Published in: 1st IEEE 5G World Forum (5GWF 2018)
DOI: 10.1109/5GWF.2018.8516975

Control Plane Fault Tolerance for Resilient Software-Defined Networking based Critical Infrastructure Communications

Fabian Kurtz, Dennis Overbeck, Caner Bektas, Christian Wietfeld
Communication Networks Institute, TU Dortmund University, Otto-Hahn-Strasse 6, 44227 Dortmund
Email: {fabian.kurtz, dennis.overbeck, caner.bektas, christian.wietfeld}@tu-dortmund.de

***Abstract*—Modern societies depend increasingly on Critical Infrastructures (CIs) such as Smart Grids (SGs) or Intelligent Transportation Systems (ITSs). These in turn rely on complex monitoring and control functionalities, which themselves require capable, flexible and robust communication infrastructures. As dedicated networks and computing resources are associated with high costs and time-consuming deployment, the upcoming fifth generation of mobile communication (5G) aims to enable cloud-based shared infrastructures via Network Function Virtualization (NFV) and Software-Defined Networking (SDN). While NFV separates hardware and logical functionalities, SDN abstracts physical data packet forwarding from programmable network control tasks such as routing. Thereby so called SDN controllers are created, which simplify the integration of heterogeneous technologies and enable the flexible addition of new features. Yet, due to the controllers' centralized nature a potential single-point-of-failure is created. Thus we present a heartbeat-based approach to SDN resilience, utilizing redundant controllers to address CI communication requirements. An empirical evaluation, on the example of particularly demanding SGs traffic, illustrates reduced end-to-end failover delays, i.e. the duration cloud-driven 5G networks cannot process requests or changes.**

I. Introduction

To improve quality of life and use resources efficiently, society increasingly relies on CIs such as SGs. Stable operation of these systems necessitates capable, flexible and highly reliable cloud platforms and communication networks. In this context SDN has emerged as a solution for managing Information and Communication Technology (ICT). Contrary to traditional networks, SDN separates all control functions, i.e. the Control Plane (CP) handling e.g. routing or prioritization, from physical packet forwarding. Thereby a Data Plane (DP) is created, in which routers and switches purely forward traffic according to the rules stored in their forwarding tables. A centralized SDN controller software configures theses tables via the Southbound Application Programming Interface (API), most commonly utilizing the OpenFlow (OF) [1] protocol. If switches encounter a packet for which no table entry exists, e.g. if new traffic flows enter the network, an OFPacketIn message is sent to the controller. There a path appropriate for the flow is calculated and installed in all relevant switches via an OFFlowMod message. Moreover, as shown by Figure 1, an application plane is created. Here functionalities of the controller, as well as those of services using the network, exist. Thus new algorithms can be deployed independently of hardware upgrades. Applications utilize the so called Northbound API to directly transmit their requirements to the SDN controller. Thereby the communication infrastructure is automatically configured to meet application demands. Yet, by centralizing control SDN creates a potential single-point-of-failure, threatening network availability and reliability. This also applies if the controller is realized as a NFV-based Virtual Network Function (VNF) with a millisecond instantiation time. Failures of the underlying hardware require the same control plane recovery measures as without virtualization. If the controller fails, existing DP flows cannot be modified while new ones cannot be established. Although active flows are not affected, the impact on CIs is severe as hard service level guarantees can no longer be enforced. Yet Wide Area Monitoring Protection and Control (WAMPAC) in SGs continuously needs to reconfigure the DP via the CP to successfully transmit grid control messages for stabilizing the power grid. Consequently CP failures endanger the entire CI's, respectively cloud's availability. While SDN's East-/Westbound APIs are designed for interactions between multiple controllers, work in this regard predominantly focuses on scalability. In contrast we utilize this API to design and evaluate a controller failover strategy on the example of SG requirements, which are among the most challenging in terms of network recovery latency.

