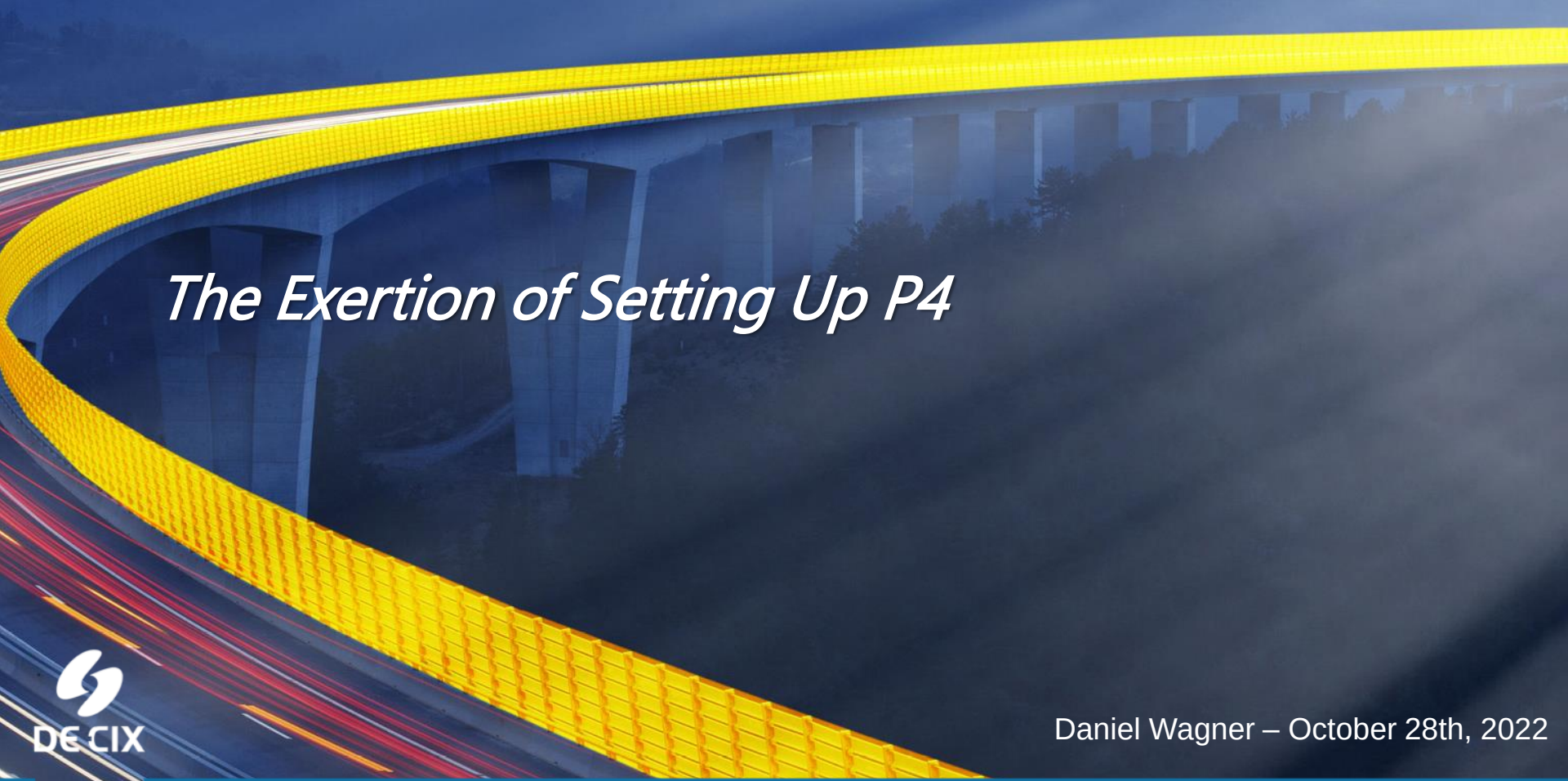


The Exertion of Setting Up P4

Daniel Wagner – October 28th, 2022

Introduction

- This experience-talk aims to
  - Recap what it took to set up P4 HW/SW
  - Help you deploy with less trouble

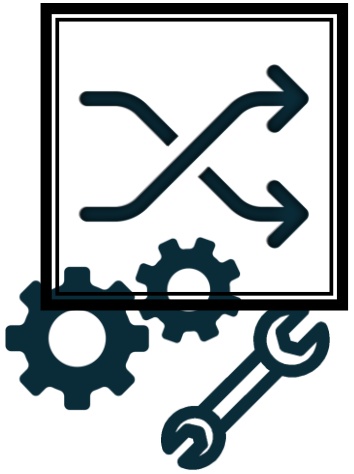
Introduction

- This experience-talk aims to
  - Recap what it took to set up P4 HW/SW
  - Help you deploy with less trouble

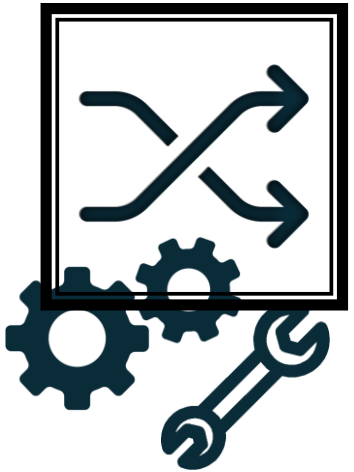
- This talk doesn‘t aim to
  - Explain what P4 does and how it works
  - Show off P4 perfomance evaluations
  - Provide you with P4 code (please see: p4.org & github.com/p4lang)
  - Bash the hardware vendor or P4 itself

Motivation

- ➔ Data plane programmability
- ➔ Develop custom protocols / solutions
- ➔ DPI / DDoS scrubbing / ARP sponge / …
- ➔ You-name-it!



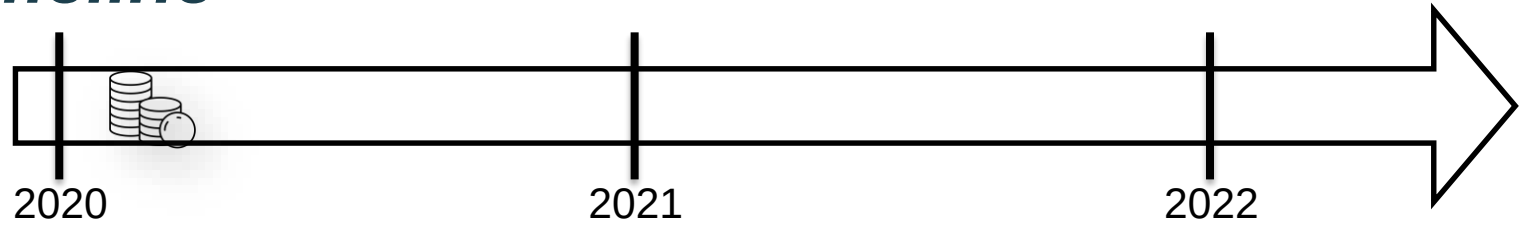
Motivation



- ➔ Data plane programmability
- ➔ Develop custom protocols / solutions
- ➔ DPI / DDoS scrubbing / ARP sponge …
- ➔ You-name-it!

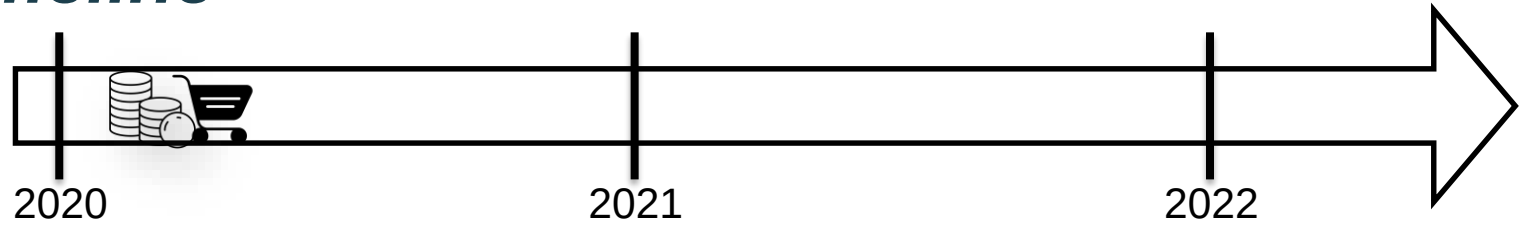
- ➔ What do I need to get started?
  …. Let‘s find out!

Timeline



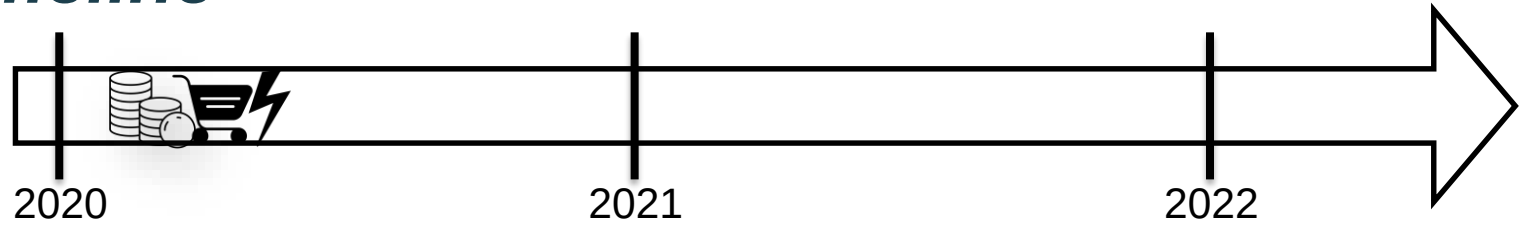
➔ February 2020: Purchase of P4 device(s)

Timeline



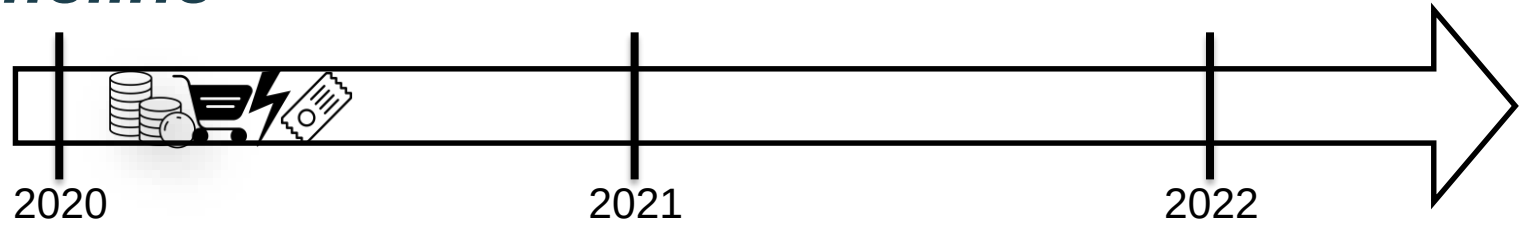
- February 2020: Purchase of P4 device(s)
- March 2020: Delivery of goods

Timeline



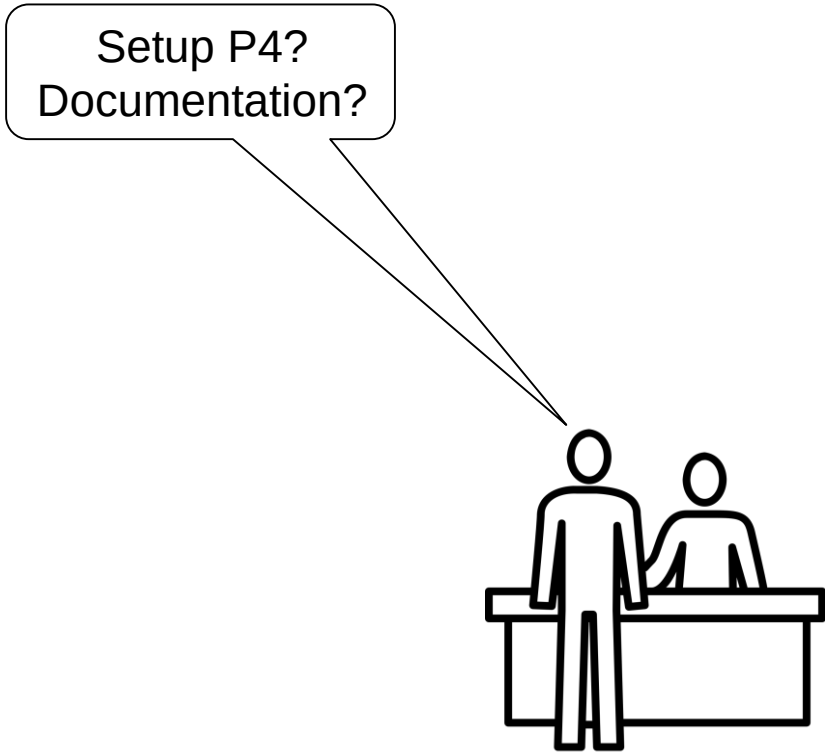
- February 2020: Purchase of P4 device(s)
- March 2020: Delivery of goods
- June 2020: Power-up & first hands-on

