TECHNISCHE HOCHSCHULE
OSTWESTFALEN-LIPPE
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS

Welcome

to Advanced Topics in Algorithms

Introduction

Dr.-Ing. Jens Otto

Group leader

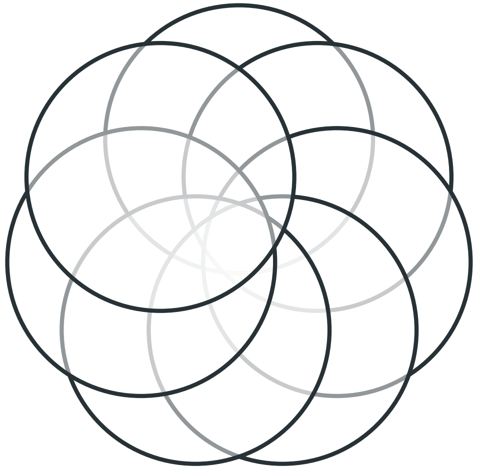
Fraunhofer IOSB-INA

Campusallee 1

32657 Lemgo, Germany

- B.Sc. and M.Sc. degrees in cognitive computer science and intelligent systems from the University of Bielefeld, Germany, in 2008 and 2010 respectively
- PhD degree at the Institute of Automation and Information Systems, Technical University of Munich, Garching, Germany in 2022
- „Command Signal Configuration for Control Strategies of Discrete Production Systems"

Research domains



Introduction

Publications

- …
- Henning, Steffen; Otto, Jens; Niggemann, Oliver; Schriegel, Sebastian: **A Descriptive Engineering Approach for Cyber-Physical Systems.** In: 19th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) Barcelona, Spain, Sep 2014.
- Otto, Jens; Henning, Steffen; Niggemann, Oliver: **Why cyber-physical production systems need a descriptive engineering approach – a case study in plug & produce.** In: 2nd International Conference on System-integrated Intelligence (SysInt) Bremen, Germany, Jul 2014.
- Dürkop, Lars; Trsek, Henning; Otto, Jens; Jasperneite, Jürgen: **A field level architecture for reconfigurable real-time automation systems.** In: 10th IEEE Workshop on Factory Communication Systems Toulouse, May 2014.
- Otto, Jens; Niggemann, Oliver: **Automatic Parameterization of Automation Software for Plug-and-Produce.** In: AAAI-15 Workshop on Algorithm Configuration (AlgoConf) Austin, Texas, USA, Jan 2015.
- Niggemann, Oliver; Henning, Steffen; Schriegel, Sebastian; Otto, Jens; Anis, Anas: **Models for Adaptable Automation Software - An Overview of Plug-and-Produce in Industrial Automation.** In: Modellbasierte Entwicklung eingebetteter Systeme (MBEES) S.: 73-82, Dagstuhl, Germany, Mar 2015.
- Otto, Jens; Vogel-Heuser, Birgit; Niggemann, Oliver: **Optimizing modular and reconfigurable cyber-physical production systems by determining parameters automatically.** In: IEEE 14th International Conference on Industrial Informatics (INDIN) S.: 1100-1105, Jul 2016
- Otto, Jens, Birgit Vogel-Heuser, and Oliver Niggemann: **Automatic parameter estimation for reusable software components of modular and reconfigurable cyber-physical production systems in the domain of discrete manufacturing.** In: IEEE Transactions on Industrial Informatics 14.1 (2018): 275-282.
- Otto, Jens; Vogel-Heuser, Birgit; Niggemann, Oliver: **Online Parameter Estimation for Cyber-Physical Production Systems.** In: at - Automatisierungstechnik at - Automatisierungstechnik, Aug 2018.
- …

https://scholar.google.com/citations?user=InrpcDkAAAAJ&hl=en

Introduction: Fraunhofer IOSB-INA

- **Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB**
- **Industrial Automation branch INA of Fraunhofer IOSB**
- **Part of the Fraunhofer Society**
- **New building 2019**

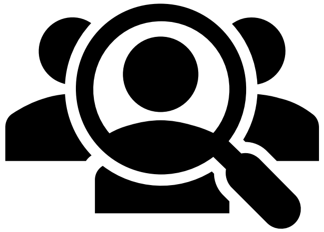
Introduction: SmartFactoryOWL

- **Research factory for intelligent automation**
- **Joint institution of the University of Applied Sciences and Arts and Fraunhofer IOSB-INA**

https://smartfactory-owl.de/?lang=en

Introduction

- **Please present yourself shortly:**
  - **Name**
  - **Background**
  - **Previous university**
  - **Topic of bachelor thesis**
  - **Main interest**
- **Your expectations for this module?**

