Evaluation of Software-Defined Networking for Power Systems

Thomas Pfeiffenberger and Jia Lei Du
Advanced Networking Center, Salzburg Research
Jakob Haringer Str. 5/3, 5020 Salzburg, Austria
{thomas.pfeiffenberger, jia.du}@salzburgresearch.at

***Abstract*—A highly available and secure communication infrastructure is one of the major prerequisites for modern power systems. In this paper we evaluate the use of a software-defined networking infrastructure in the domain of energy communication networks. The advantages and potential risks of using software-defined networking at the current stage of development in productive networks are investigated. Test scenarios are defined based on IEC 61850 traffic specification to perform traffic measurements using the OpenFlow standard and real network devices. Software-defined networks have the potential to offer significant advantages over conventional networks. However, we will show through our analysis and measurements that there are still open issues that need to be solved before usage in a productive environment.**

I. Introduction

Electrical grids are evolving into so-called smart grids. Visions of future power systems include increased efficiency, reliability, robustness, faster innovation cycles, and more decentralized power generation with the integration of sustainable energy sources. To achieve these goals, highly sophisticated measurement and control systems are required, which in turn are based on state-of-the-art communication technologies. Communication technologies for smart grids need to be reliable, robust, adaptive, readily deployable, secure and support higher bandwidth than in the past. A multitude of heterogeneous services and applications may have to be supported with different requirements regarding bandwidth and latency. Software-defined networking (SDN) is a new trend in communication technologies in recent years that may help to realize these requirements. In this paper we will evaluate the suitability of software-defined networking for energy communication networks. To the best of our knowledge only one paper has been published that applies the available research in SDN to the area of energy communication networks at the time of writing. In [1] a system is described that enables energy network operators to add and remove IEDs and monitoring nodes to a substation using a configuration loader. The configuration loader parses IED files to extract communications requirements and determine a suitable network configuration that is downloaded to the network devices through an OpenFlow controller. The implementation was tested in an emulated network. In our paper we instead focus on i) the advantages and potential risks of using software-defined networking at the current stage of development in productive networks and ii) investigate if SDNs can fulfill communication requirements of energy communication networks regarding properties like latency and failover times. Software-defined networking offers numerous advantages over conventional networks in the areas of network operation and setup, security and quality of service. But at the current development stage SDN potentially suffers from possible reliability and security issues related to incomplete specifications and the use of a centralized controller. However, solutions for these issues have already been proposed in the research community. We will also show through performance measurements using real network devices that early implementations of software-defined networks already show promising results but may not be suitable for productive use in energy communication networks yet.

Pfeiffenberger, T.; Jia Lei Du, "Evaluation of software-defined networking for power systems," Intelligent Energy and Power Systems (IEPS), 2014 IEEE International Conference on, pp.181-185, 2-6 June 2014 doi: 10.1109/IEPS.2014.6874175, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6874175&isnumber=6874158

©2014 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

II. SDN for Energy Communication Networks

In conventional communication networks, traffic flows are established based on forwarding rules that are locally determined using distributed algorithms. In contrast to this approach, traffic flows in software-defined networks (SDNs) are centrally configured by network applications via so-called controllers [2], [3]. This effectively decouples the control plane, which determines where traffic is sent, from the data plane, which forwards packets to their destinations (Fig. 1). One simple example for a configuration setting in a SDN would be "forward all packets from IP address X to port Y". The configuration of flows can be done proactively or reactively. In the case of using proactive flow configurations, network devices are programmed in advance. When packets that match some pre-programmed patterns arrive at a network device, the respective associated actions are performed. When using reactive flow configurations, packets for which no suitable pre-determined rules can be found are sent from the network devices to the controller. Network applications then determine how to handle the packet and can choose to insert new proactive rules for these packet types in the network devices via the controller. Network applications can be programmed or purchased in application stores and combined to solve specific network problems. To support decision making network applications can obtain detailed traffic statistics from network devices and thus construct an up-to-date global