Timeline

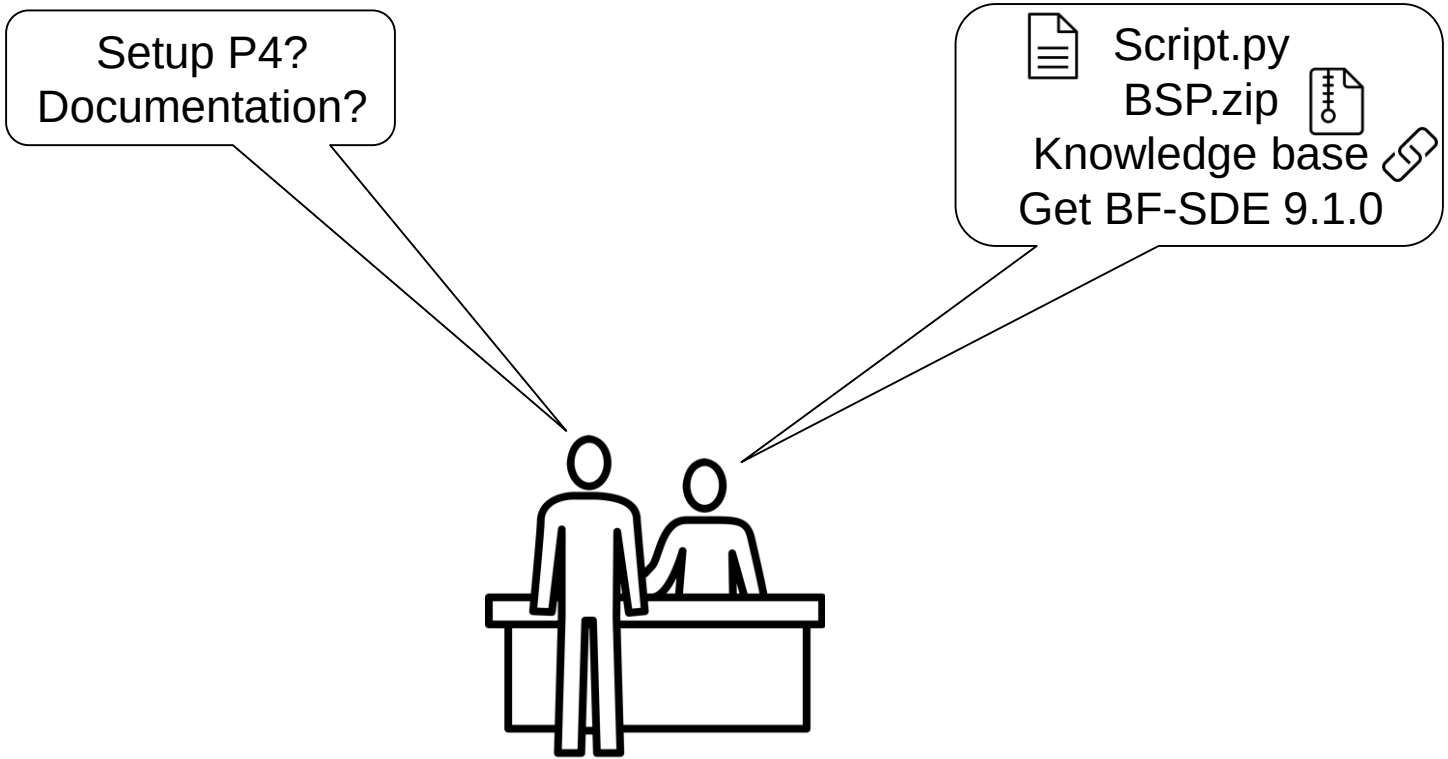


- ➔ February 2020: Purchase of P4 device(s)
- ➔ March 2020: Delivery of goods
- ➔ June 2020: Power-up & first hands-on
- ➔ July 2020: Registration at vendor‘s help desk & first support ticket