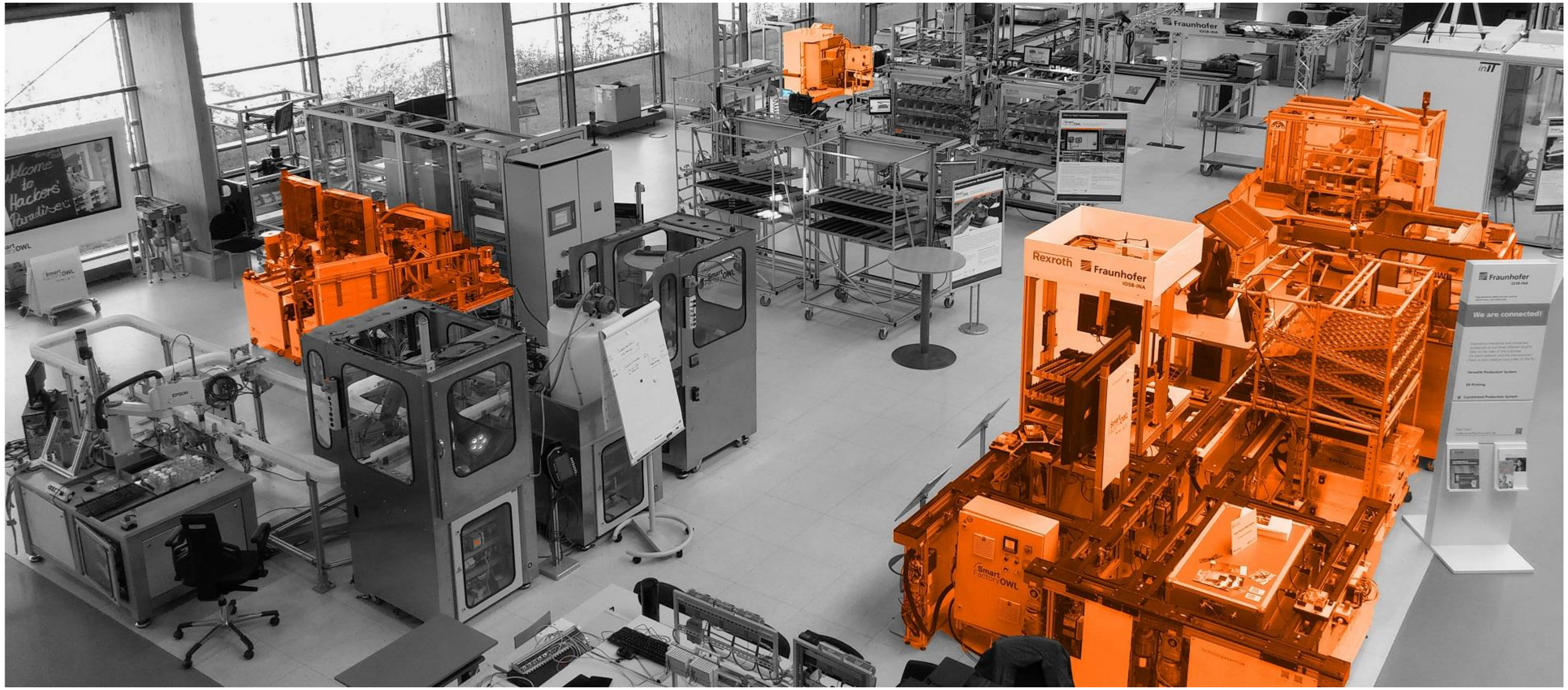


Members of this Lecture

Motivation: Use Case Production Systems

Motivation: Use Case Production Systems

Plant 1 | Plant 2 | Plant 3

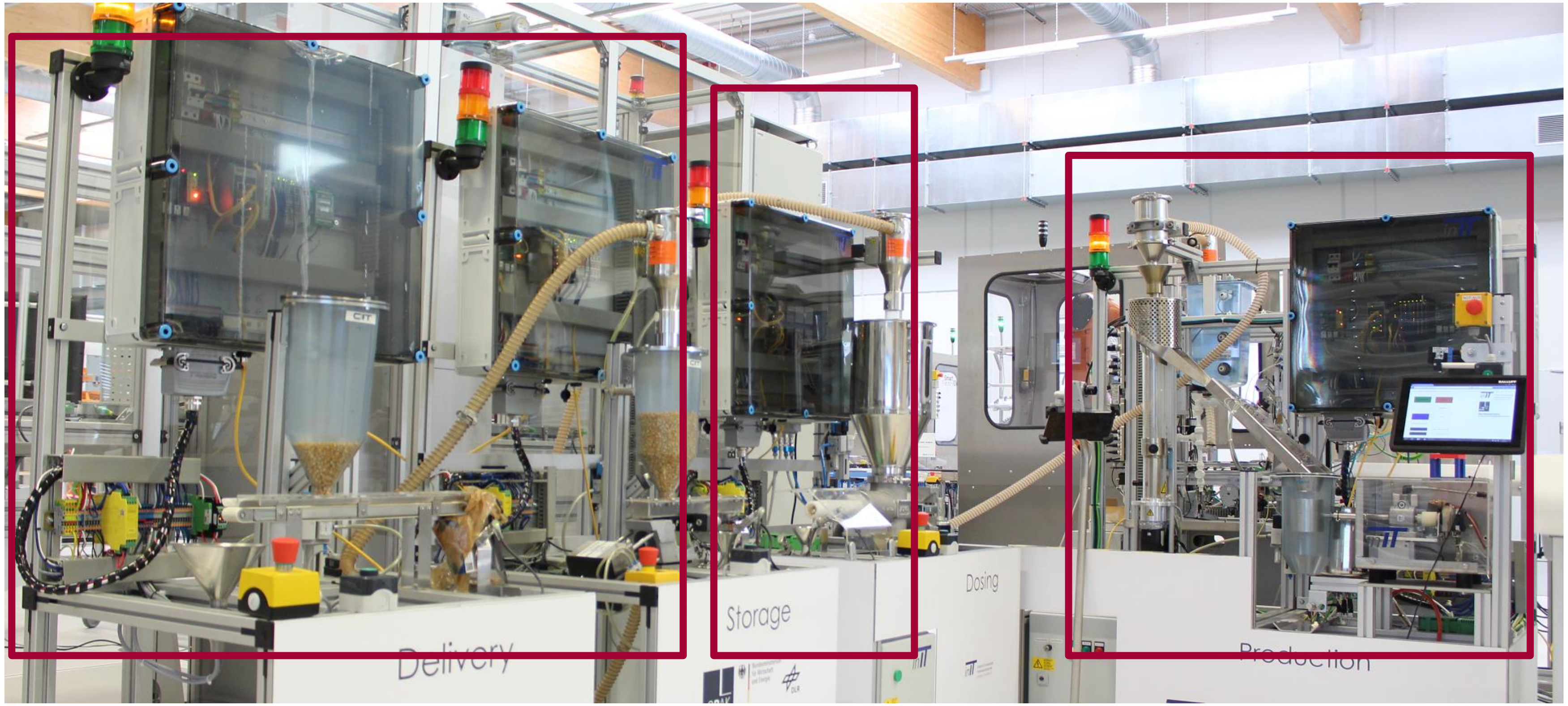


Motivation: Use Case Production Systems

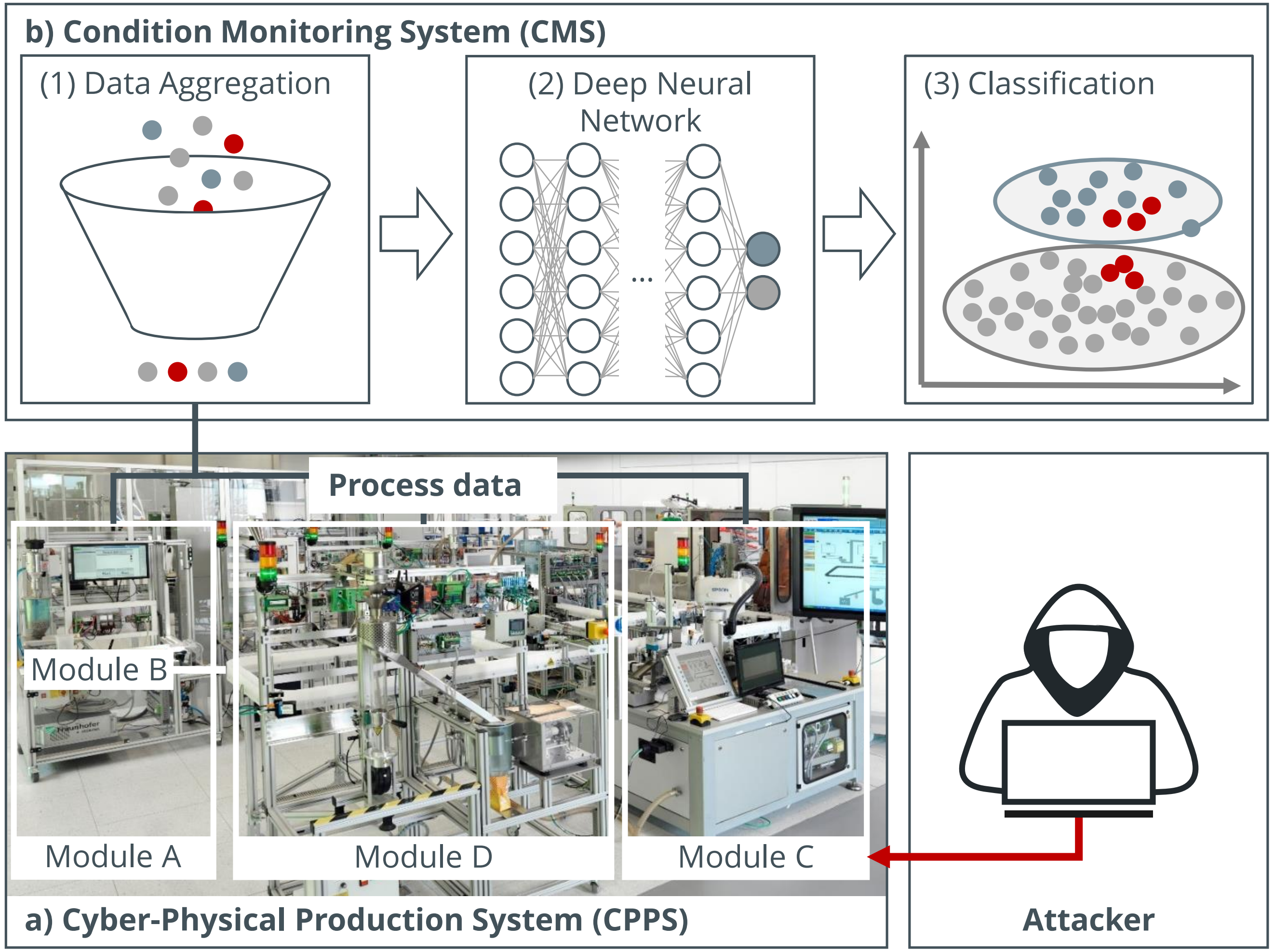
Production module 1

Production module 2

Production module 3



Motivation: Adversarial Examples



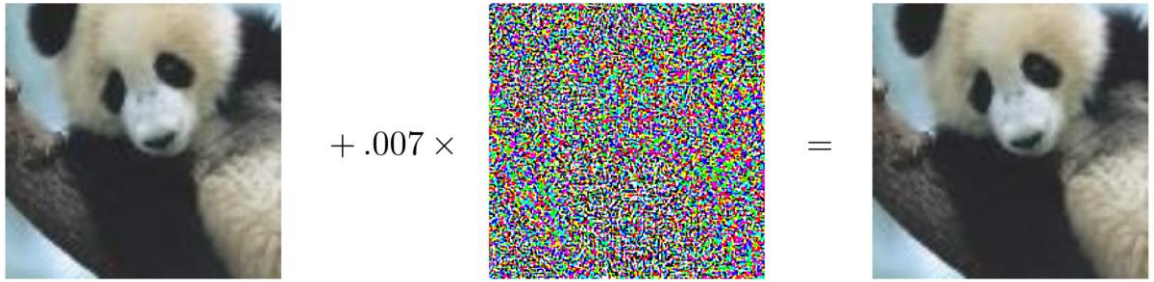
Motivation: Adversarial Examples

Deep Neural Network (DNN):