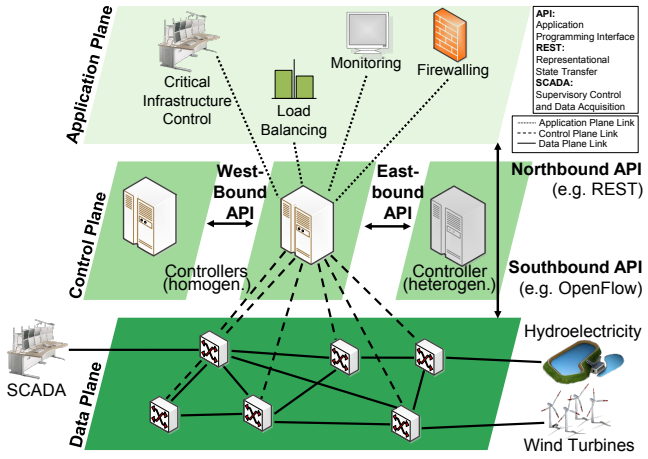

Figure 1: The Architecture of Software-Defined Networking

978-1-5386-4982-4/18/$31.00 © 2018 IEEE

© 2018 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, including reprinting/republishing this material for advertising or promotional purposes, collecting new collected works for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

II. Related Work

Originally developed for deployment in cloud datacenters, SDN has become a key component of 5G networks, enabling efficient management of all kinds of communication infrastructures. While various works dealing with resiliency of SDN based networks are available, most focus on the DP, such as [2], [3] or [4]. Papers which address CP robustness, i.e. the SDN Controller, predominantly cover performance metrics such as flow setup rates after failover [5]. However, the delay incurred by CP failure mitigation strategies is rarely studied and does not meet the demands of CI communication. Four main approaches to CP resilience are discussed in literature, as depicted by Figure 2. Voting algorithms can also be used for ensuring CP redundancy. A set of controllers determines the correct control decision either on a case-by-case basis or elects a new primary in case the original fails. While robust against even byzantine failures (i.e. if a devices sends differing results to its peers), performance is reduced due to the complex consensus finding process, c.f. [6] and [7]. A variation of such voting systems is presented in [8], where redundant brokers in the CP collect messages from all controllers and act only on commands issued by a majority. In contrast, the strategy refined in this work builds on a primary/secondary controller cluster. Here at least two interconnected controllers (one in hot-standby) run in parallel, perpetually synchronizing states between internal databases. In case a device fails, the other takes over, keeping the network responsive to changes. Recovery delays observed in [9] and [10] are on the order of seconds and thus too slow for use in CI. Common Address Redundancy Protocol (CARP) [11] and its proprietary equal Virtual Router Redundancy Protocol (VRRP) [12], as used by e.g. Openstack Neutron [13], also utilize the primary/secondary paradigm. A virtual, shared Internet Protocol (IP) address enables failover between devices typically within seconds [14]. [15] presents a variation of this scheme which utilizes SDN to improve the duration of CP recovery. The primary/secondary approach can also employ external databases, shared among controllers. This trades synchronization for database replication overhead, as shown by [16], [17] and [18].

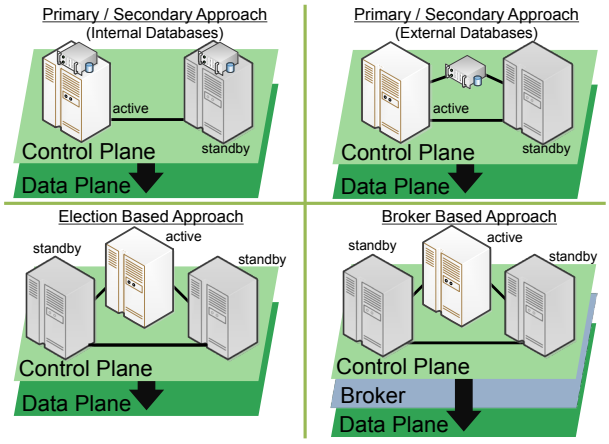

Figure 2: Overview of SDN Controller Resilience Strategies

III. Software-Defined Networking for Critical Infrastructures

This section introduces the demands of SG communications and establishes our SDN controller resiliency concept for CIs.