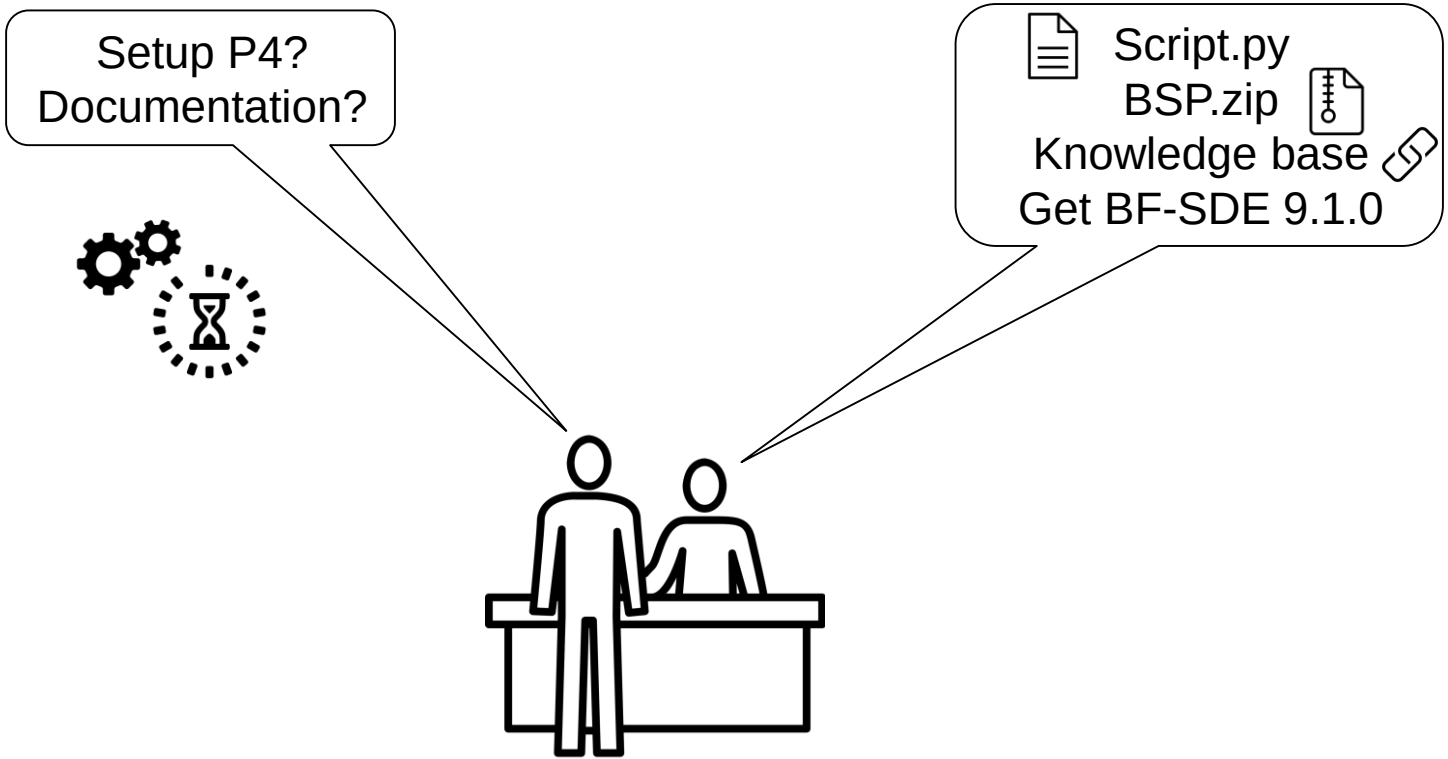
First Help Desk Ticket



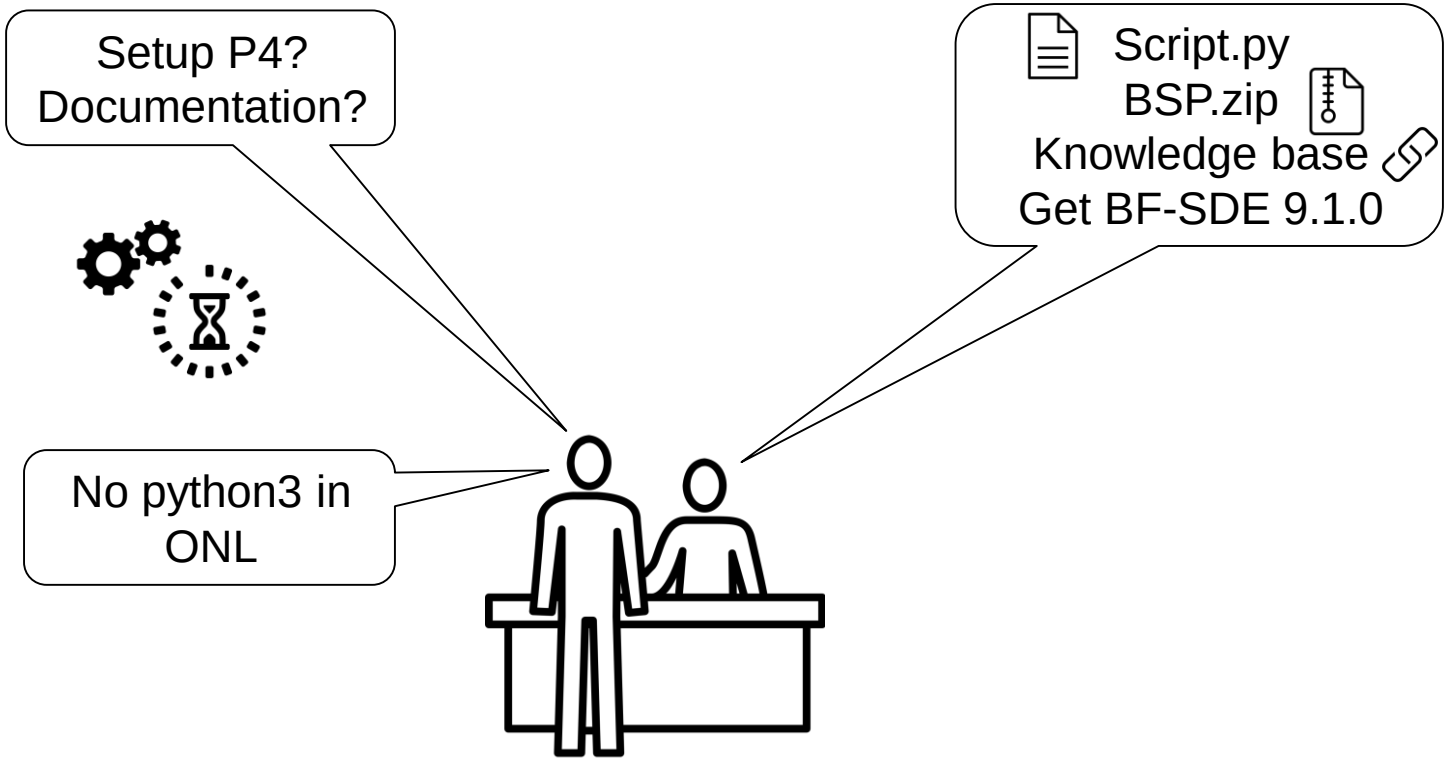
First Help Desk Ticket



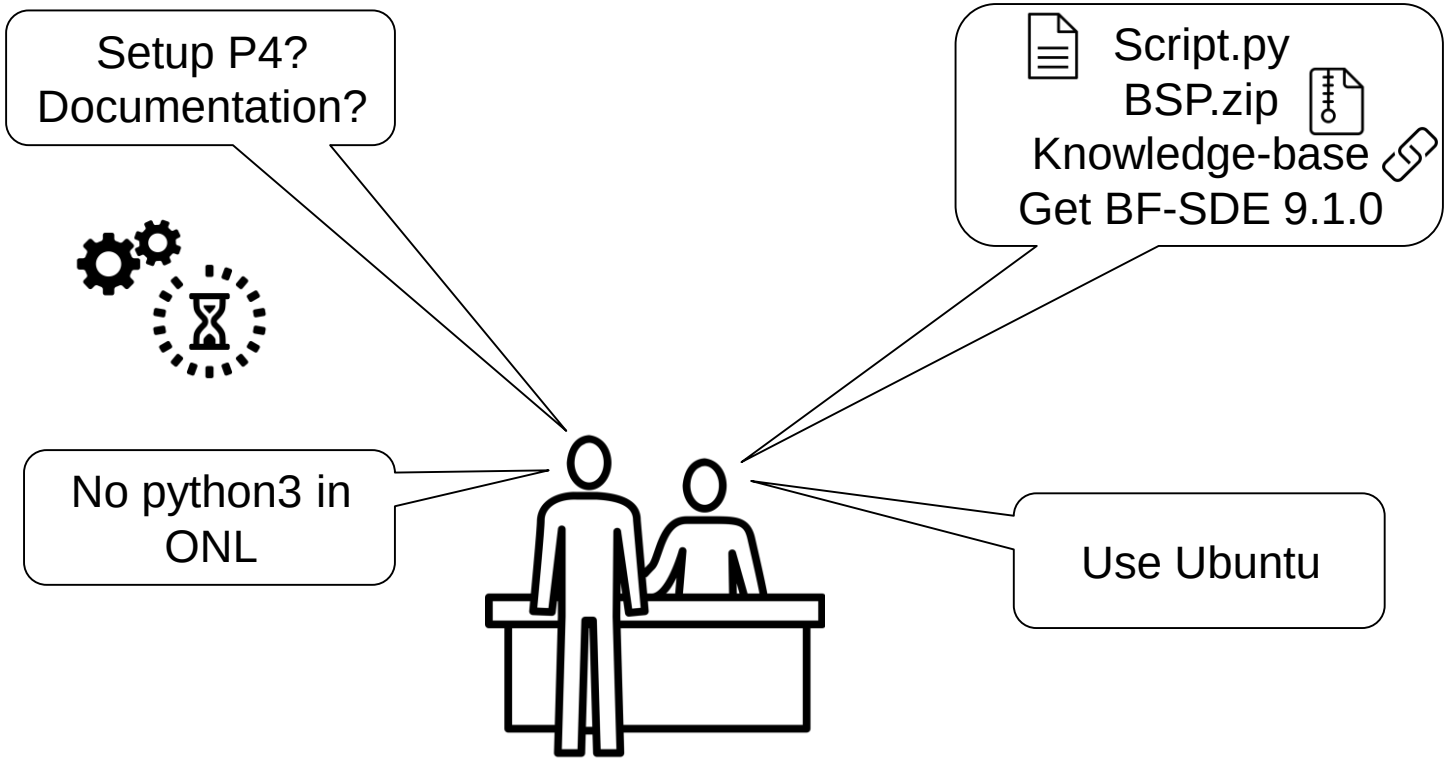
First Help Desk Ticket



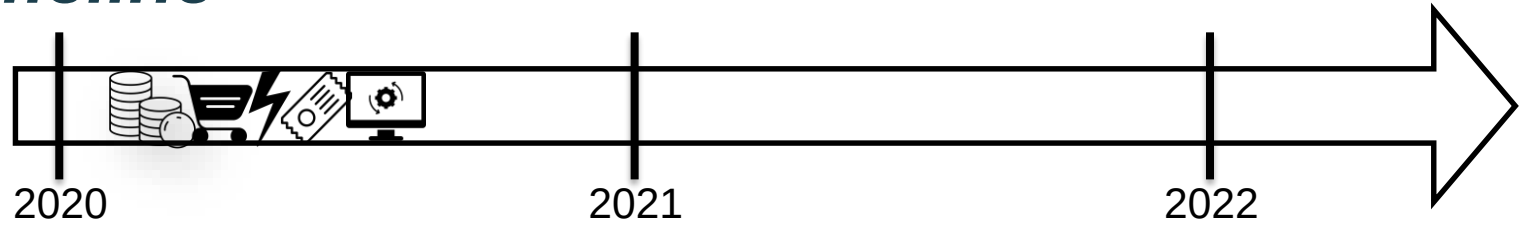
First Help Desk Ticket



First Help Desk Ticket

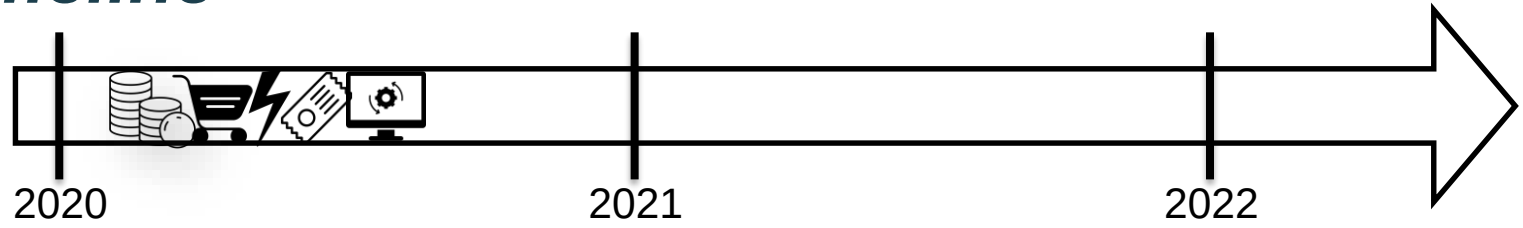


Timeline



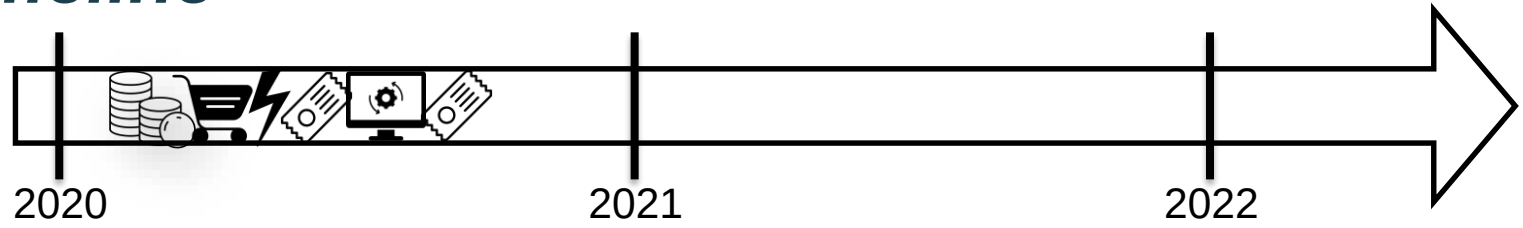
- ➔ July 2020: Acquisition of BSP and installer scripts
- ➔ August 2020: Checking for OS compatiblity infos, none found
- ➔ September 2020: Successfully installed Ubuntu Server 18.10
  Found errors in knowledge-base

Timeline



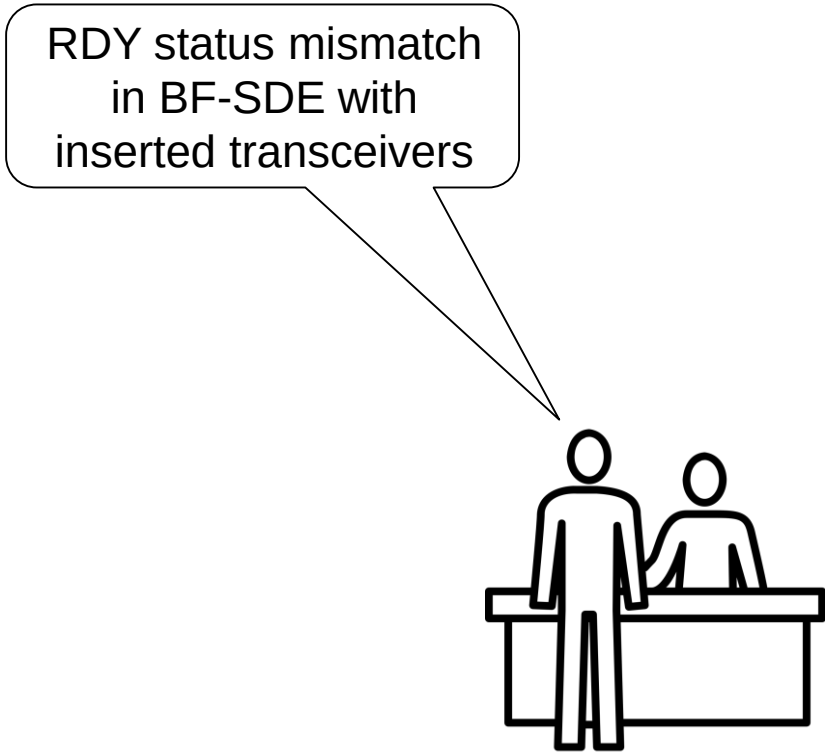
- ➔ July 2020: Acquisition of BSP and installer scripts
- ➔ August 2020: Checking for OS compatiblity infos, none found
- ➔ September 2020: Successfully installed Ubuntu Server 18.10
  Found errors in knowledge-base
  Successfully (?) installed BF-SDE 9.1.0

Timeline

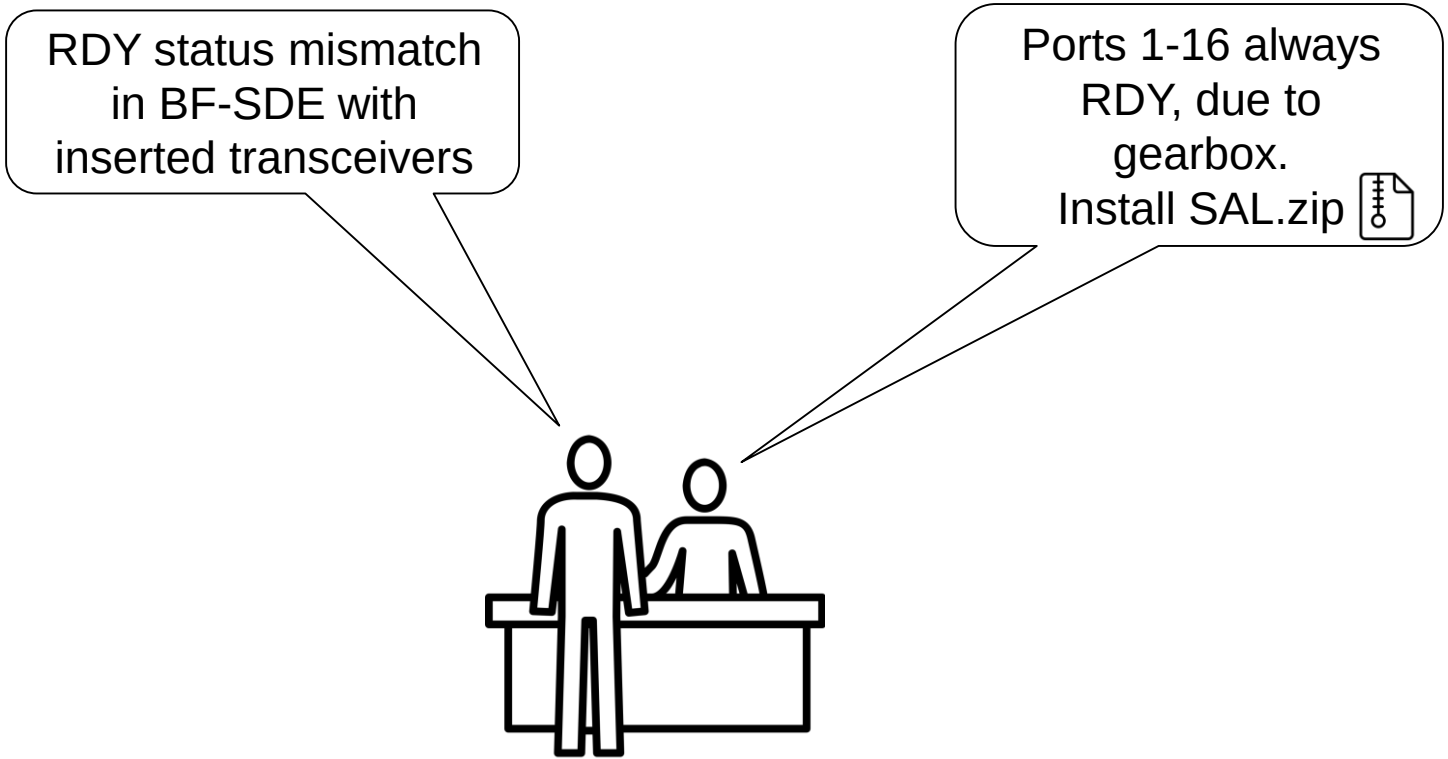


- ➔ July 2020: Acquisition of BSP and installer scripts
- ➔ August 2020: Checking for OS compatiblity infos, none found
- ➔ September 2020: Successfully installed Ubuntu Server 18.10
  Found errors in knowledge-base
  Successfully (?) installed BF-SDE 9.1.0
- ➔ October 2020: Second support ticket