- Given is a DNN: $\boldsymbol{F(x, \theta) = Y}$
- DNN classifies input $\boldsymbol{x}$ as class $\boldsymbol{Y}$, using a parameter set $\boldsymbol{\theta}$

Adversarial Example (AE):

- AE $\boldsymbol{x'}$ deviates minimally from the original **x**
- $\boldsymbol{x'}$ is generated by applying a perturbation $\boldsymbol{\Delta_x}$
- $\boldsymbol{x'}$ is incorrectly classified by the DNN: $\boldsymbol{F(x', \theta) \neq Y}$



„panda“ perturbation „gibbon“

Motivation: Adversarial Examples

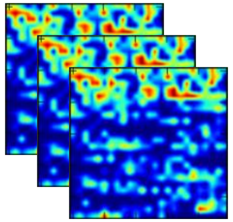
(1) Process data

Time	X
09:59	0
10:00	81.16
10:01	42.33
10:02	123.49
10:03	124.65
10:04	50.81
10:05	123.97
10:06	0
…	

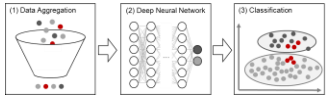
(2) CyberProtect

(1) Select process data attributes
(2) Select perturbation
(3) Apply perturbation

(3) Adversarial Examples



Condition Monitoring

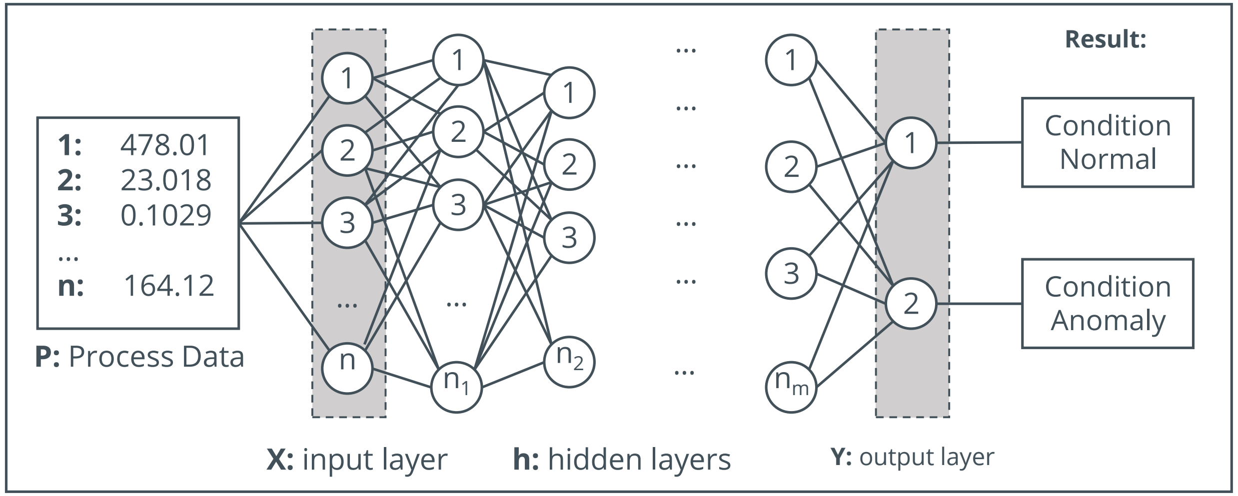


Objective: Prevent misclassification caused by adversarial example attacks

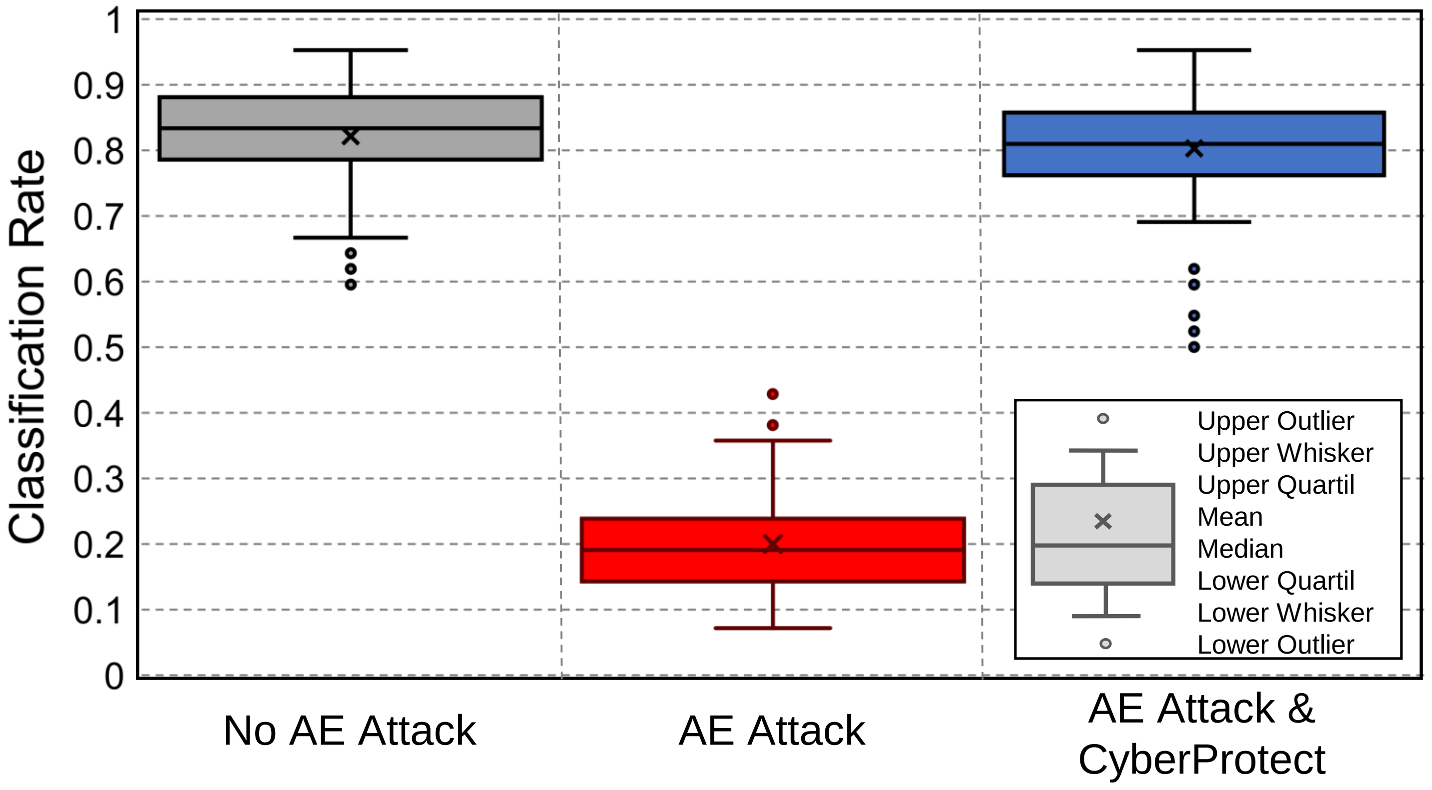
Motivation: Adversarial Examples

- Secom dataset recorded from semi-conductor manufacturing process
  - 590 variables from sensor signal
  - 1567 manufacturing cycles
  - Labeled as normal or anomaly production cycle
- Implementation based on python libraries *Tensorflow* and *Cleverhans*