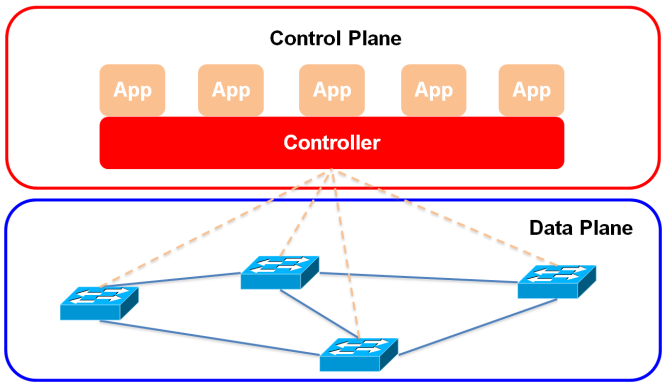

Fig. 1: Software-Defined Networking

network view. One common standard for the implementation of software defined networks is OpenFlow. The OpenFlow standard defines a communication protocol between network switches forming the data plane and one or multiple controllers forming the control plane. In this paper we will use OpenFlow version 1.3 for our analysis of SDN-related risks and benefits.

A. Risk Analysis

One key risk of a SDN is related to the availability of the controller. To mitigate the risk of controller unavailability usually the use of multiple controllers in an SDN is suggested. A solution that is already in productive use is presented in [4]. Network devices are controlled by a single controller with multiple replicated controllers on warm standby. System with multiple active controllers are described in [5], [6]. In difference to the solution above, all controller nodes are actively used for load-balancing. Commercials solutions with teams of controllers and failover mechanisms [7] or up to 5 controllers in active/active mode [8] are also available. However, no further technical details are given in these cases. Another approach to mitigate the effects of a permanent controller failure is to proactively set up important flow paths without an expiration time. The ability to dynamically control flows in the network would be lost during the blackout but the already established flow paths would continue working as pre-programmed.

A second type of potential security vulnerability in OpenFlow-based SDNs is caused by the lack of specifications for granular access control and the resolution of conflicting flow rules. This can result in conflicting flow rules in the network devices caused by network applications with different goals or by multiple controllers accessing the same network device. FlowVisor [9] uses virtualization techniques to create fully separated virtual network slices in the data plane of real network devices. FortNOX [10] administrator flow rules take precedence over security application rules which again take precedence over standard application rules. Resolution strategies are applied in cases when conflicts arise between flow rules issued by applications with the same authorization level. To ensure correctness of the network flows, VeriFlow [11] can be used to check rules to be inserted into network devices by the controller in real-time. Properties that can be verified include reachability, loop-freeness, and consistency.

Potential security issues that originate from the data plane are shown in [12]. SDNs operating in reactive mode are vulnerable to information disclosure attacks. In reactive mode the first packet of a new flow is forwarded to the control plane resulting in a longer round trip time for the corresponding reply. Thus it is possible to find out if a flow between two devices already exists and for example reveal applications that are installed on the devices. This type of attack can only be executed if flow aggregation is used. Flow aggregation summarizes multiple flows in a single rule using wildcards. If the rules inserted in network devices are separated for each flow, this attack is not possible. Other suggested means to counter this type of attack are the use of proactive flows or increasing the variance of measurable response times through randomization. A denial-of-service attack through the creation of packets with randomly modified headers which will eventually lead to overflowing flow tables in the network devices is also described in [12]. One suggested solution is to apply rate limiting to reduce the number of incoming packets per interface.

B. Advantages of SDN

The basic idea of SDN is to route traffic through the network using a central controller. By monitoring network-wide state, the controller obtains an up-to-date view of the network and can dynamically adapt flows in real-time. From a technical point of view, the concept of SDN allows a wide range of traffic engineering and security applications. For example flows can be dynamically rerouted based on load or failure scenarios to guarantee certain bandwidth and latency properties or implement fast failover mechanisms. Or flows could be dynamically and transparently rerouted for security inspection. From an economic point of view, SDN has potential in simplifying and reducing costs of network setup and operation through standardization, centralization, simplified simulation and automatic verification.