A. Smart Grid Requirements

IEC 61850 [19], standardized by the International Electrotechnical Commission (IEC), is a key SG communication protocol. As shown by Figure 3, it provides three types of messages. Settings are handled via Manufacturing Message Specification (MMS) packets, while event notifications and actions such as busbar switching are performed via Generic Object Oriented Substation Events (GOOSE). Measurements are sent via Sampled Values (SV) messages, e.g. to detect short circuits for power line protection. Due to their importance for SG stability, SV and GOOSE specify a maximum end-to-end delay of 10 to 20ms (Type 1A-P1 and 1B-P2/3), regardless of failures in the communication network. Hence, fault detection, switching to a backup SDN controller and finally fault recovery have to be completed below these thresholds.

B. Concept & Implementation of the Resilient SDN Controller

To meet the requirements outlined, the primary-secondary controller resilience strategy is refined. Failure detection is realized via a heartbeat, e.g. controllers notify each other in fixed intervals of their availability. If either fails, the heartbeat or its corresponding acknowledgements are missing. False positives, i.e. switching to a new primary without failure event, are precluded via a detection multiplier. Hence a timeout is triggered only after multiple lost heartbeat packets. Heartbeats are sent via a dedicated link between controllers, avoiding false positives due to high delays (e.g. caused by network flooding). Furthermore, route calculation and failure detection processes are decoupled and run in parallel to reduce recovery delay. The heartbeat is sent via the control plane switch, as any inter-controller link failures would otherwise result in two active primaries, issuing ambiguous commands. Transitioning delay from secondary to primary mode is dependent on the backlog of unanswered OFPacketIn messages accumulated during failover and verifying controller state against DP forwarding tables. Further potential inconsistencies are avoided by continuous synchronization. Such issues occur e.g. in case the primary fails after it sent a OFFlowMod to the DP, without updating the secondary, causing it to send potentially conflicting instructions. Synchronization also allows replacing failed devices without downtime, improving system availability.

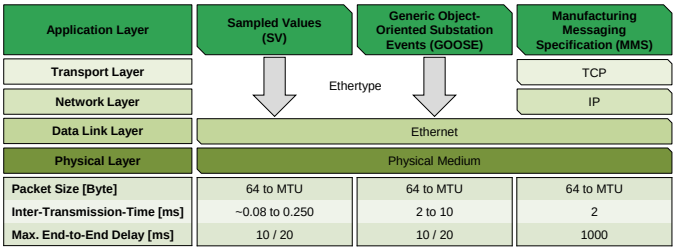

Figure 3: IEC 61850 - Message Types and Requirements

IV. Testing Setup and Evaluation Scenario

The testing setup consists of the following components: Eight identical servers, based on Intel Xeon D-1518 Central Processing Units (CPUs) (four cores at 2,2GHz) with 16GB RAM and six 1GBase-T Ethernet Ports (two by Intel I210, four by Intel I350). Ubuntu 16.04.3 LTS (v4.13.0-32-generic x86-64 Kernel) is used as Operating System (OS) on all devices. Figure 4 depicts the evaluation setup and scenario, as well as the chain of events in case of a controller failure. Two devices are used as hosts, exchanging traffic over the DP. This network is formed by four computers configured as virtual switches by running Open vSwitch (v2.8.2). To avoid any interference of measurements on performance two additional networks are available. The CP distributes the controller's commands to the DP, via a dedicated switch (Zyxel GS1900-24E). An additional maintenance network serves to control experiments and collect measurements. A fork of the open source Floodlight [20] Java project, called Software-defined Universal Controller for Communications in Essential SystemS (SUCCESS) [4], is used as primary and secondary SDN controller. Developed by the Communications Networks Institute (CNI) it specifically targets CI communications and implements the resiliency approach under study in this work. To replicate the conditions of the SG wide area protection use case, Host A sends IEC 61850 GOOSE messages to Host B with an Inter-Transmission Time (ITT) of $10ms$. For evaluation of the resiliency approach it is necessary to trigger an event which involves controller interaction. Therefore each packet of the traffic flow is created to not match any entry in the switches' forwarding tables. Hence, an OFPacketIn is sent to the controller, which then calculates and installs an appropriate path. All devices of the testing setup are synchronized via Precision Time Procotol (PTP) [21] with a maximum clock deviation of $152\mu s$, facilitating a precise measurement of the sequence of events during failover. Controller failures are induced by interrupting all its processes on the corresponding server. Measurements are based on at least 1000 runs to achieve sufficient confidence in results.