DNN Architecture:
- 590 input neurons
- 4 hidden layers
- ReLU as activation



Motivation: Adversarial Examples

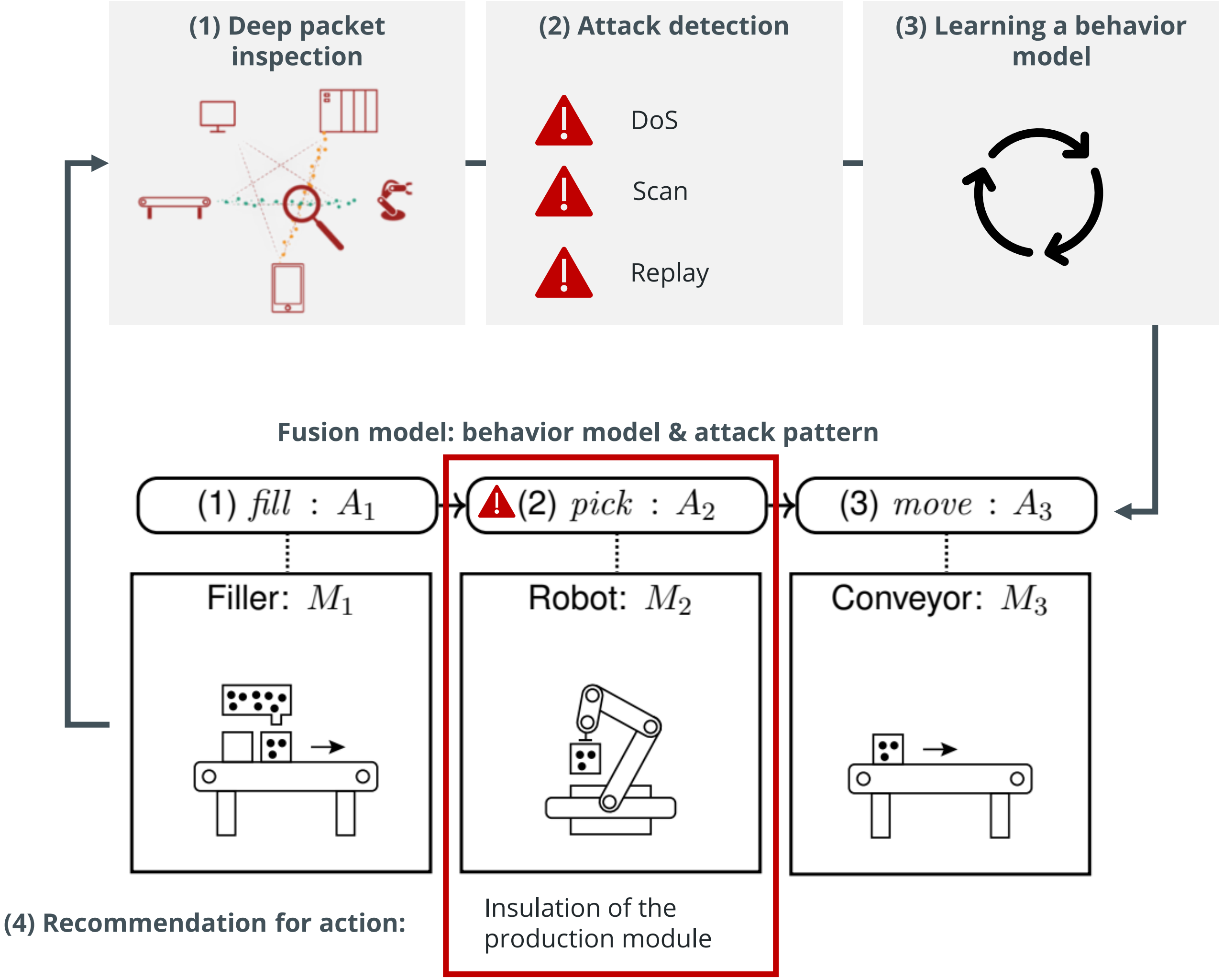


CyberProtect prevents misclassification caused by AE attacks

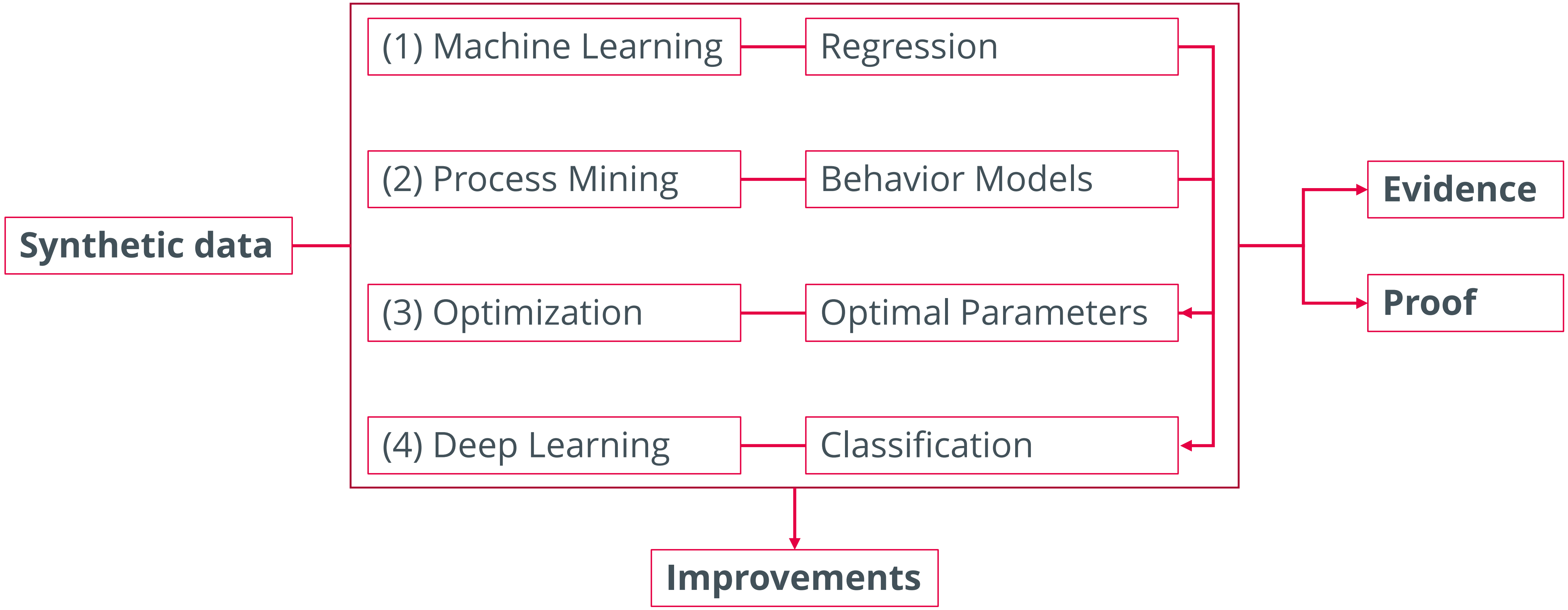
Motivation: Research Project HAIP

Fusion model consisting of behavior model and attack patterns:

(1) Analysis of data packets (deep packet inspection)
(2) Attack detection through pattern analysis of communication data
(3) Learning a behavior model from process data (process mining)
(4) Recommended action for the plant operator
    (1) Insulation of the production module
    (2) Shutdown of the production plant



Overview: Advanced Topics in Algorithms



Overview: Practical Part

Scipy

NetworkX

NumPy

Matplotlib

Scikit-learn

Keras

gplearn

PM4Py

Pandas

...