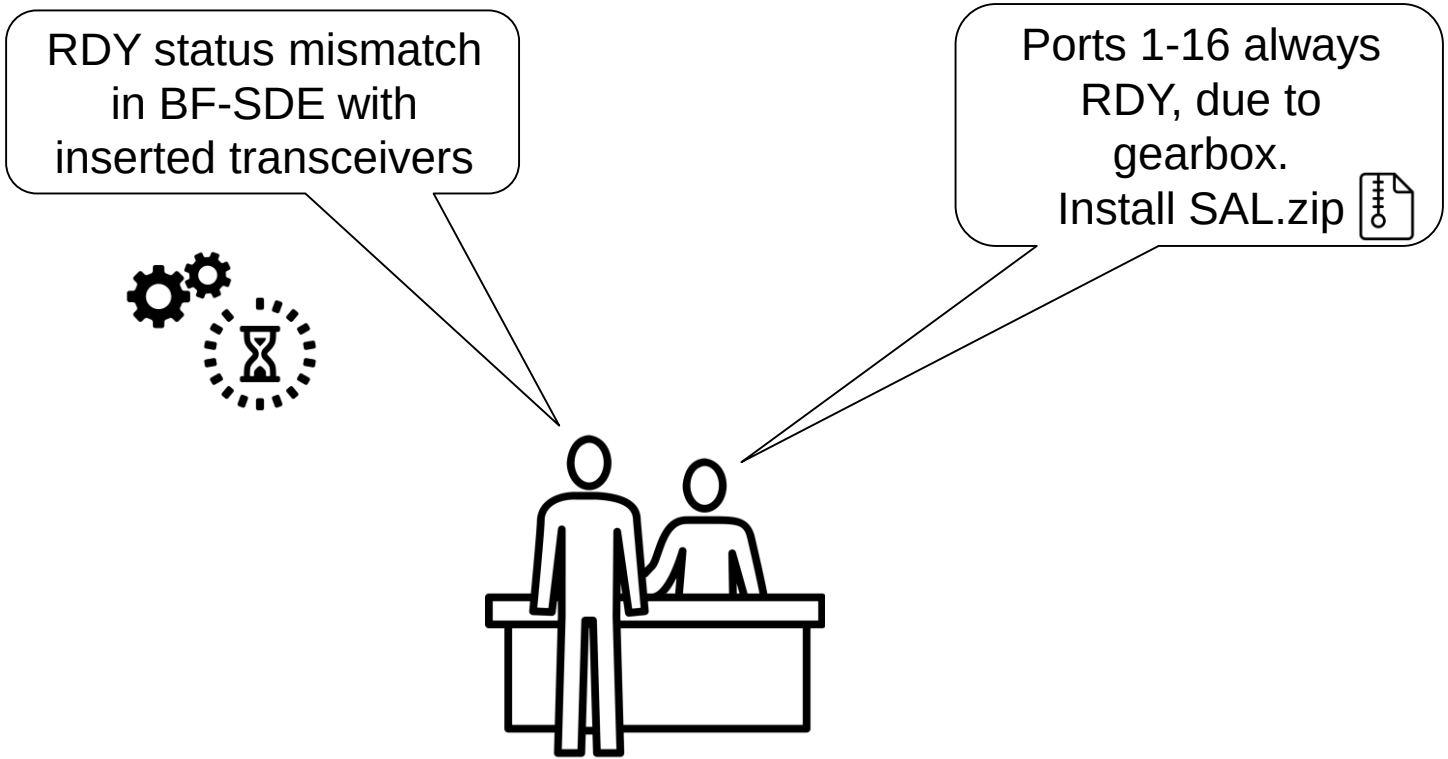
Second Help Desk Ticket



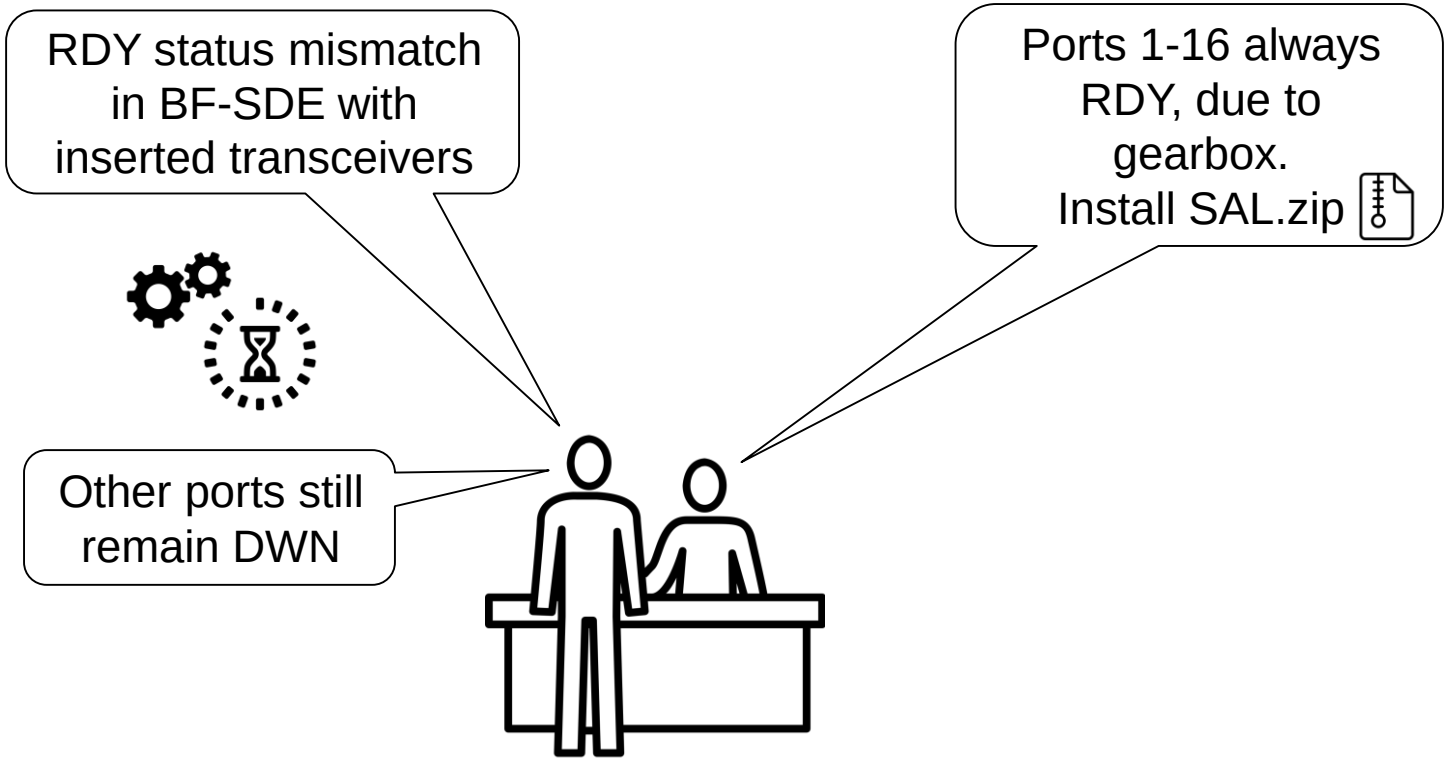
Second Help Desk Ticket



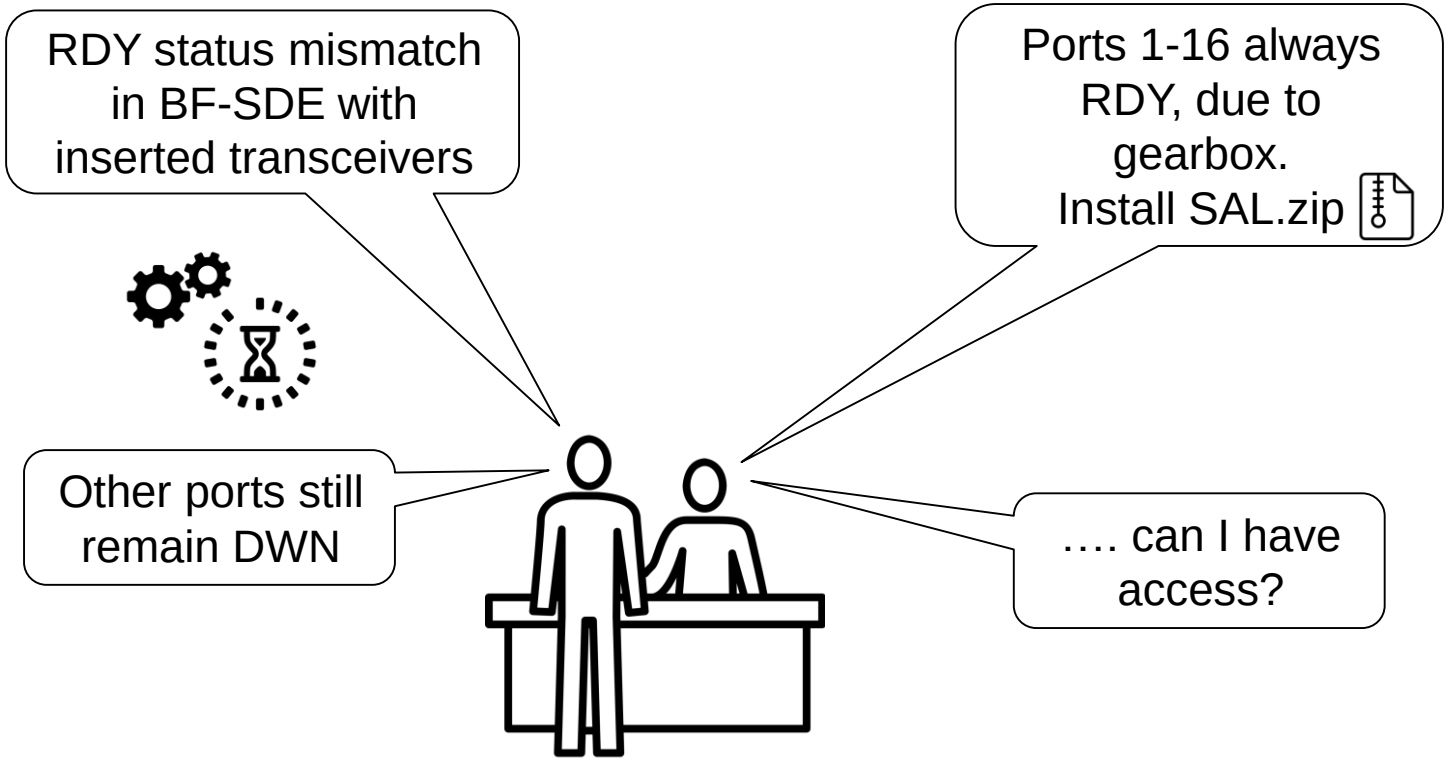
Second Help Desk Ticket



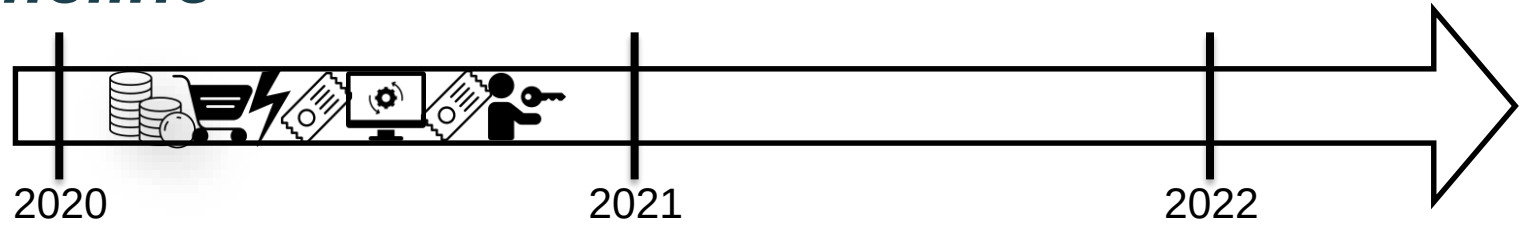
Second Help Desk Ticket



Second Help Desk Ticket



Timeline

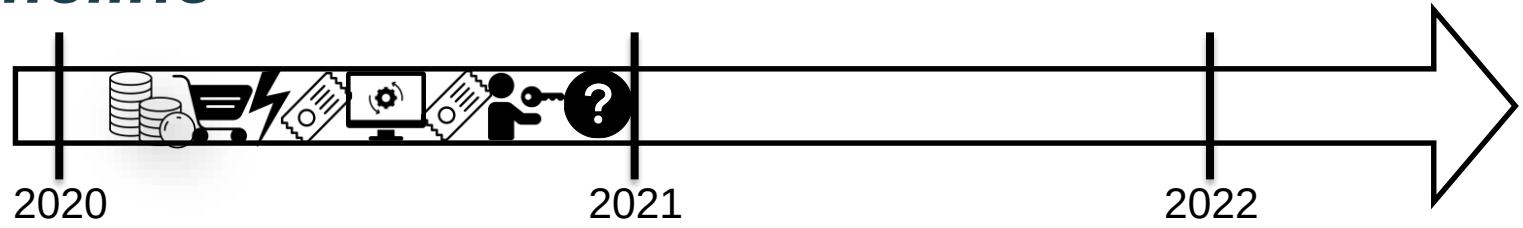


➔ November 2020: Switch access for help desk

Vendor enters M&A, help desk irresponsive, Jira offline

Ticket never closed / resolved

Timeline



➔ November 2020: Switch access for help desk

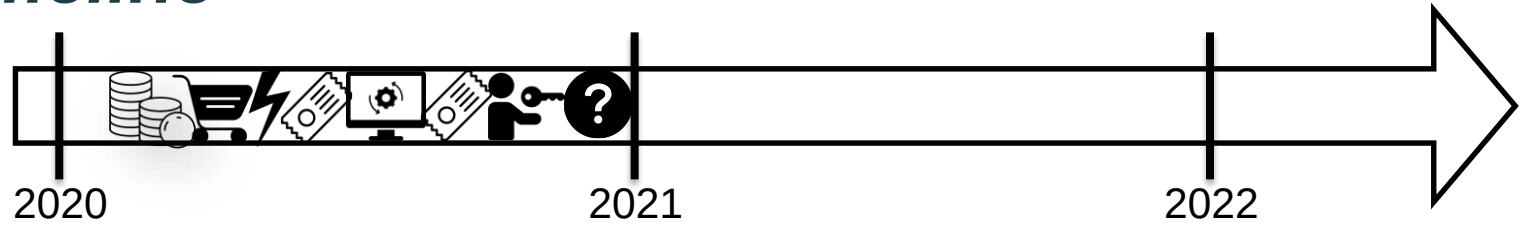
Vendor enters M&A, help desk irresponsive, Jira offline

Ticket never closed / resolved

➔ November 2020: Asking friends with same switch for help

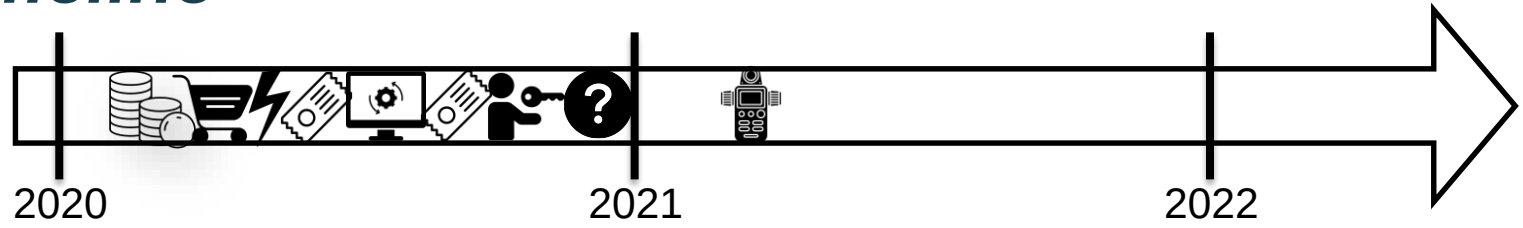
→ Have the exact same issues

Timeline



➔ …the days passed

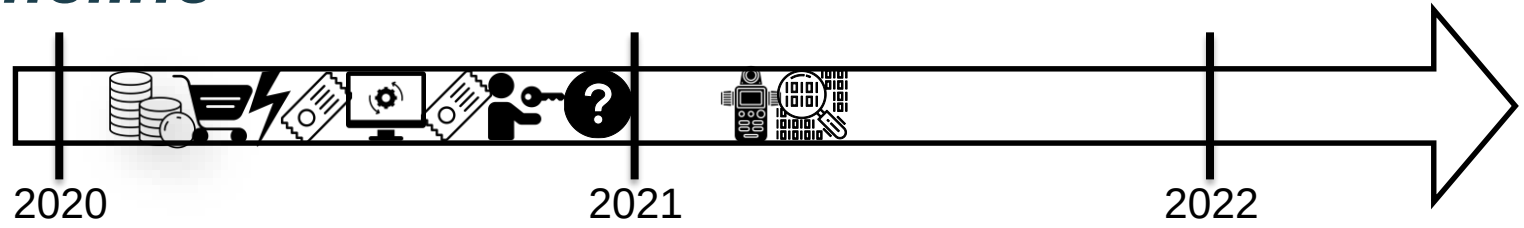
Timeline



➔ …the days passed

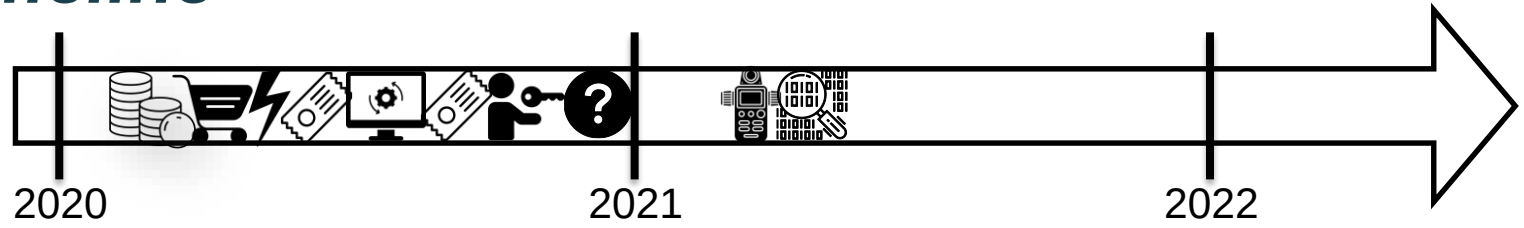
➔ March 2021: Light level metering → All ports are in RST

Timeline



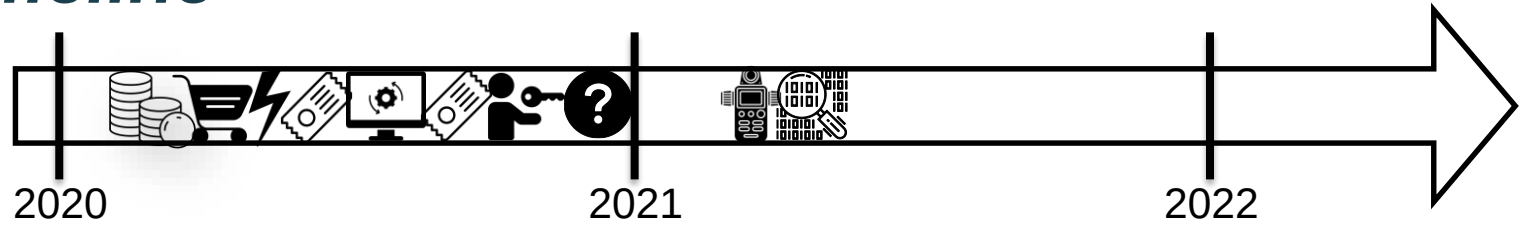
- …the days passed
- March 2021: Light level metering → All ports are in RST
- March 2021: Bit level debugging of ports‘ registers