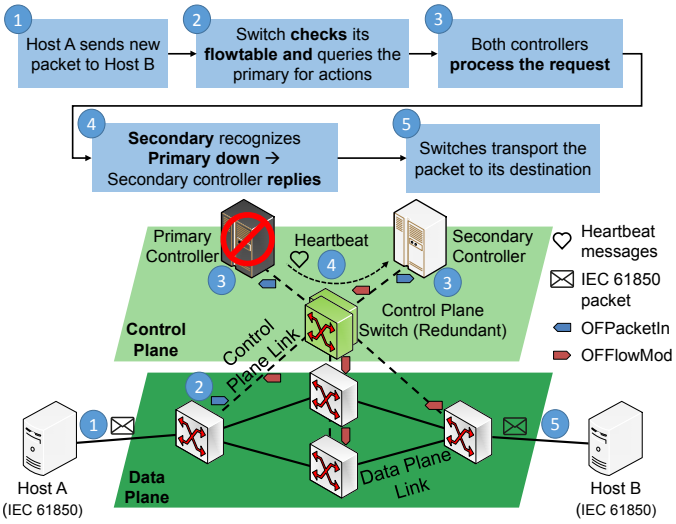

Figure 4: Evaluation Testing Setup and Scenario

V. Evaluation Results

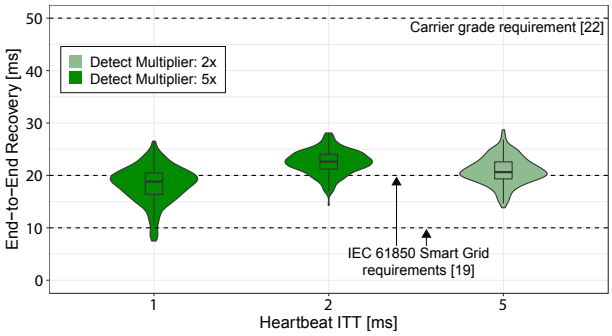

Figure 5: End-to-End Recovery for Various Heartbeat ITT

Figure 5 depicts the end-to-end recovery observed for heartbeats with an ITT of 1, 2 and $5ms$. As new flows enter the DP in $10ms$ intervals, high controller load causes perceivable jitter of heartbeat messages and thus false positives. Hence, detect multipliers of two and five are used, respectively. Median recovery delays range from 18 to $22ms$, with an ITT of $1ms$ (i.e. $5ms$ timeout) being the fastest and $2ms$ the slowest. While having the same detection timeout of $10ms$, heartbeats with $5ms$ ITT are faster in this case than those with $2ms$ as controller load is reduced by having to handle 2.5 times fewer messages. This observation applies also to $1ms$ ITT, which loses most of its theoretical advantage in recovery delay to computational overhead. Also, garbage collection of the Java programming language, as used by our controllers, randomly adds delay spikes. In all cases end-to-end recovery delays stay below carrier grade requirements [22] with $28ms$ maximum. Demands of IEC 61850 are met in the median, respectively with low outliers, at $1ms$ heartbeat ITT.

The composition of the observed end-to-end recovery delays for $1ms$ ITT are shown by Figure 6. Calculating the path for a new traffic flow takes about $3ms$, with few $\sim 7ms$ peaks. Failure detection is achieved in a mean of $9ms$, with few instances down to $\sim 4ms$ and up to $23ms$. Transitioning from failed to redundant controller requires a mean of $10ms$, with a minimum of $\sim 4ms$ and maximal $\sim 15ms$. Total end-to-end recovery delay depends on the failure's timing, influencing