Implementation

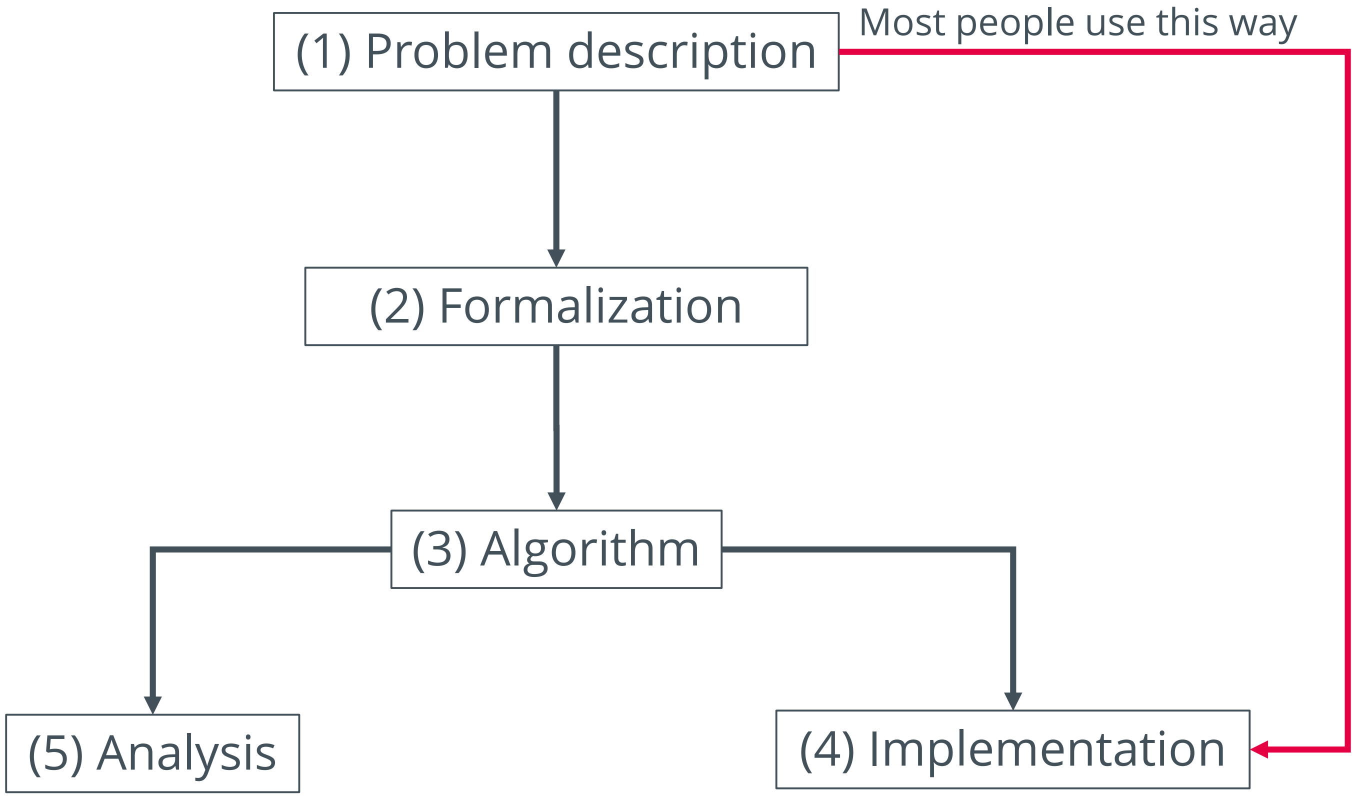
Documentation

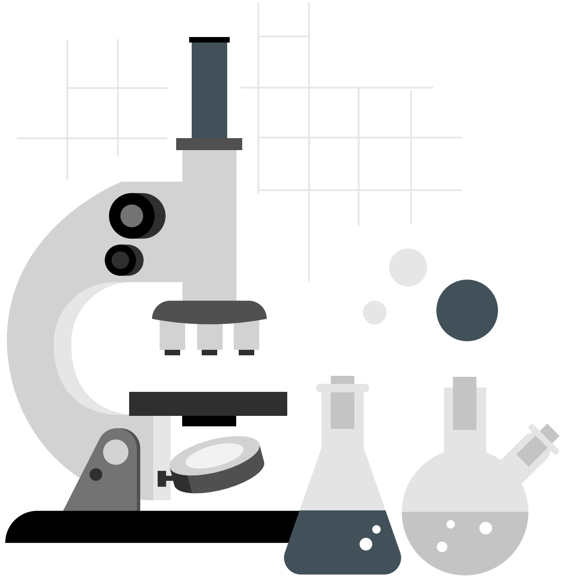
LaTeX

TikZ

...

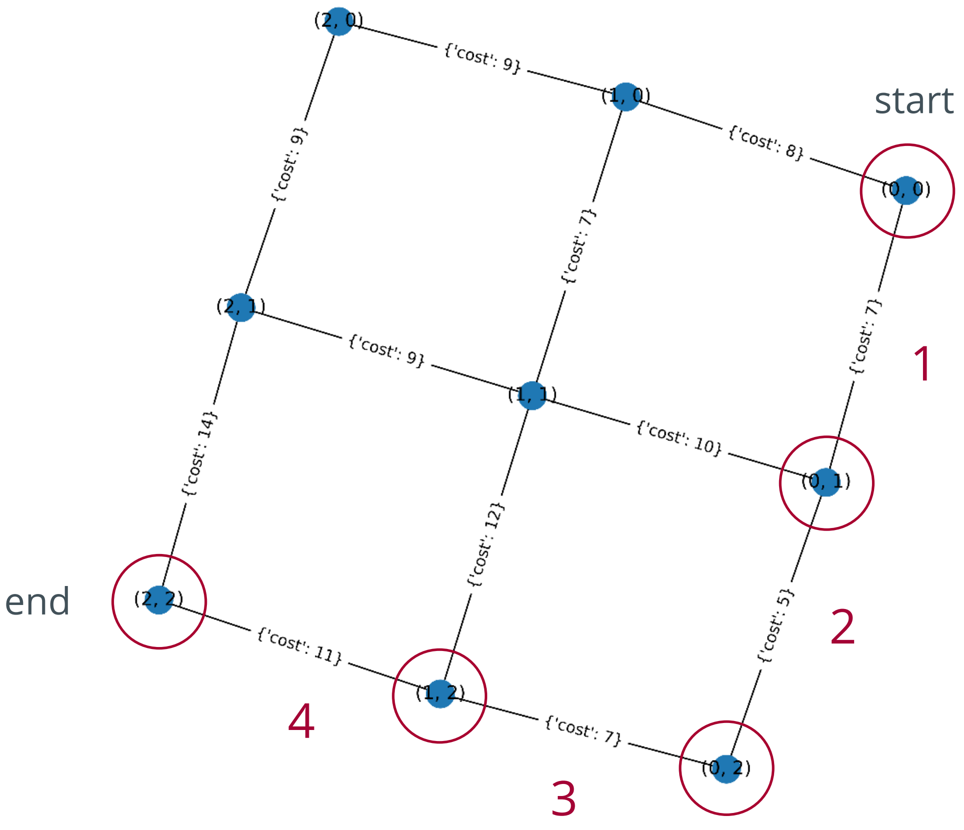
Algorithm Design





It's not rocket science

Algorithm Design: Example A*



cost of the path from the start node

$$f(n) = g(n) + h(n)$$

next node

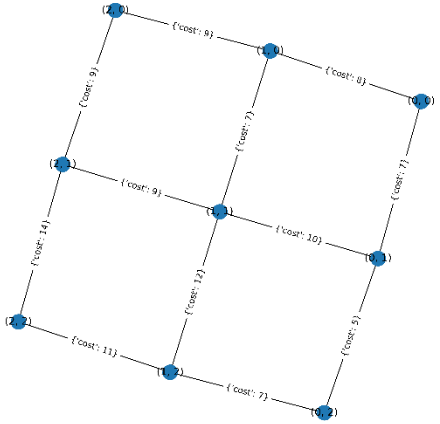
next	end	Euclidean distance
(1, 0)	(2, 2)	2.2360
(0, 1)	(2, 2)	2.2360
(0, 0)	(2, 2)	2.8284
(1, 1)	(2, 2)	1.4142
(0, 2)	(2, 2)	2.0
(2, 0)	(2, 2)	2.0
(1, 2)	(2, 2)	1.0
(2, 1)	(2, 2)	1.0
(2, 2)	(2, 2)	0.0

Algorithm Design: (1) Problem Description

The shortest path of a directed graph should be calculated.

