communications*. pp. 1–9. doi:10.1109/INFOCOM.2016.7524498 (https://doi.org/10.1109%2FINFOCOM.2016.7524498). ISBN 978-1-4673-9953-1. S2CID 17491115 (https://api.semanticscholar.org/CorpusID:17491115).
46. Chuluundorj, Zorigtbaatar; Taylor, Curtis; Walls, Robert; Shue, Craig (6 December 2021). "Can the User Help? Leveraging User Actions for Network Profiling". *2021 Eighth International Conference on Software Defined Systems (SDS)*. pp. 1–8. doi:10.1109/SDS54264.2021.9732164 (https://doi.org/10.1109%2FSDS54264.2021.9732164). ISBN 978-1-6654-5820-7. S2CID 244036711 (https://api.semanticscholar.org/CorpusID:244036711).
47. Lei, Yunsen; Lanson, Julian; Kaldawy, Remy; Estrada, Jeffrey; Shue, Craig (11 November 2020). "Can Host-Based SDNS Rival the Traffic Engineering Abilities of Switch-Based SDNS?". *2020 11th International Conference on Network of the Future (NoF)*. pp. 91–99. doi:10.1109/NoF50125.2020.9249110 (https://doi.org/10.1109%2FNoF50125.2020.9249110). ISBN 978-1-7281-8055-7. S2CID 221505891 (https://api.semanticscholar.org/CorpusID:221505891).
48. "OpenFlow: Proactive vs Reactive" (http://networkstatic.net/openflow-proactive-vs-reactive-flows/). *NetworkStatic.net*. 2013-01-15. Retrieved 2014-07-01.
49. "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.
50. Pentikousis, Kostas; Wang, Yan; Hu, Weihua (2013). "Mobileflow: Toward software-defined mobile networks". *IEEE Communications Magazine*. **51** (7): 44–53. Bibcode:2013IComM..51g..44P (https://ui.adsabs.harvard.edu/abs/2013IComM..51g..44P). doi:10.1109/MCOM.2013.6553677 (https://doi.org/10.1109%2FMCOM.2013.6553677). S2CID 10655582 (https://api.semanticscholar.org/CorpusID:10655582).
51. 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.

52. Costa-Requena, Jose; Liyanage, Madhusanka; Ylianttila, Mika; De Oca, Edgardo Montes; Santos, Jesus Llorente; Guasch, Vicent Ferrer; Ahokas, Kimmo; Premsankar, Gopika; Luukkainen, Sakari; Perez, Oscar Lopez; Itzazelaia, Mikel Uriarte; Ahmad, Ijaz (2015). "SDN and NFV Integration in Generalized Mobile Network Architecture". *2015 European Conference on Networks and Communications (EuCNC)*. pp. 154–158. doi:10.1109/EuCNC.2015.7194059 (https://doi.org/10.1109%2FEuCNC.2015.7194059). ISBN 978-1-4673-7359-3. S2CID 2453962 (https://api.semanticscholar.org/CorpusID:2453962).
53. Liyanage, Madhusanka; Ylianttila, Mika; Gurtov, Andrei (2014). "Securing the Control Channel of Software-Defined Mobile Networks". *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. pp. 1–6. doi:10.1109/WoWMoM.2014.6918981 (https://doi.org/10.1109%2FWoWMoM.2014.6918981). ISBN 978-1-4799-4786-7. S2CID 1378181 (https://api.semanticscholar.org/CorpusID:1378181).
54. 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.
55. "SD-WAN: What it is and why you'll use it one day" (https://www.networkworld.com/article/947944/sd-wan-what-it-is-and-why-you-ll-use-it-one-day.html). Network World. 2016-02-10. Retrieved 2016-06-27.
56. 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.
57. Kerravala, Zeus (13 September 2016). "Aerohive Introduces the Software-defined LAN" (https://www.networkworld.com/article/955256/aerohive-introduces-the-software-defined-lan.html). Network World. Retrieved 1 November 2016.
58. 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.
59. 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.
60. 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.
61. 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).
62. 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).
63. 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.
64. Feamster, Nick (2010). "Outsourcing home network security". *Proceedings of the 2010 ACM SIGCOMM workshop on Home networks*. pp. 37–42.
65. 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.

66. 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.
67. Kampanakis, Panos; Perros, Harry; Beyene, Tsegereda. *SDN-based solutions for Moving Target Defense network protection* (https://people.engr.ncsu.edu/hp/papers/Panos.pdf) (PDF). Retrieved 16 February 2022.
68. 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*.
69. 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.
70. Canini, Marco; Venzano, Daniele; Peresini, Peter; Kostic, Dejan; Rexford, Jennifer; et al. (2012). *A NICE Way to Test OpenFlow Applications*. NSDI. pp. 127–140.
71. Bernardo and Chua (2015). *Introduction and Analysis of SDN and NFV Security Architecture (SA-SECA)*. 29th IEEE AINA 2015. pp. 796–801.
72. 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.
73. T. Zhu; et al. (October 18, 2016). "MCTCP: Congestion-aware and robust multicast TCP in Software-Defined networks". *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. S2CID 28159768 (https://api.semanticscholar.org/CorpusID:28159768).
74. 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.
75. 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.
76. William, Stalling (2016). "Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud". *Pearson Education*.
77. 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.
78. "Platform to Multivendor Virtual and Physical Infrastructure" (http://www.cisco.com/go/esp).
79. Graham, Finnie (December 2012). "The Role Of DPI In An SDN World". *White Paper*.
80. Series, Y. (May 2015). "Global Information Infrastructure, Internet Protocol Aspects And Next Generation Networks". *ITU-T Y.2770 Series, Supplement on DPI Use Cases and Application Scenarios*.
81. Canovas, Alejandro (2020). "A robust multimedia traffic SDN-Based management system using patterns and models of QoE estimation with BRNN" (https://www.sciencedirect.com/science/article/pii/S1084804519303583). *Journal of Network and Computer Applications*. **150** 102498. doi:10.1016/j.jnca.2019.102498 (https://doi.org/10.1016%2Fj.jnca.2019.102498). hdl:10251/163292 (https://hdl.handle.net/10251%2F163292). S2CID 210925444 (https://api.semanticscholar.org/CorpusID:210925444).

82. Rego, Albert (2019). "Adapting reinforcement learning for multimedia transmission on SDN" (https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3643). *Transactions on Emerging Telecommunications Technologies*. **30** (9) e3643. doi:10.1002/ett.3643 (https://doi.org/10.1002%2Fett.3643). hdl:10251/186852 (https://hdl.handle.net/10251%2F186852). S2CID 182028234 (https://api.semanticscholar.org/CorpusID:182028234).

Retrieved from "https://en.wikipedia.org/w/index.php?title=Software-defined_networking&oldid=1322313751"