In [13] it is shown how software-defined networking based on OpenFlow significantly simplifies the creation of VLANs as the configuration can be performed centrally and not switch-by-switch anymore. More generally, tools like FlowVisor [9] can be used to create isolated virtual networks and simplify the setup of multitenant networks on shared infrastructure. The authors of [4] state that software-defined networking simplifies simulation and verification as it is easier to simulate a deterministic central server than the asynchronous behavior of distributed routing protocols. Using tools like FortNOX [10] and VeriFlow [11] network configurations can be automatically checked in advance or in in real-time for security violations and conflicting or incorrect flow rules before deployment in the real network.

Software-defined networking allows granular dynamic traffic engineering, thus available bandwidth can be efficiently used and there is less need for overprovisioning [4]. Another example for a traffic engineering application is Aster*x, which can be used for load-balancing in arbitrary networks [14]. The authors point out that every service in the network can have its own load-balancing strategy and the load-balancing can be quickly adapted to changing requirements or changes in the network structure.

SDN can be used to centrally control traffic flows and their entire path. This can increase network security as unidirec-

tional networks flows can be created or flows can be routed over devices that are considered more secure. In addition, there exists a wide range of security applications based on OpenFlow. Cloudwatcher [15] uses software defined networking to transparently reroute flows so that every network flow is guaranteed to pass by and be inspected by a network security device. In [16] the controller transparently assigns and frequently but irregularly reassigns random virtual IP addresses to hosts. This approach makes it more difficult for attackers to identify active IP addresses and to identify hosts behind IP address. VAVE [17] helps prevent IP spoofing attacks through source address validation by matching new flows against known flow paths. If a packet arrives from a network device that is not part of the SDN network, the controller checks if the source address of the packet is known to be part of the SDN. If that is the case, the spoofing attack is prevented by denying the requested flow. As a final example, in [18] the information gathering capabilities of OpenFlow are used in combination with self-organizing maps to detect distributed denial-of-service attacks.

III. Communication Requirements

Reliability and robustness in the local area communication network of an electrical substation is a main requirement for such a critical infrastructure that needs to be highly available. Various redundancy protocols and network topologies such as rings are used to make the network resilient against communication failures. It depends on the importance of the application and the impact of a potential failure, which effort is spent to increase the resilience of the communication system [19], [20].

To exchange data objects between Intelligent Electronic Devices (IED), different traffic types and communication requirements for services and applications are defined in [21]. To evaluate a communication infrastructure, knowledge about the different traffic profiles, services and applications is necessary. The quality of the transmission of the different flows is based on the capabilities of the involved communication devices to fulfil these requirements. Queuing delay and loss in communication devices are primarily dependent on the performance of the network devices. In addition, traffic and network management can also help to improve network performance and reliability beyond the capability of a single network device.

IV. Testbed and Testdefinition

The main focus of the testbed is to evaluate the setup and use of a software-defined networking architecture for an electrical substation but not to create a complete and correct model of a substation regarding sent data or the communication of services. The scope of our practical evaluation is to benchmark the communication performance and behaviour of our system under test (SUT). The SUT simulates a very simple IED communication pattern based on substation topology T1-1 as defined on page 18 in [22]. Further requirements for the setup of the testbed are extracted from the ZUQDE project [23]. The first test case evaluates the functionality of quality of service properties for software-defined flows. The second test case evaluates the performance in a failover scenario.