Textual description of the problem

Input



directed graph → **Algorithm** → shortest path

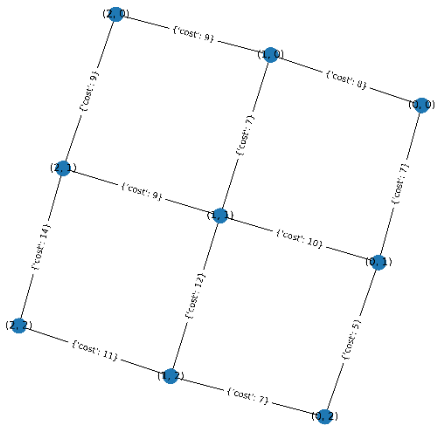
Output

[(0, 0), (0, 1), (0, 2), (1, 2), (2, 2)]

https://networkx.org

Algorithm Design: (2) Formalization

Input



directed graph → **Algorithm** → shortest path

Output

[(0, 0), (0, 1), (0, 2), (1, 2), (2, 2)]

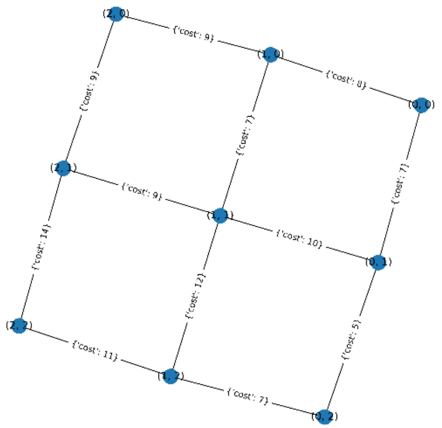
LaTeX

The directed graph D is defined as quadruple $D = (L, l_0, l_e, T)$. The finite set L describes the nodes, where l_0 describes the start node and l_e describes the end node. $T \subseteq L \times L$ describes the transitions between nodes.

$P = ((p_1, p_2)_i \mid i \in \{1, \ldots, m\})$ is defined as an n-tuple of...

Algorithm Design: (3) Algorithm

Input



directed graph → **Algorithm** → shortest path

Output

[(0, 0), (0, 1), (0, 2), (1, 2), (2, 2)]

LaTeX

\usepackage[linesnumbered,ruled,vlined]{algorithm2e}

Algorithm 1: The A^* ...

Input: Directed graph D
Output: n-tuple of point tuple

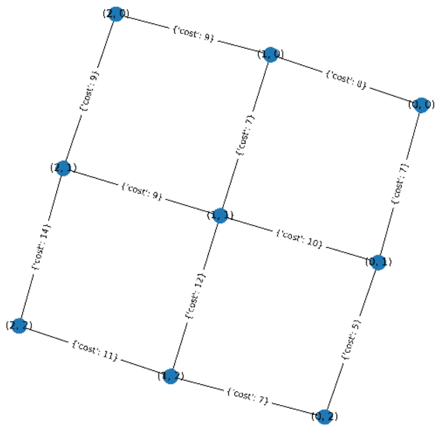
```
// Step 1:  (Calculate shortest path):
```

$P \leftarrow A^*(D)$

return P

Algorithm Design: (3) Algorithm

Input



Output

directed graph → **Algorithm** → shortest path → [(0, 0), (0, 1), (0, 2), (1, 2), (2, 2)]

LaTeX

\usepackage[linesnumbered,ruled,vlined]{algorithm2e}

Algorithm 1: The A^* ...

Input: Directed graph D
Output: n-tuple of point tuple

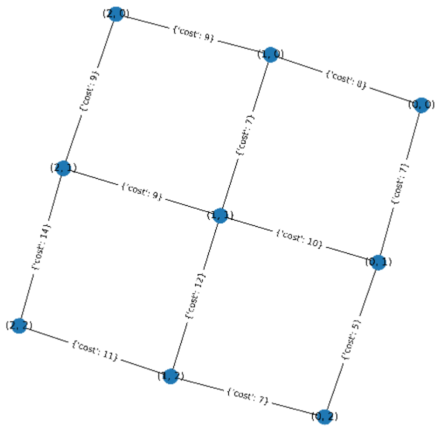
`// Step 1: (Calculate shortest path):`

$P \leftarrow A^*(D)$

return P

Algorithm Design: (4) Implementation

Input



directed graph → **Algorithm** → shortest path

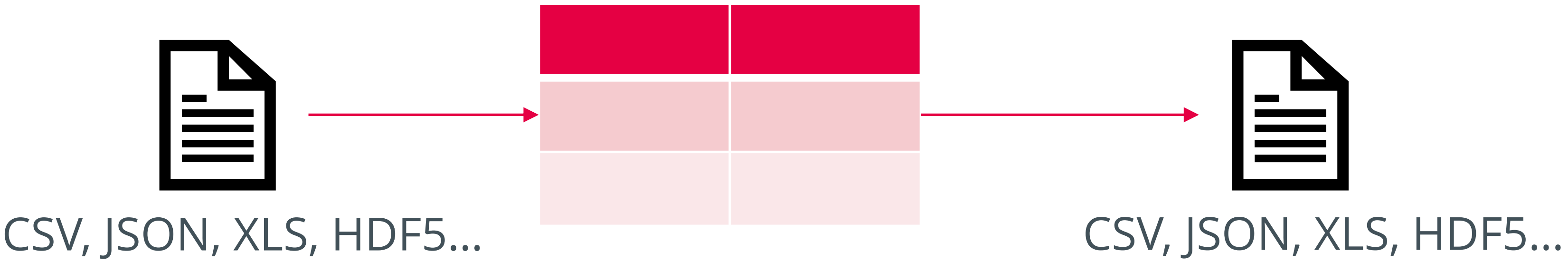
Output

[(0, 0), (0, 1), (0, 2), (1, 2), (2, 2)]

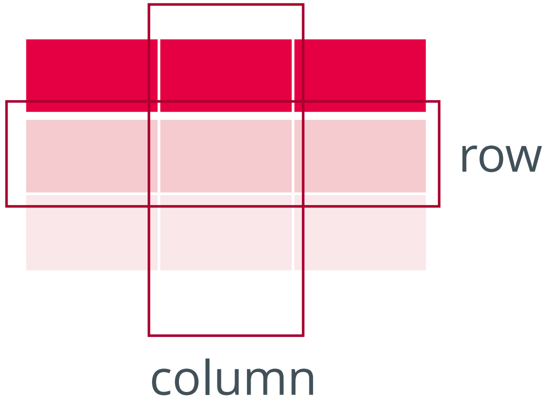
Python

```
Run Cell | Run Below | Debug Cell
# %%
import random
import math
import networkx as nx
import matplotlib.pyplot as plt
import matplotlib

from networkx.classes.graph import Graph
from networkx.algorithms.shortest_paths.weighted import _weight_function
from heapq import heappush, heappop
from itertools import count

matplotlib.rcParams.update({'font.size': 22, 'font.family': 'Arial'})

def star(G, source, target, heuristic=None, weight="weight"):

    push = heappush
    pop = heappop
    weight = _weight_function(G, weight)
```