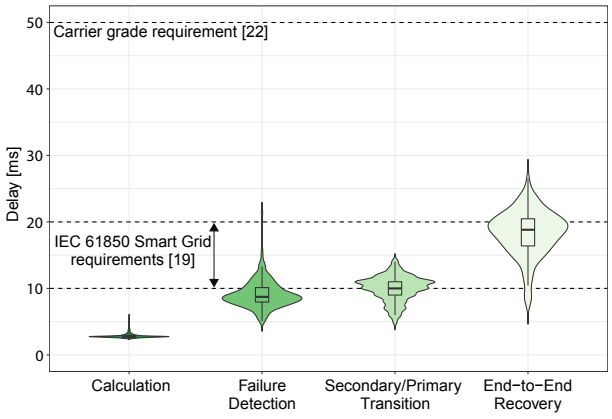

Figure 6: Recovery Delay Composition for $1ms$ Heartbeats

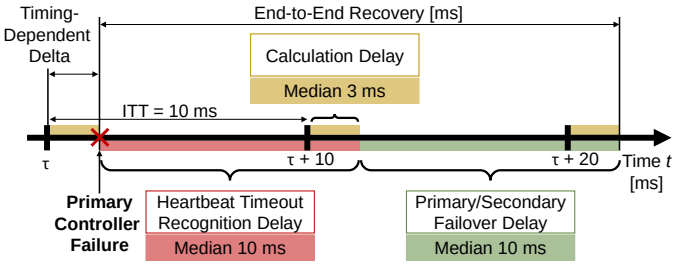

Figure 7: End-to-End Recovery Delay Sequence

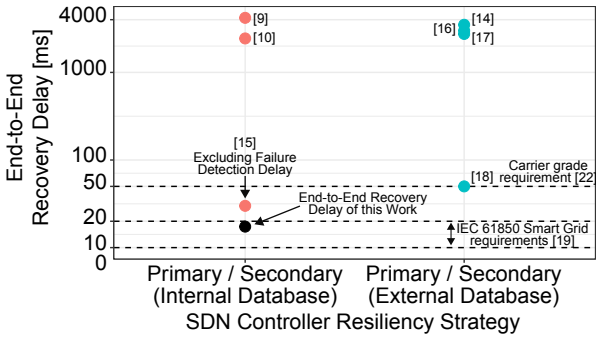

Figure 8: Results in Comparison to Related Works

which of the tasks running in parallel finishes last. Figure 7 illustrates this. If the primary fails directly after receiving an OFPacketIn, detection occurs $\sim 1ms$ after calculation, recovering another $4ms$ later, i.e. $\sim 8ms$ after failure at the fastest. Recovery takes longer, should the primary fail shortly ahead of sending its reply. With the fastest outliers for detection and transition ($\sim 4ms$) the secondary is ready $1ms$ after the next OFPacketIn. For recovery to complete, this also has to be processed, adding $2ms$ of remaining calculation. Hence, recovery beginning from the original DP request takes $\sim 13ms$. Results are put into context of related work by Figure 8. As can be seen, the achieved SDN DP recovery delay improves upon the current state of the art, with only [15] reaching below $50ms$ (but not including failure detection).

VI. Conclusion and Outlook

In this work a heartbeat based approach to SDN controller failover is refined and evaluated against the requirements of CI communications. Network states are continuously synchronized across two redundant controllers, with concurrent failure detection and control decision calculation. Thus failure recovery delays are reduced, as correct solutions are already prepared and only need to be sent to the DP. While, e.g. due to inefficiencies of the Java programming language, the strictest class of $10ms$ end-to-end delays imposed by the SG protocol IEC 61850 are not always met, areas for optimization are identified. Carrier grade requirements are fulfilled, facilitating the solution's deployment in cloud infrastructures. Future work will focus on scaling the approach to larger scenarios and further performance improvements. Additionally, the impact of deploying the robust controllers as cloud VNFs will be analyzed and a comparison with the concept of [8] is targeted.

Acknowledgement

This work has been carried out in the course of research unit 1511 *'Protection and control systems for reliable and secure operations of electrical transmission systems'*, funded by the German Research Foundation (DFG) and the Franco-German Project *BERCOM* (FKZ: 13N13741) co-funded by the German Federal Ministry of Education and Research (BMBF).

References

[1] *OpenFlow Switch Specification Version 1.5.1*, Open Networking Foundation, 2015. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf.