Figure 2 shows the SUT with the PC-based IED emulator and two HP2920 switches. The IED emulator is a Linux-based

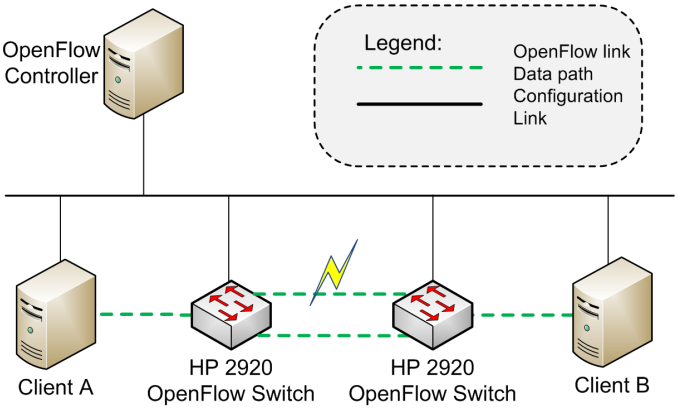

Fig. 2: OFSE Grid Evaluation Testbed

PC with two network cards and a traffic generator. The traffic generator [24] is able to generate network traffic with different traffic profiles, e.g. GOOSE traffic or FTP traffic. The traffic generator sends out packets on client 1 and receives the packets on client 2 for further processing. The outgoing packet flow can be configured with different parameters including packet length or sending interval. Based on the received packets, packet losses, latency and jitter can be calculated.

The ethernet links between the PCs and the HP2920 switches are 1000Base-T links. The links between the two HP2920 switches are also 1000Base-T links, but the link speed is reduced to 100 Mbit/s in the configuration of the HP2920 to measure the quality of service properties. To support the simulation of a failover the two HP2920 switches are connected redundantly to each other. The switches run firmware release WB 15.14.0002 and support OpenFlow protocol version 1.0 and 1.3. For the tests, we use two different OpenFlow controllers. For the first scenario the OpenDayLight controller [25] and for the second scenario the HP controller [7] is used.

The measurement network and the configuration and control network are separated. This way the measurement itself is not influenced by the configuration and control traffic of the OpenFlow controller and the configuration traffic of the measurement tool.

All devices in the testbed are synchronized to a local GPS-equipped NTP server using the SNTP protocol [26].

In the first test scenario we generate two GOOSE-based traffic flows from client 1 to client 2. One GOOSE flow is prioritized. The overall data rate of a GOOSE flow is about 200 kbit/s. The network load in the network between client 1 and client 2 is very high as client 1 also generates an additional flow to client 2 with an overall data rate of about 440 Mbit/s. The second test scenario uses a single GOOSE traffic flow from client 1 to client 2. During the transmission of the GOOSE packets, the ethernet cable of the active link between the two switches is disconnected. The duration for both test scenarios is about 300 seconds.

For test case 1, simple packet forwarding based on MAC addresses was configured on both switches using OpenFlow. For test case 2, OpenFlow groups were used. A group consists of a list of action buckets and a group type that determines