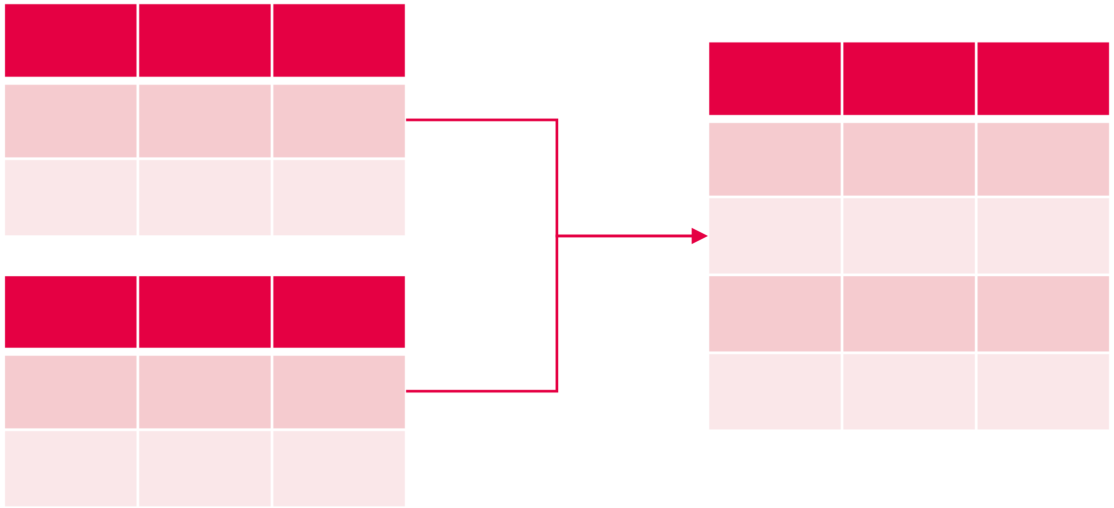
Excurse: Pandas Data Frame



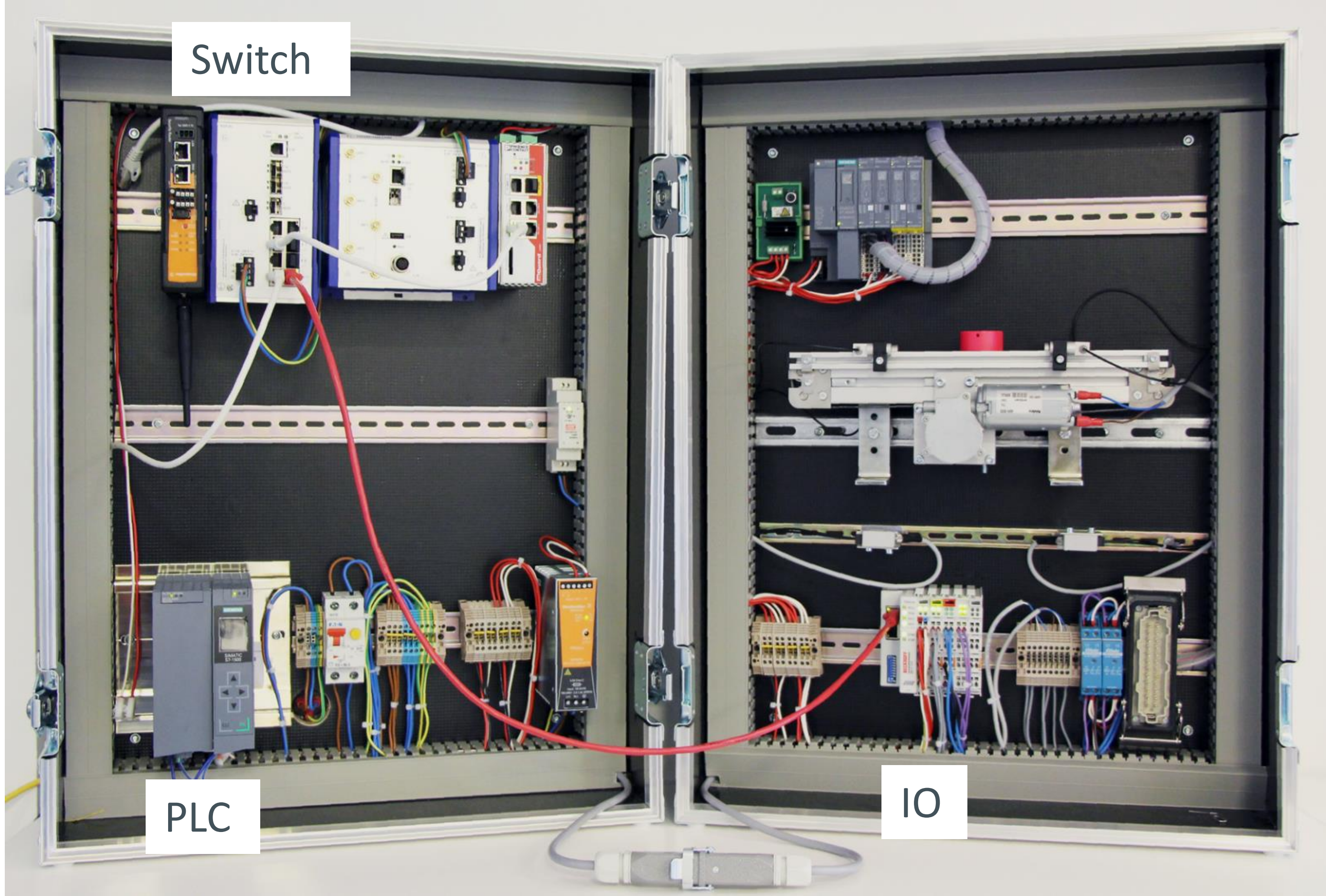
Data Frame



Merge, join, concatenate and compare

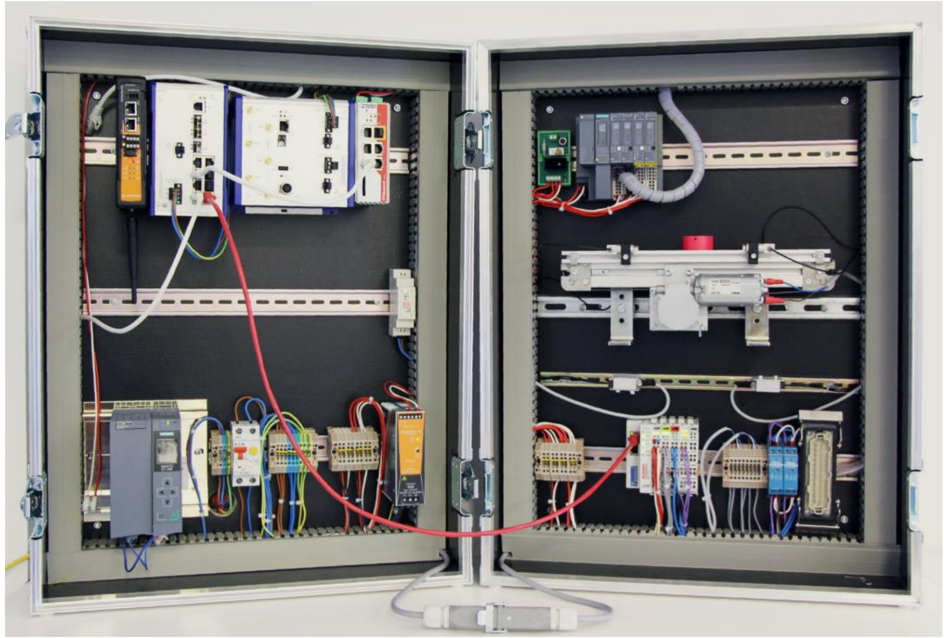


Excurse: Deep Packet Inspection



Excurse: Deep Packet Inspection

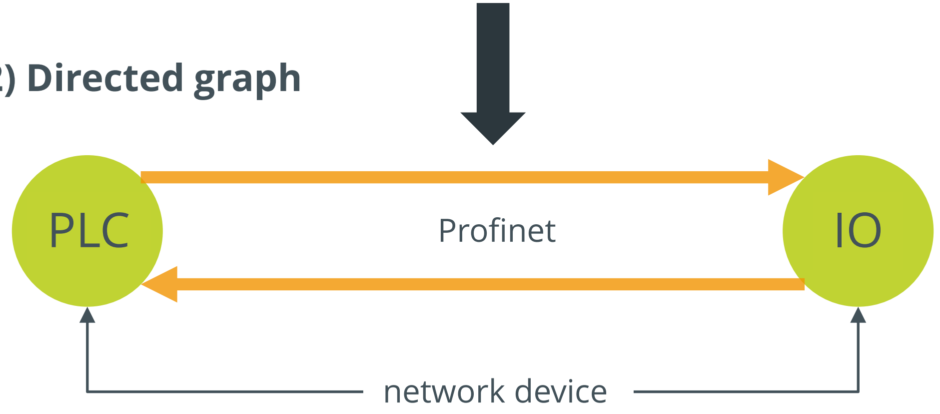
Test environment



(1) Constructing a directed graph from network packets

Source	Destination	Protocol	Process data	Time stamp	Attack
PLC	IO	Profinet	Engine On (1)	16:00:00	?
IO	PLC	Profinet	Sensor Off (2)	16:00:10	?
…					