Timeline



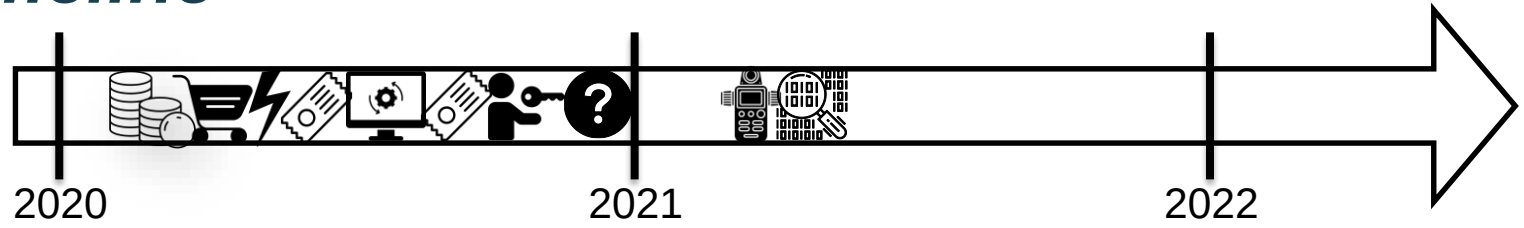
- …the days passed
- March 2021: Light level metering → All ports are in RST
- March 2021: Bit level debugging of ports‘ registers
- April 2021: Pushing bits in registers via SPI
  Reading rst_n register for port 37: „0“
  Writing „1“ to rst_n register for port 37
  Reading rst_n register for port 37:

Timeline



- …the days passed
- March 2021: Light level metering → All ports are in RST
- March 2021: Bit level debugging of ports‘ registers
- April 2021: Pushing bits in registers via SPI
  Reading rst_n register for port 37: „0“
  Writing „1“ to rst_n register for port 37
  Reading rst_n register for port 37: „XXXXXXXX“

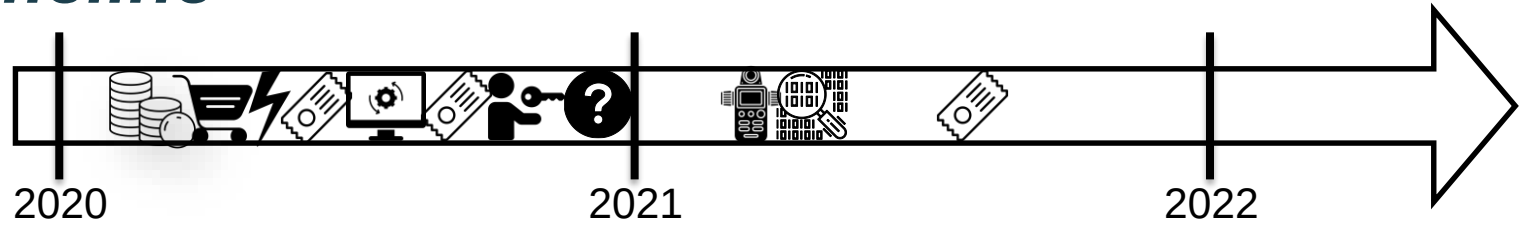
Timeline



➔ …the days passed

➔ July 2021: Help desk (new) comes to life

Timeline

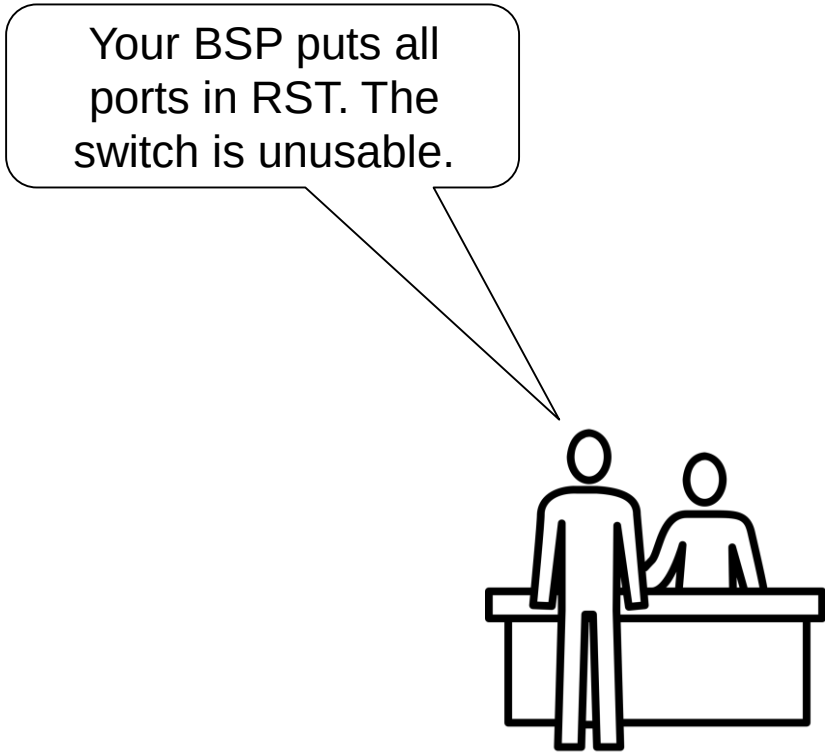


➔ …the days passed

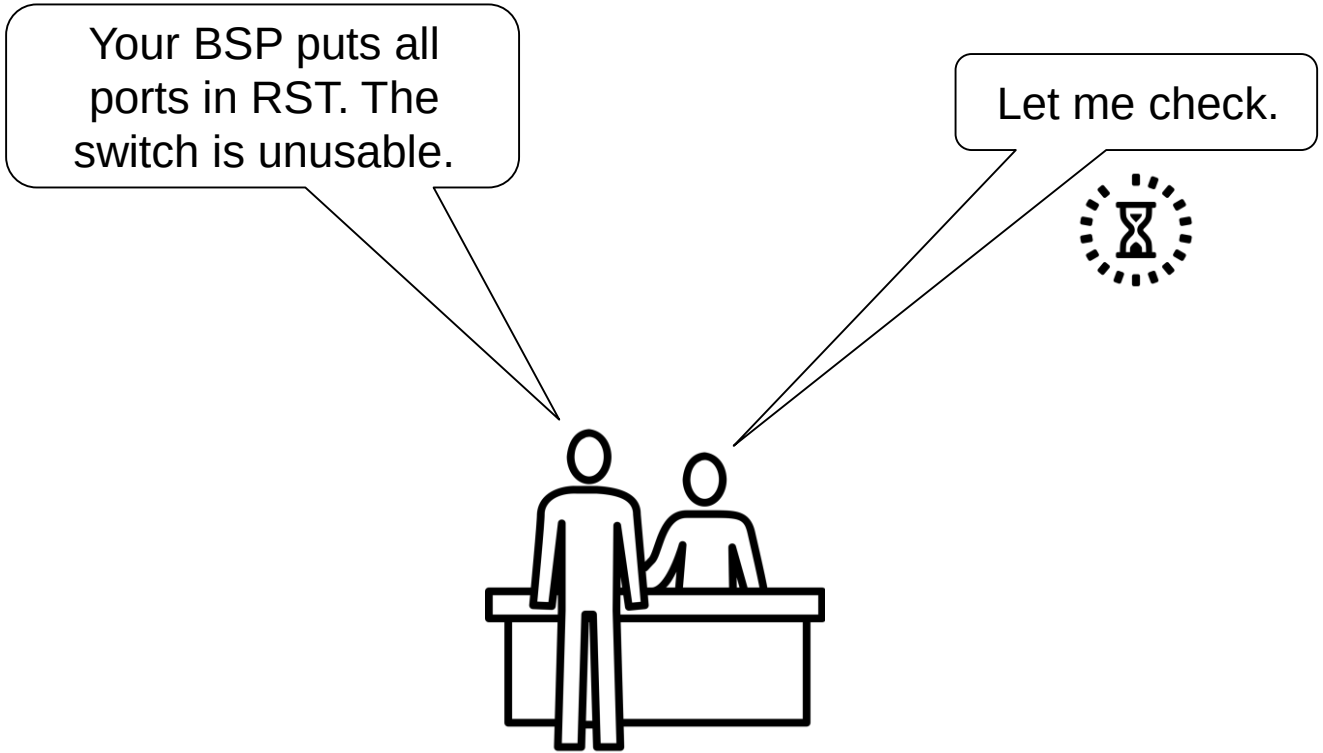
➔ July 2021: Help desk (new) comes to life