[2] N. van Adrichem, B. Van Asten and F. Kuipers, 'Fast Recovery in Software-Defined Networks', in *European Workshop on Software Defined Networks (EWSDN)*, Sep. 2014, pp. 61–66.

[3] A. Aydeger *et al.*, 'Software Defined Networking for Resilient Communications in Smart Grid Active Distribution Networks', in *IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–6.

[4] N. Dorsch, F. Kurtz, F. Girke and C. Wietfeld, 'Enhanced Fast Failover for Software-Defined Smart Grid Communication Networks', in *IEEE Global Comm. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.

[5] F. Botelho *et al.*, 'Design and Implementation of a Consistent Data Store for a Distributed SDN Control Plane', in *European Dependable Computing Conference (EDCC)*, Sep. 2016, pp. 169–180.

[6] K. ElDefrawy and T. Kaczmarek, 'Byzantine Fault Tolerant Software-Defined Networking (SDN) Controllers', in *IEEE Computer Software and Applications Conf. (COMPSAC)*, vol. 2, Jun. 2016, pp. 208–213.

[7] H. Li, P. Li, S. Guo and S. Yu, 'Byzantine-Resilient Secure Software-Defined Networks with Multiple Controllers', in *IEEE International Conference on Communications (ICC)*, Jun. 2014, pp. 695–700.

[8] F. Kurtz and C. Wietfeld, 'Advanced controller resiliency in software-defined networking enabled critical infrastructure communications', in *International Conference on Information and Communication Technology Convergence (ICTC)*, Oct. 2017, pp. 673–678.

[9] M. A. S. Santos *et al.*, 'Decentralizing SDN's Control Plane', in *IEEE Conference on Local Computer Networks*, Sep. 2014, pp. 402–405.

[10] K. C. Fang, K. Wang and J. H. Wang, 'A Fast and Load-Aware Controller Failover Mechanism for Software-Defined Networks', in *International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, Jul. 2016, pp. 1–6.

[11] *Common Address Redundancy Protocol*, [Online]. Available: https://marc.info/?l=openbsd-misc&m=106642790513590, OpenBSD, 2003.

[12] S. Nadas, 'Virtual Router Redundancy Protocol Version 3 for IPv4 and IPv6', Internet Engineering Task Force, RFC 5798, Mar. 2010.

[13] *Openstack*, OpenStack Foundation, 2018. [Online]. Available: https://www.openstack.org/.

[14] L. Sidki, Y. Ben-Shimol and A. Sadovski, 'Fault Tolerant Mechanisms for SDN Controllers', in *IEEE Conference on Network Function Virtualization and Software Defined Networks*, Nov. 2016, pp. 173–178.

[15] S. Yoon *et al.*, 'Fast Controller Switching for Fault-Tolerant Cyber-Physical Systems on Software-Defined Networks', in *IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*, Jan. 2017, pp. 211–212.

[16] Y.-C. Chan, K. Wang and Y.-H. Hsu, 'Fast Controller Failover for Multi-domain Software-Defined Networks', in *European Conference on Networks and Com. (EuCNC)*, Jun. 2015, pp. 370–374.

[17] D. Suh *et al.*, 'Toward Highly Available and Scalable Software Defined Networks for Service Providers', *IEEE Communications Magazine*, vol. 55, no. 4, pp. 100–107, Apr. 2017.

[18] V. Pashkov, A. Shalimov and R. Smeliansky, 'Controller Failover for SDN Enterprise Networks', in *International Science and Technology Conf. (Modern Networking Technologies)*, Oct. 2014, pp. 1–6.

[19] *IEC 61850: Communication Networks and Systems for Power Utility Automation*, International Electrotechnical Commission TC57.

[20] *Floodlight Controller Version 1.0*, Project Floodlight, 2015. [Online]. Available: http://www.projectfloodlight.org/floodlight/.

[21] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems 1588–2002*, The Institute of Electrical and Electronics Engineers, Inc.

[22] B. Niven-Jenkins *et al.*, 'Requirements of an MPLS Transport Profile (RFC 5654)', Tech. Rep., Sep. 2009.