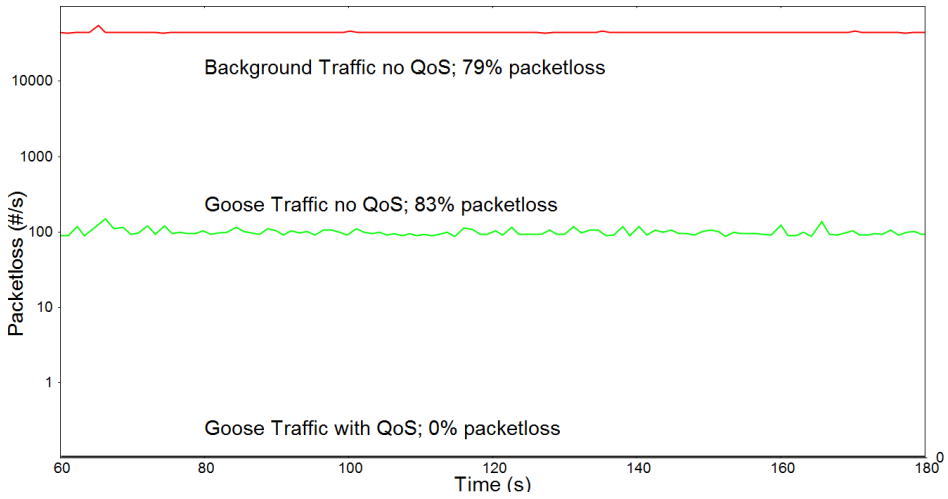

Fig. 3: Test scenario 1; Packet loss

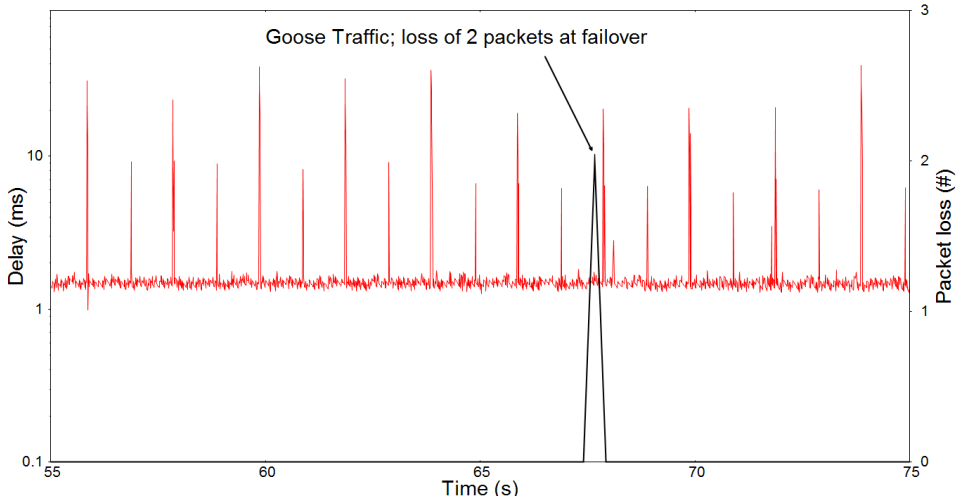

Fig. 4: Test scenario 2; Failover

which action buckets are executed for an incoming packet. In our test case the fast failover group type was used and incoming packets were processed by the first live action bucket. The liveliness of an action bucket can for example depend on the liveliness of a port as in our case. Fast failover takes places locally without communication to the controller to minimize delay.

V. Results

Figure 3 shows the packet loss per second of the generated flows for scenario 1. The prioritized GOOSE flow shows no packet loss. Its minimum delay during the measurement was 0.061 ms, the maximum delay for a few packets were 152 ms. The mean delay was calculated as 0.903 ms. The flows without prioritization had a packet loss of about 80% during the measurement. The result of test scenario 1 shows that an OpenFlow software-defined network can generally be used to transport high priority traffic but also that the delay of prioritized traffic can still vary significantly.

The result of the second test scenario is shown in Figure 4. The graph shows the latency of an emulated GOOSE traffic between client 1 and client 2. The overall mean delay is about 2 ms, this value is significantly higher than in the first test. These higher values are due to the implementation of the HP2920 switch. For the second scenario group tables with fast failover functionalities were used which are processed in software instead of in hardware on the switches. The peak values for the latency are in the range of 50-70 ms and occur about every second. This is probably also due to the processing of the flow in software on the switches and may be caused by regular operations triggered in the switch or by the controller.

VI. Conclusion

The use of OpenFlow-based software-defined networking in productive use in power systems cannot necessarily be recommended at this stage yet. There are still a few open issues that need to be solved, both from a specification point of view as shown through our analysis as well as regarding implementations as shown through our measurements.

On the other hand, SDN and OpenFlow offer important conceptual advantages over conventional networks and basic

functionalities such as traffic prioritization and failover behavior already work as intended. One conceptual advantage of using OpenFlow in an energy communication network as we experienced during our tests is for example the fact that flows can be entirely controlled and protections paths can be entirely defined in advance. Thus, in case of a failure, the system behaves in a well-defined and predictable way.

As part of our ongoing research project we will continue to follow the rapid development in OpenFlow and apply SDN to wide-area energy communication networks.

ACKNOWLEDGMENTS

The work described in this paper was part of the project 'Open Flow Secure Grid (OFSE-Grid)', which was funded by the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT).