Third support ticket

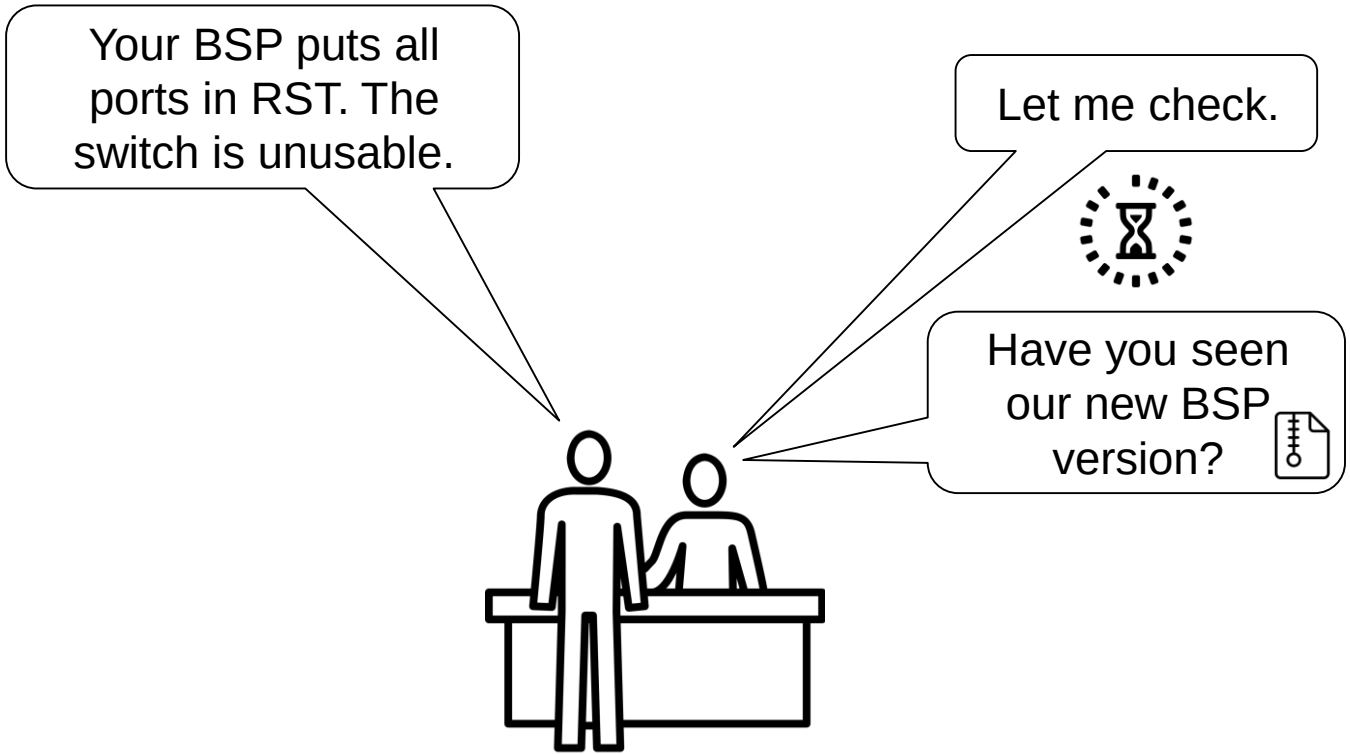
Third Help Desk Ticket



Third Help Desk Ticket



Third Help Desk Ticket

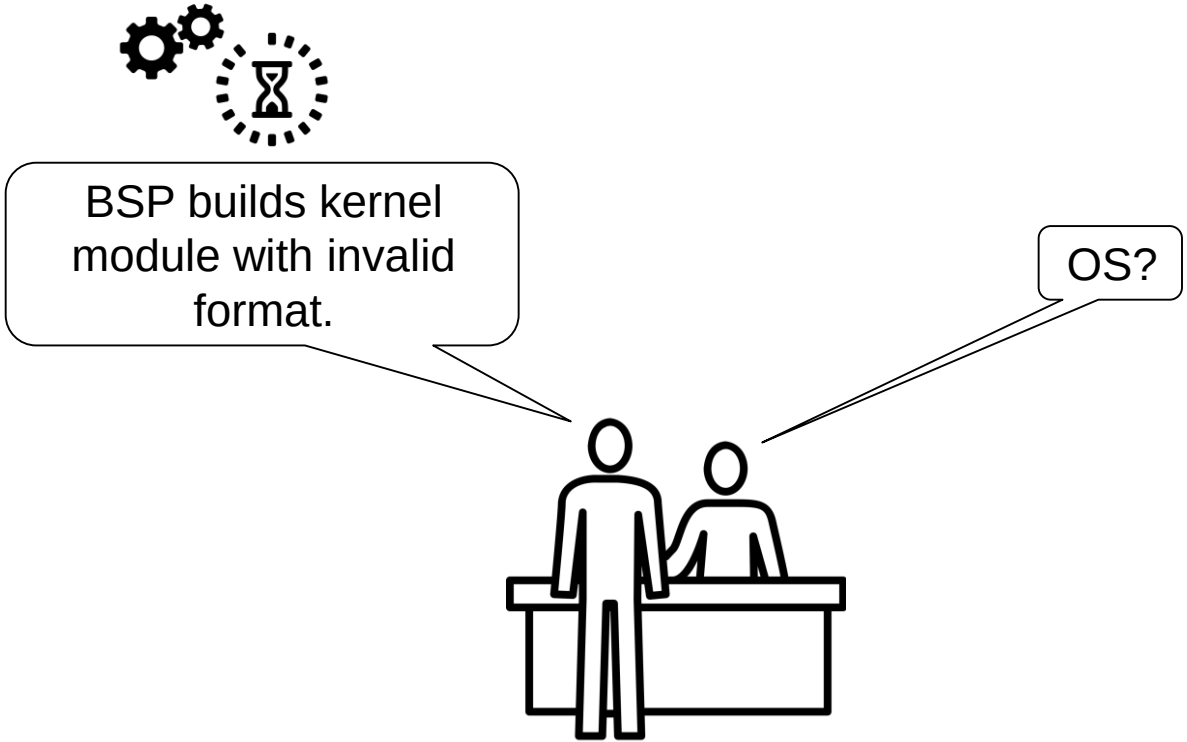


Third Help Desk Ticket

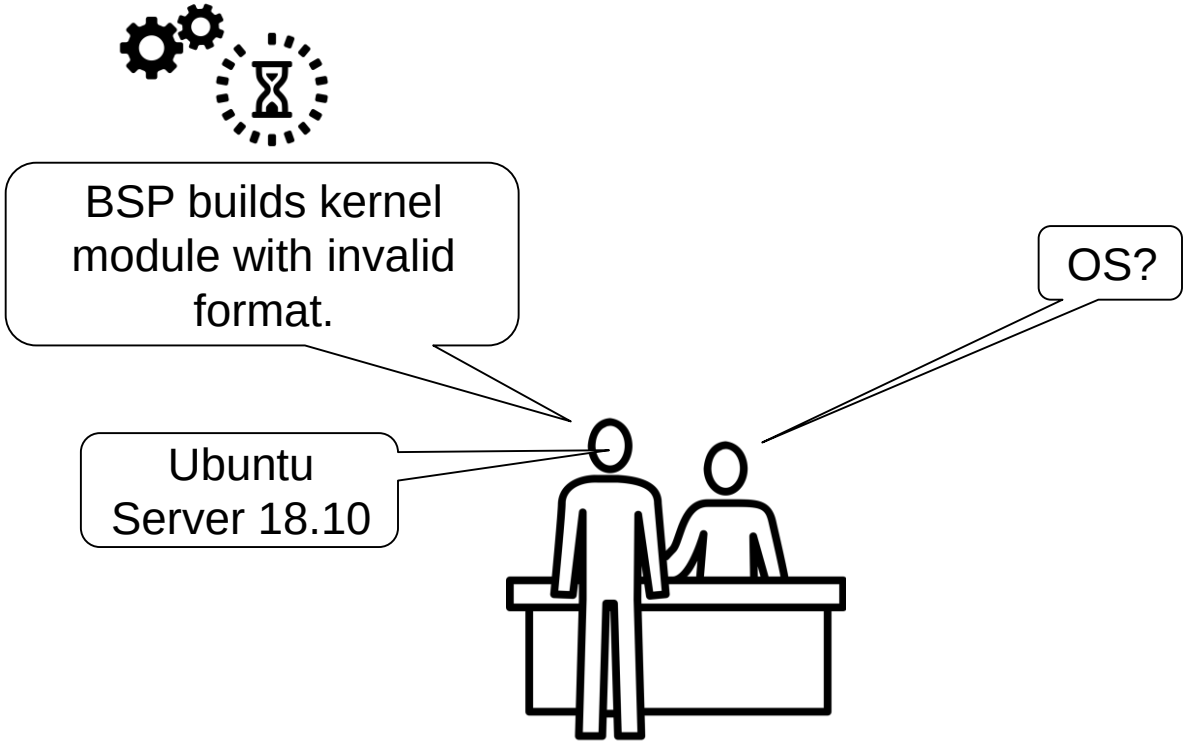
Third Help Desk Ticket

BSP builds kernel module with invalid format.

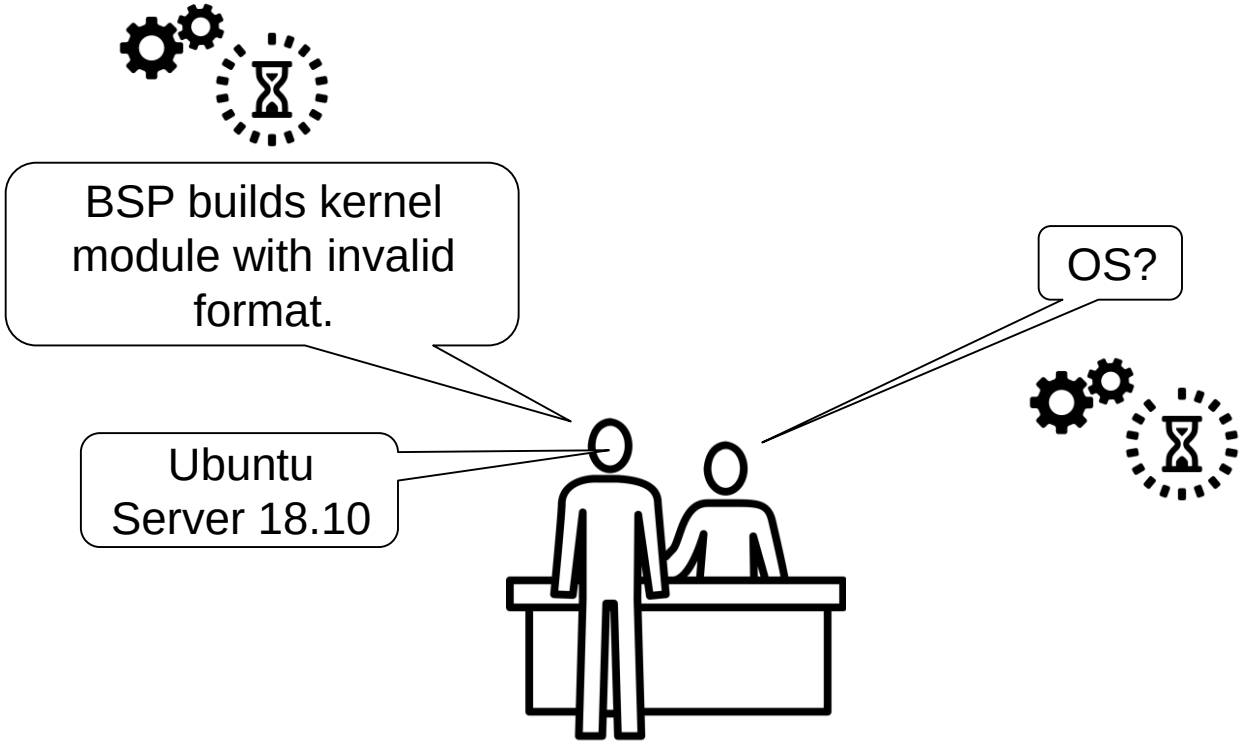
Third Help Desk Ticket



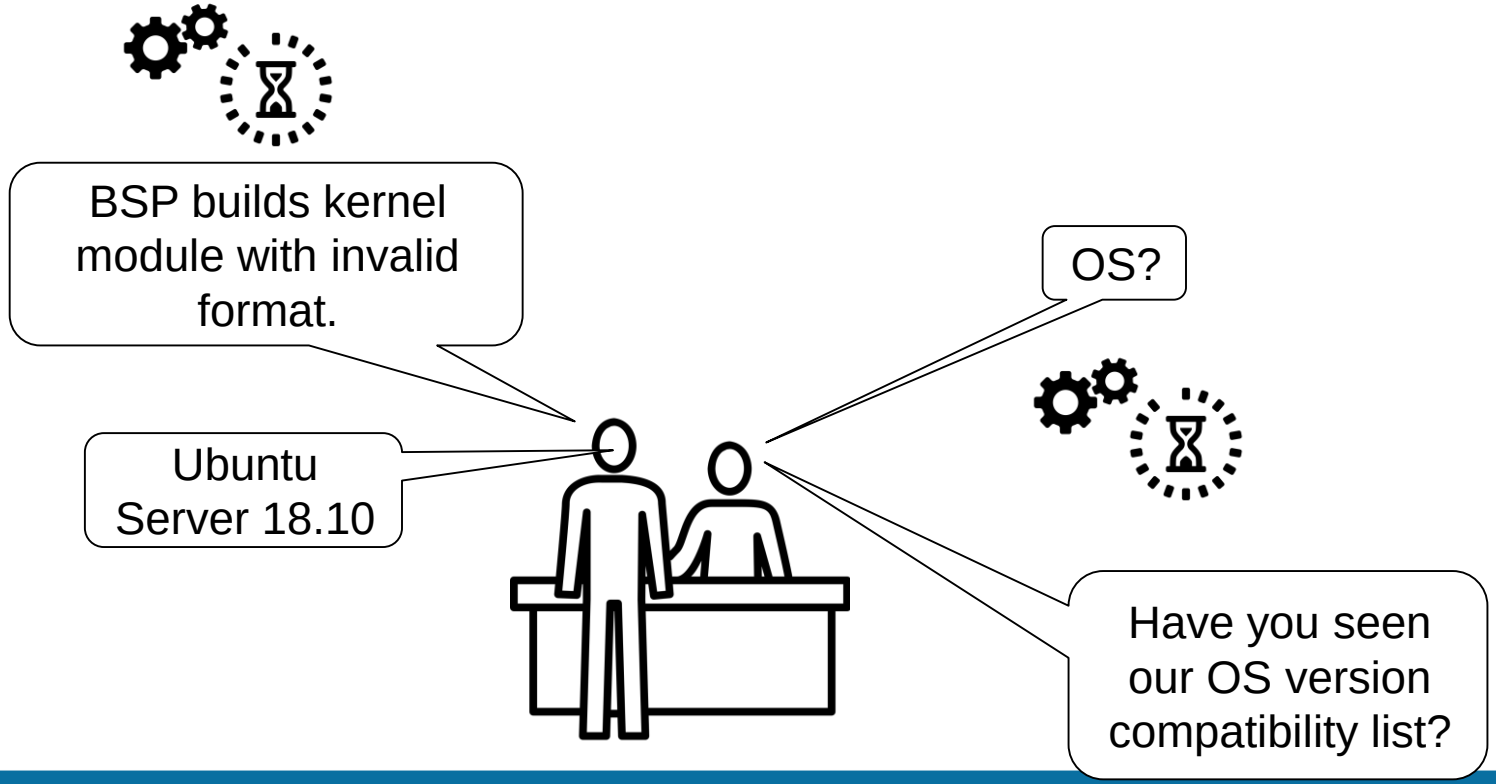
Third Help Desk Ticket



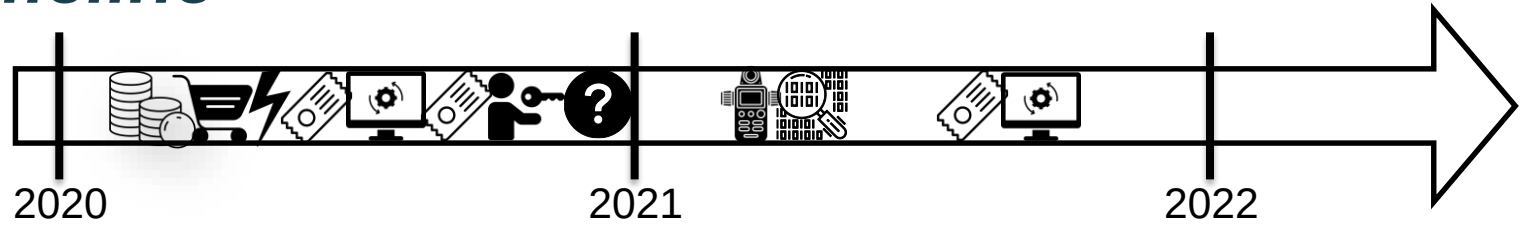
Third Help Desk Ticket



Third Help Desk Ticket

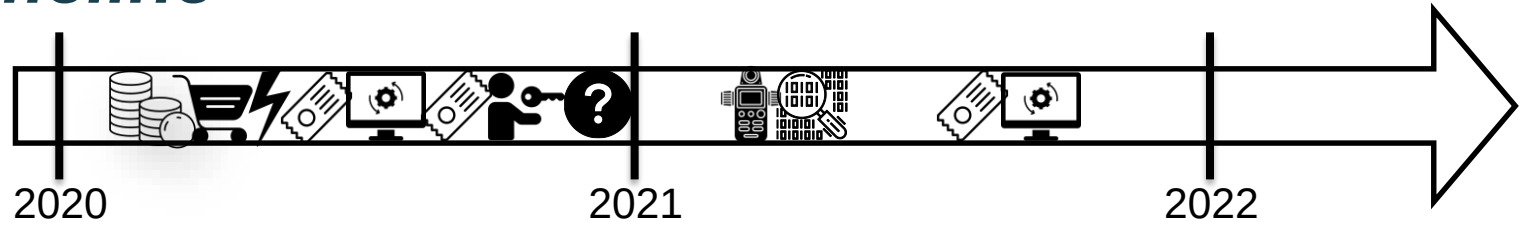


Timeline



➔ August 2021: Installation of Ubuntu Server 18.04 LTS, kernel 4.18

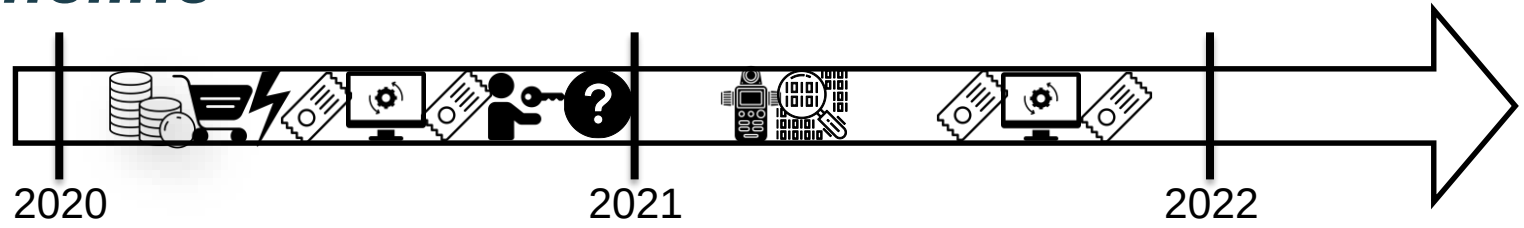
Timeline



➔ August 2021: Installation of Ubuntu Server 18.04 LTS, kernel 4.18

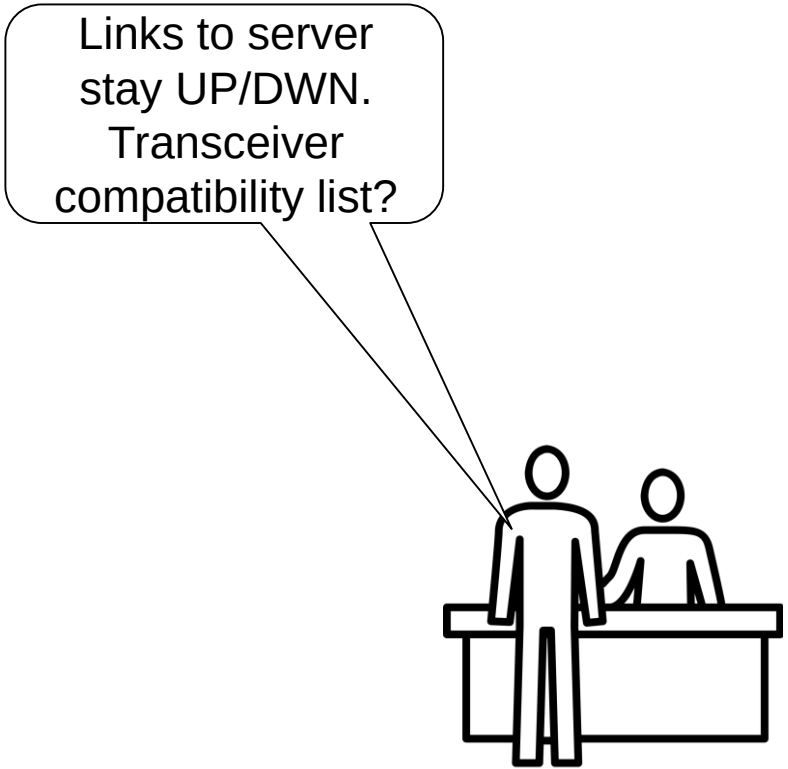
➔ Septmeber 2021: Port RDY status fixed (1-16 still weird)
Loops come UP/UP, link to server remains UP/DWN
Friends with same switch report on the same progress