REFERENCES

[1] A. Cahn, J. Hoyos, M. Hulse, and et al., "Software-defined energy communication networks: From substation automation to future smart grids," in *Smart Grid Communications (SmartGridComm), 2013 IEEE Int. Conf. on*, 2013, pp. 558–563.

[2] "Software-defined networking: The new norm for networks," White Paper, Open Networking Foundation, Apr. 2012.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[4] S. Jain, A. Kumar, S. Mandal, and et al., "B4: Experience with a globally-deployed software defined wan," in *Proc. of the ACM SIGCOMM 2013 Conf. on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 3–14.

[5] V. Yazici, M. Sunay, and A. Ercan, "Architecture for a distributed openflow controller," in *Signal Processing and Communications Applications Conf. (SIU), 2012 20th*, 2012, pp. 1–4.

[6] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proc. of the 2010 Internet Network Management Conf. on Research on Enterprise Networking*, ser. INM/WREN'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 3–3.

[7] *HP VAN SDN Controller Administrator Guide, Edition 1*, Hewlett-Packard, Nov. 2013.

[8] *Cisco Extensible Network Controller Deployment Guide, Release 1.0*, Cisco, Oct. 2013.

[9] R. Sherwood, G. Gibb, and K.-K. e. a. Yap, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, 2009.

[10] P. Porras, S. Shin, V. Yegneswaran, and et al., "A security enforcement kernel for openflow networks," in *Proc. of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 121–126.

[11] A. Khurshid, W. Zhou, M. Caesar, and et al., "Veriflow: Verifying network-wide invariants in real time," in *Proc. of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 49–54.

[12] R. Klöti, "Openflow: A security analysis," Master's thesis, Swiss Federal Institute of Technology, Zurich, 2013.

[13] Y. Yamasaki, Y. Miyamoto, and J. Yamato et al, "Flexible access management system for campus vlan based on openflow," in *Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th Int. Symposium on*, 2011, pp. 347–351.

[14] N. Handigol, R. Johari, and N. McKeown, "Aster*x: Load-balancing as a network primitive," *9th GENI Engineering Conf. (Plenary)*, 2010.

[15] S. Shin and G. Gu, "Cloudwatcher: Network security monitoring using openflow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?)," in *Network Protocols (ICNP), 2012 20th IEEE Int. Conf. on*, 2012, pp. 1–6.

[16] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: Transparent moving target defense using software defined networking," in *Proc. of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 127–132.

[17] G. Yao, J. Bi, and P. Xiao, "Source address validation solution with openflow/nox architecture," in *Network Protocols (ICNP), 2011 19th IEEE Int. Conf. on*, 2011, pp. 7–12.

[18] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *Local Computer Networks (LCN), 2010 IEEE 35th Conf. on*, 2010, pp. 408–415.

[19] R. Moore and M. Goraj, "Ethernet for iec61580," PAC World, Tech. Rep., September 2010. [Online]. Available: "http://www.pacw.org/no-cache/issue/september2010issue/coverstory/ethernetforiec61850/completearticle/1/print.html"

[20] N. Yadav and E. Kapadia, "Ip and ethernet communication," Grid Interop, Tech. Rep., 2010.

[21] *Communication networks and systems in substations- Part 5*, International Electrotechnical Commission (IEC) Std., 2003.

[22] *Communication networks and systems in substations- Part 1*, International Electrotechnical Commission (IEC) Std., 2003.

[23] T. Rieder and W. Schaffer, "Smart grids modellregion salzburg: Zentrale spannungs- (u) und blindleistungsregelung (q) mit dezentralen einspeisungen in der demoregion salzburg," Salzburg AG, Neue Energien 2020, Tech. Rep., April 2012.

[24] T. Fichtel and T. Pfeiffenberger, "CMT II: An agent based framework for comprehensive ip measurements," in *Tridentcom 2006, Barcelona*, 2006.

[25] (2014, Jan.) Opendaylight openflow controller. [Online]. Available: http://www.opendaylight.org

[26] D. Mills, U. Delaware, J. Martin, and et al., "Network time protocol version 4," RFC5905, IETF, June 2010.