Timeline

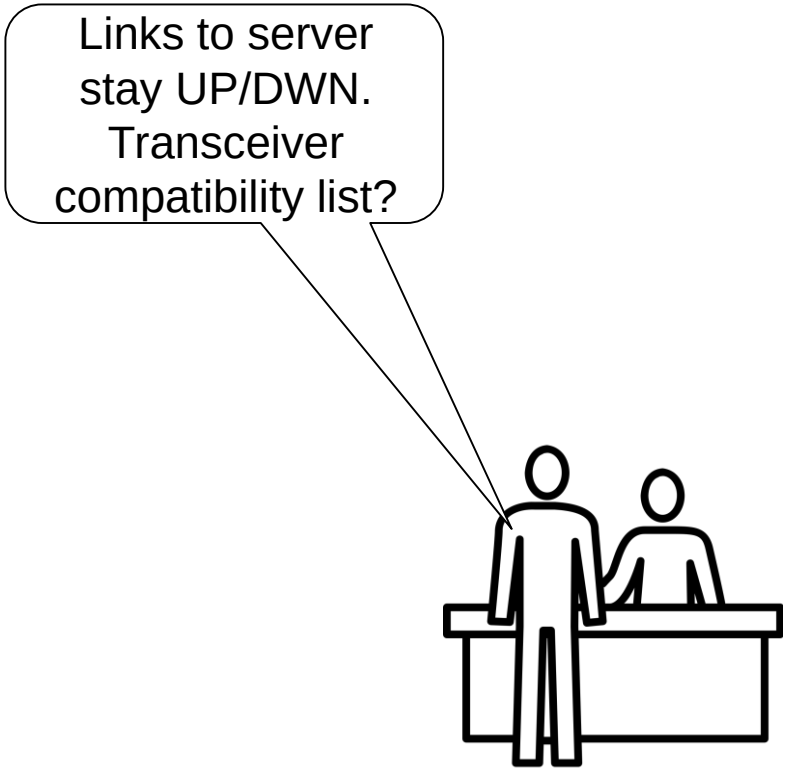


- ➔ August 2021: Installation of Ubuntu Server 18.04 LTS, kernel 4.18
- ➔ Septmeber 2021: Port RDY status fixed (1-16 still weird)
  Loops come UP/UP, link to server remains UP/DWN
  Friends with same switch report on the same progress
- ➔ October 2021: Fourth support ticket

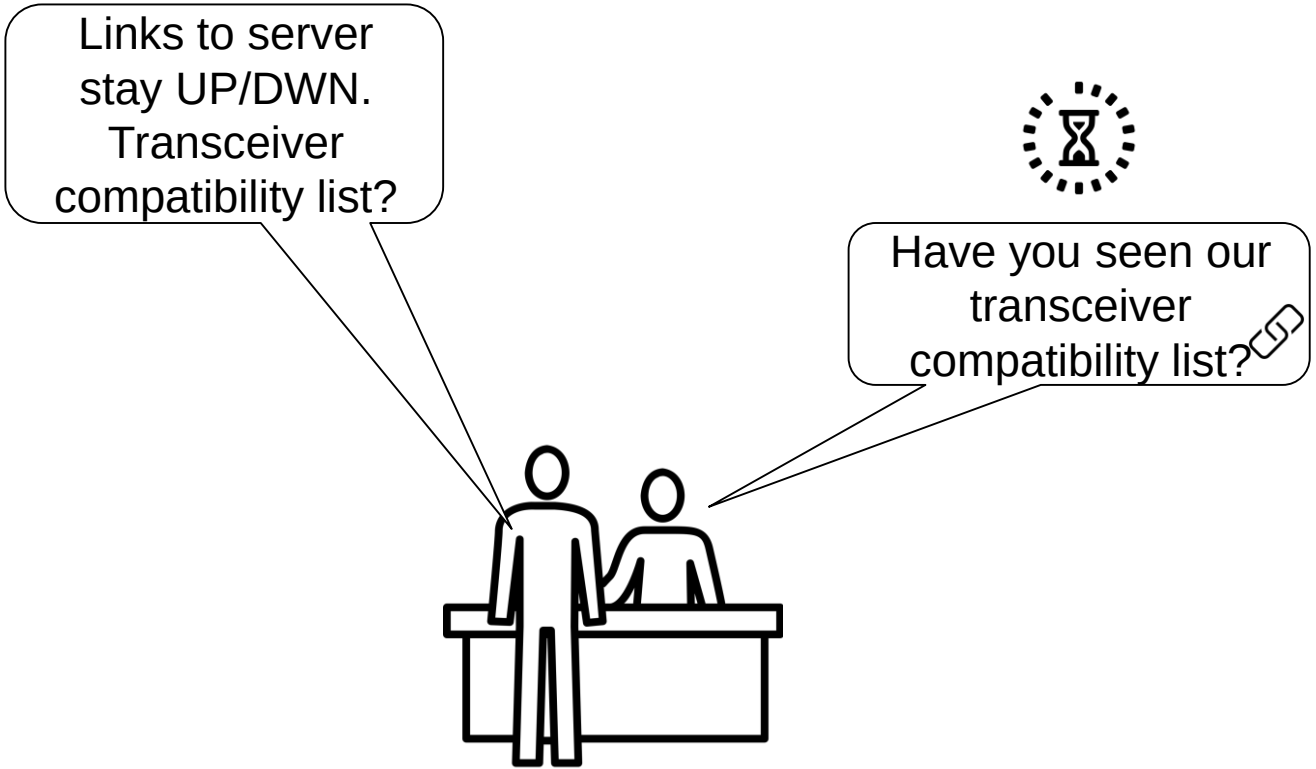
Fourth Help Desk Ticket



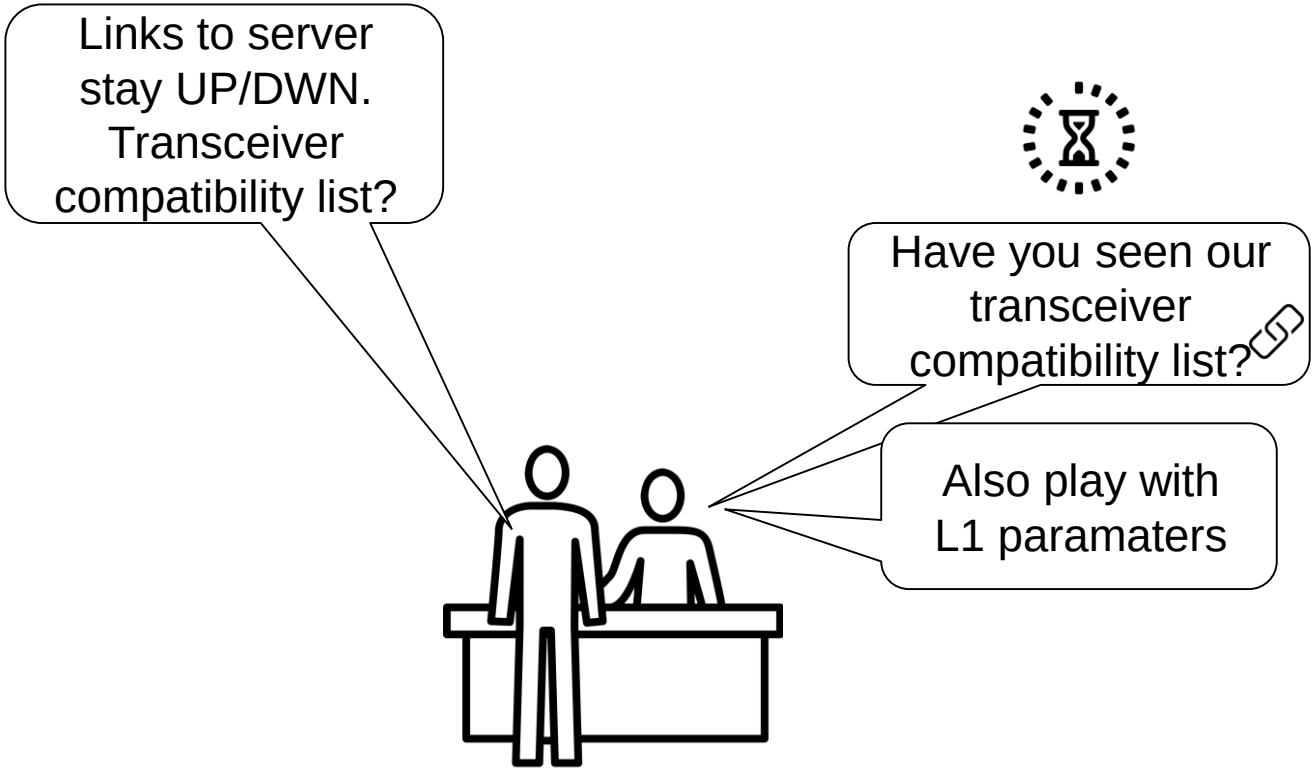
Fourth Help Desk Ticket



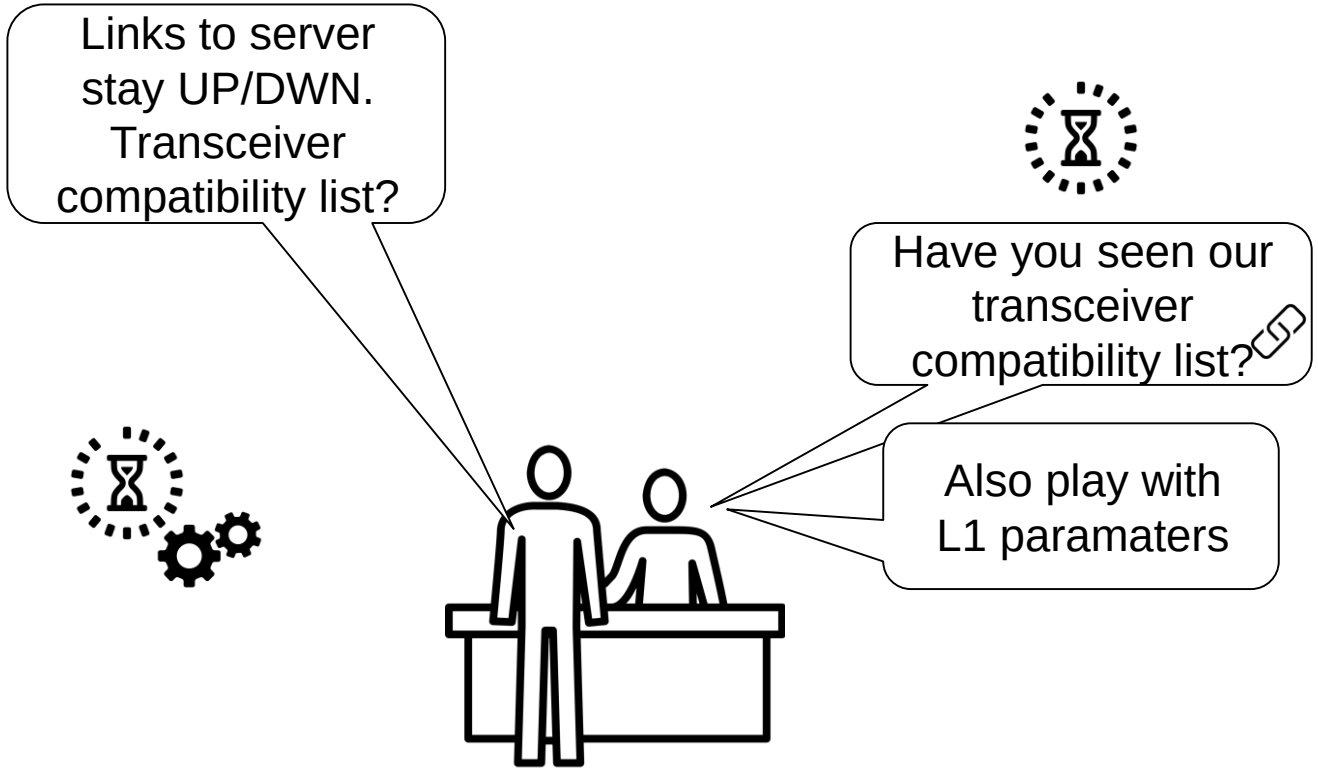
Fourth Help Desk Ticket



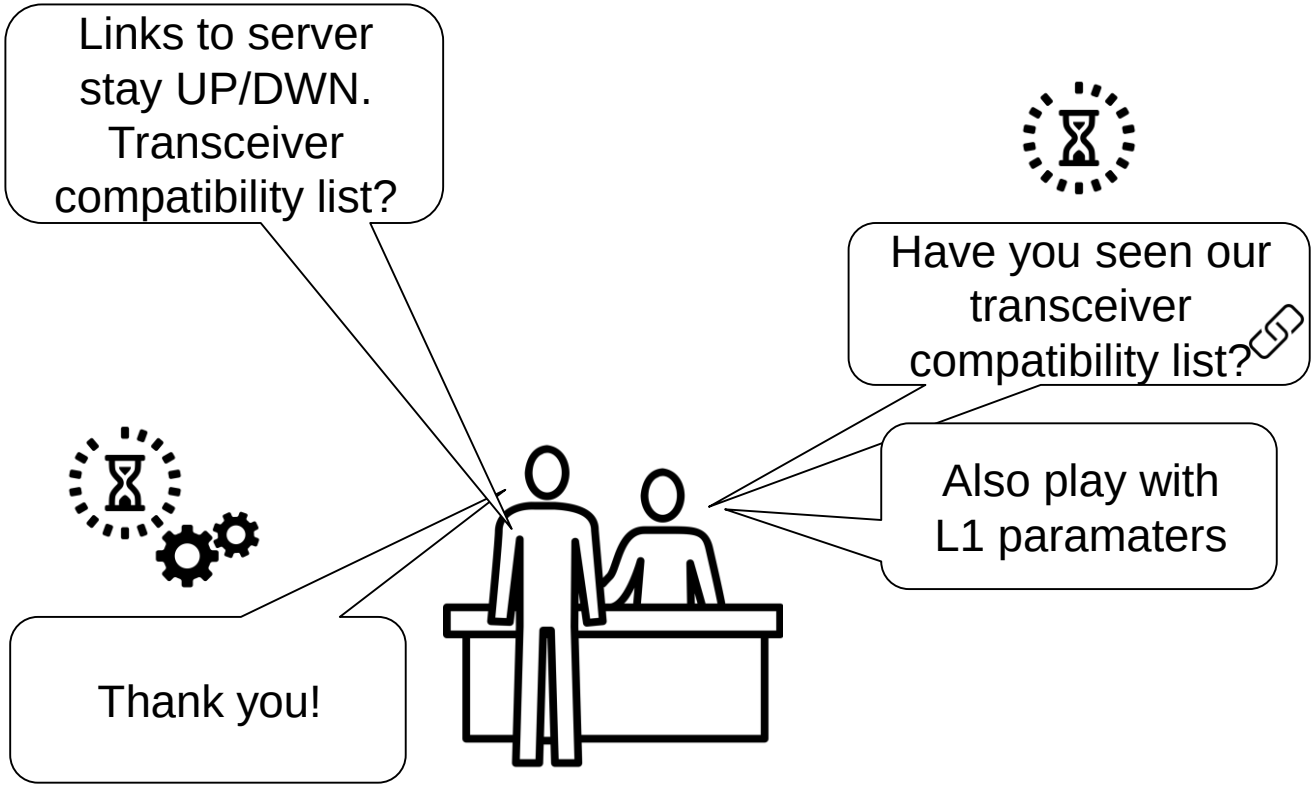
Fourth Help Desk Ticket



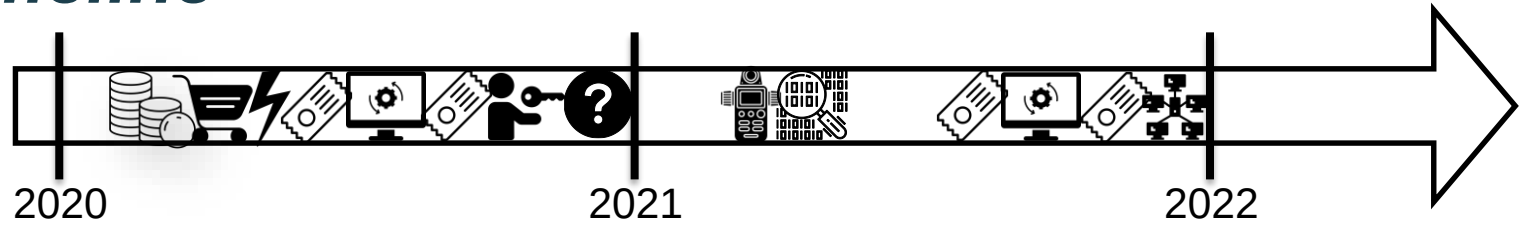
Fourth Help Desk Ticket



Fourth Help Desk Ticket

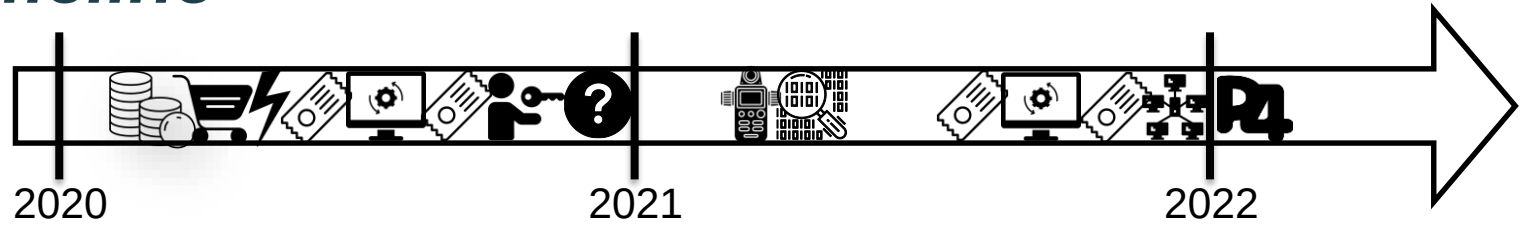


Timeline



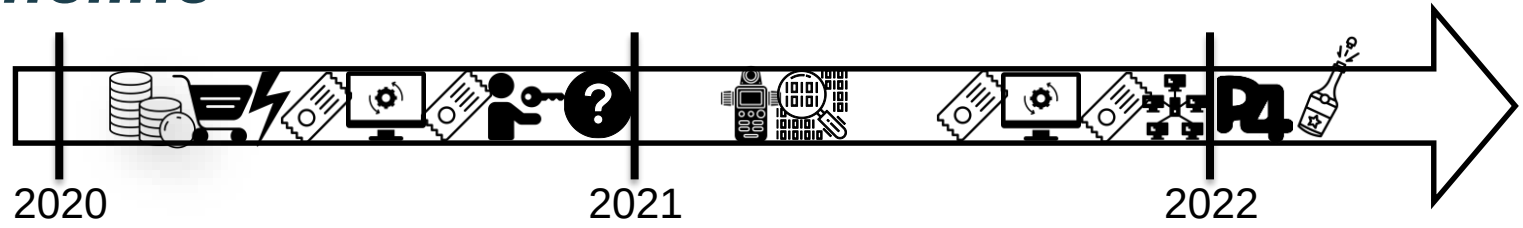
➔ December 2021: Link to server UP/UP

Timeline



- December 2021: Link to server UP/UP
- February 2022: Learning / writing / debugging P4 code

Timeline



- December 2021: Link to server UP/UP
- February 2022: Learning / writing / debugging P4 code
- March 2022: Proof-of-Concept – simple ICMP ping

Summary

➔ Exertion of 2 years

Summary

- Exertion of 2 years
- Software in use:
  - OS: Ubuntu 18.04
  - Kernel: 4.18.0-10-generic
  - BSP: bf-reference-bsp-9.5.0
  - BF-SDE: bf-sde-9.5.0
  - SAL: none (allegedly solves port 1-16 issues)

Summary

- Exertion of 2 years
- Software in use:
  - OS: Ubuntu 18.04
  - Kernel: 4.18.0-10-generic
  - BSP: bf-reference-bsp-9.5.0
  - BF-SDE: bf-sde-9.5.0
  - SAL: none (allegedly solves port 1-16 issues)
- Documentation / fixes emerged as we struggled

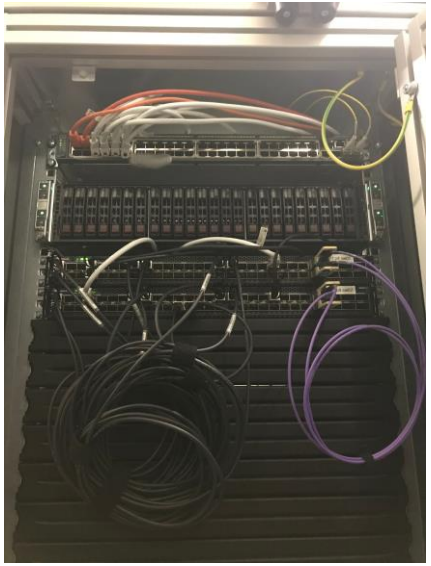
Summary

- Exertion of 2 years
- Software in use:
  - OS: Ubuntu 18.04
  - Kernel: 4.18.0-10-generic
  - BSP: bf-reference-bsp-9.5.0
  - BF-SDE: bf-sde-9.5.0
  - SAL: none (allegedly solves port 1-16 issues)
- Documentation / fixes emerged as we struggled
- Might be better to invest in the Wedge Switches

Summary

- Exertion of 2 years
- Software in use:
  - OS: Ubuntu 18.04
  - Kernel: 4.18.0-10-generic
  - BSP: bf-reference-bsp-9.5.0
  - BF-SDE: bf-sde-9.5.0
  - SAL: none (allegedly solves port 1-16 issues)
- Documentation / fixes emerged as we struggled
- Might be better to invest in the Wedge Switches

2020

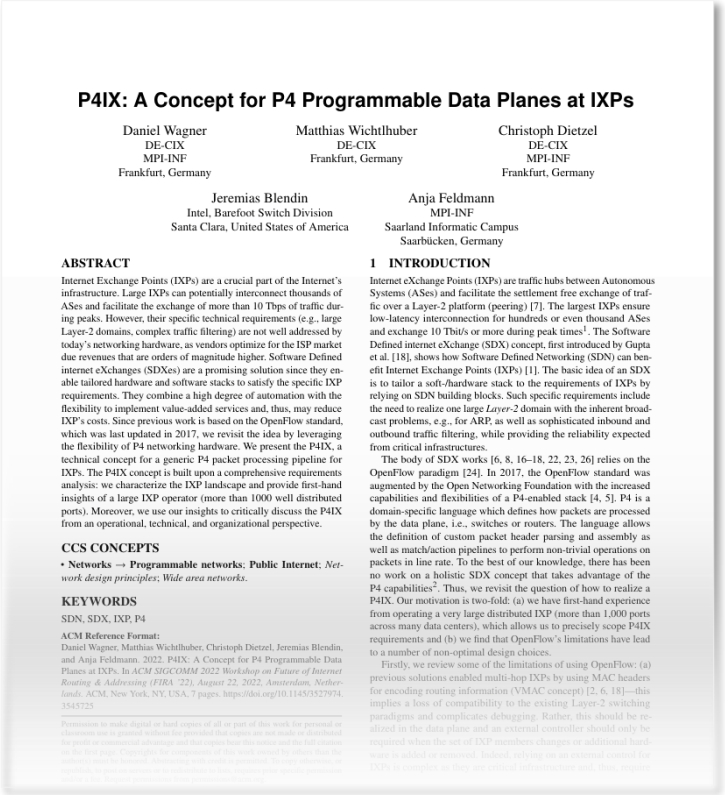


2022

Summary

- Exertion of 2 years
- Software in use:
  - OS: Ubuntu 18.04
  - Kernel: 4.18.0-10-generic
  - BSP: bf-reference-bsp-9.5.0
  - BF-SDE: bf-sde-9.5.0
  - SAL: none (allegedly solves port 1-16 issues)
- Documentation / fixes emerged as we struggled
- Might be better to invest in the Wedge Switches

SIGCOMM'22:



P4IX: A Concept for P4 Programmable Data Planes at IXPs

Daniel Wagner
DE-CIX
MPI-INF
Frankfurt, Germany

Matthias Wichtlhuber
DE-CIX
Frankfurt, Germany

Christoph Dietzel
DE-CIX
MPI-INF
Frankfurt, Germany

Jeremias Blendin
Intel, Barefoot Switch Division
Santa Clara, United States of America

Anja Feldmann
MPI-INF
Saarland Informatic Campus
Saarbücken, Germany

ABSTRACT

Internet Exchange Points (IXPs) are a crucial part of the Internet's infrastructure. Large IXPs can potentially interconnect thousands of ASes and facilitate the exchange of more than 10 Tbps of traffic during peaks. However, their specific technical requirements (e.g., large Layer-2 domains, complex traffic filtering) are not well addressed by today's networking hardware, as vendors optimize for the ISP market due revenues that are orders of magnitude higher. Software Defined internet eXchanges (SDXes) are a promising solution since they enable tailored hardware and software stacks to satisfy the specific IXP requirements. They combine a high degree of automation with the flexibility to implement value-added services and, thus, may reduce IXP's costs. Since previous work is based on the OpenFlow standard, which was last updated in 2017, we revisit the idea by leveraging the flexibility of P4 networking hardware. We present the P4IX, a technical concept for a generic P4 packet processing pipeline for IXPs. The P4IX concept is built upon a comprehensive requirements analysis: we characterize the IXP landscape and provide first-hand insights of a large IXP operator (more than 1000 well distributed ports). Moreover, we use our insights to critically discuss the P4IX from an operational, technical, and organizational perspective.

CCS CONCEPTS

• **Networks** → **Programmable networks**; **Public Internet**; *Network design principles*; *Wide area networks*.

KEYWORDS

SDN, SDX, IXP, P4

ACM Reference Format:

Daniel Wagner, Matthias Wichtlhuber, Christoph Dietzel, Jeremias Blendin, and Anja Feldmann. 2022. P4IX: A Concept for P4 Programmable Data Planes at IXPs. In *ACM SIGCOMM 2022 Workshop on Future of Internet Routing & Addressing (FIRA '22), August 22, 2022, Amsterdam, Netherlands.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3527974.3545725

1 INTRODUCTION

Internet eXchange Points (IXPs) are traffic hubs between Autonomous Systems (ASes) and facilitate the settlement free exchange of traffic over a Layer-2 platform (peering) [7]. The largest IXPs ensure low-latency interconnection for hundreds or even thousand ASes and exchange 10 Tbit/s or more during peak times[1]. The Software Defined internet eXchange (SDX) concept, first introduced by Gupta et al. [18], shows how Software Defined Networking (SDN) can benefit Internet Exchange Points (IXPs) [1]. The basic idea of an SDX is to tailor a soft-/hardware stack to the requirements of IXPs by relying on SDN building blocks. Such specific requirements include the need to realize one large *Layer-2* domain with the inherent broadcast problems, e.g., for ARP, as well as sophisticated inbound and outbound traffic filtering, while providing the reliability expected from critical infrastructures.

The body of SDX works [6, 8, 16–18, 22, 23, 26] relies on the OpenFlow paradigm [24]. In 2017, the OpenFlow standard was augmented by the Open Networking Foundation with the increased capabilities and flexibilities of a P4-enabled stack [4, 5]. P4 is a domain-specific language which defines how packets are processed by the data plane, i.e., switches or routers. The language allows the definition of custom packet header parsing and assembly as well as match/action pipelines to perform non-trivial operations on packets in line rate. To the best of our knowledge, there has been no work on a holistic SDX concept that takes advantage of the P4 capabilities[2]. Thus, we revisit the question of how to realize a P4IX. Our motivation is two-fold: (a) we have first-hand experience from operating a very large distributed IXP (more than 1,000 ports across many data centers), which allows us to precisely scope P4IX requirements and (b) we find that OpenFlow's limitations have lead to a number of non-optimal design choices.

Firstly, we review some of the limitations of using OpenFlow: (a) previous solutions enabled multi-hop IXPs by using MAC headers for encoding routing information (VMAC concept) [2, 6, 18]—this implies a loss of compatibility to the existing Layer-2 switching paradigms and complicates debugging. Rather, this should be realized in the data plane and an external controller should only be required when the set of IXP members changes or additional hardware is added or removed. Indeed, relying on an external control for IXPs is unreliable as they are critical infrastructures and, thus, require


Thank you for your attention!
Any questions?