(2) Directed graph

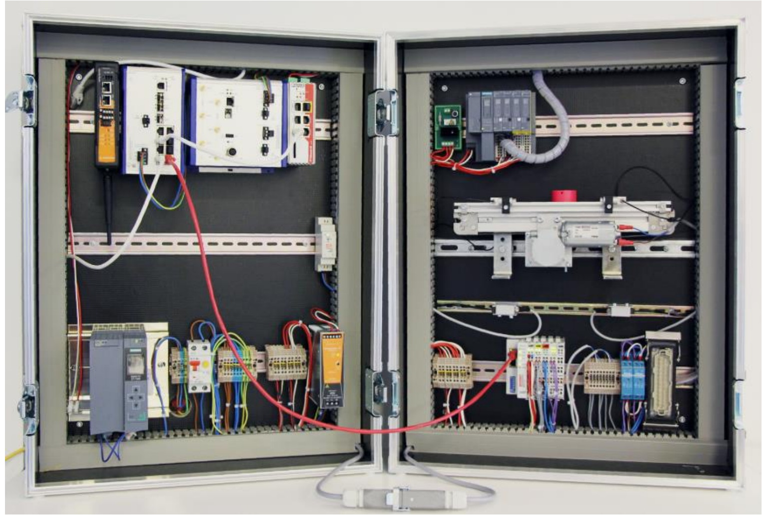


Excurse: Deep Packet Inspection

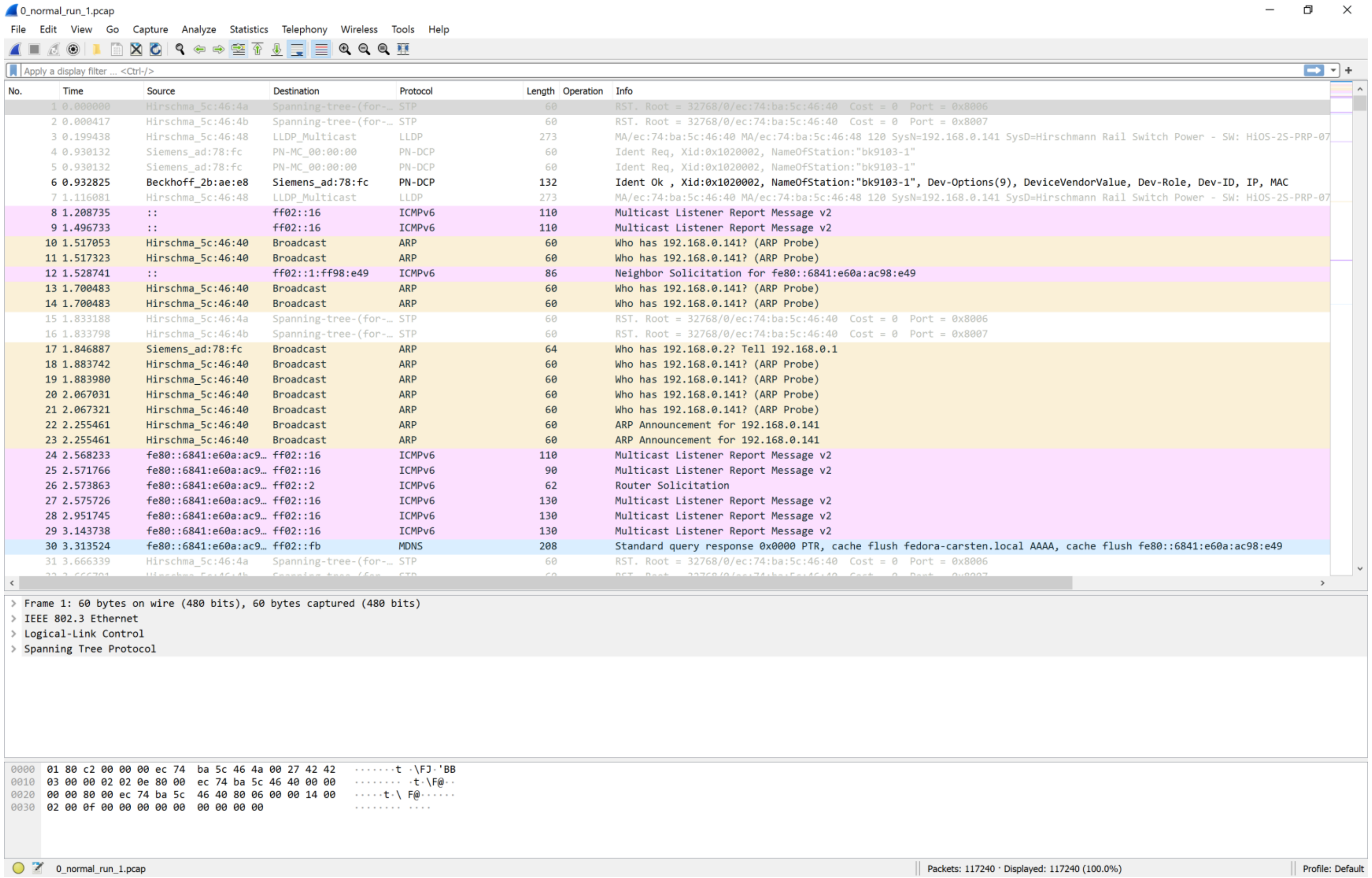
7 test scenarios

No.	Name	Anomaly
0	**normal**	**false**
1	dos_icmp_io_device	true
2	pn_replay_io_device	true
3	**pn_scan**	**true**
4	tcp_syn_scan	true
5	manipulated_plc_software	true
6	arp_cache_poisoning	true

Test environment



Excurse: Deep Packet Inspection



Excurse: Deep Packet Inspection

Pcap

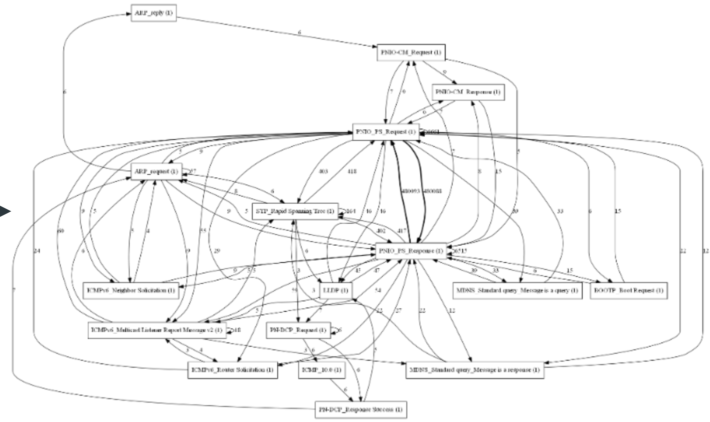
(1) Attack
(2) Normal

Event log

case_id: enumerated scenario
activity: what action is performed
Start_time: when action started
Ressource: who started the action

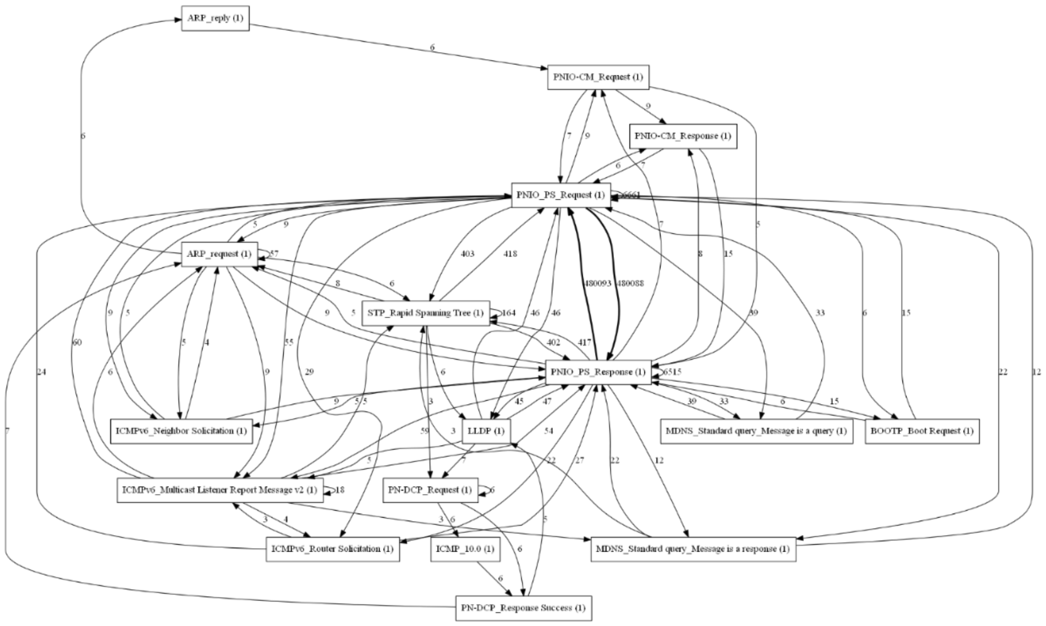
Action (e.g. request / response)

pm4py

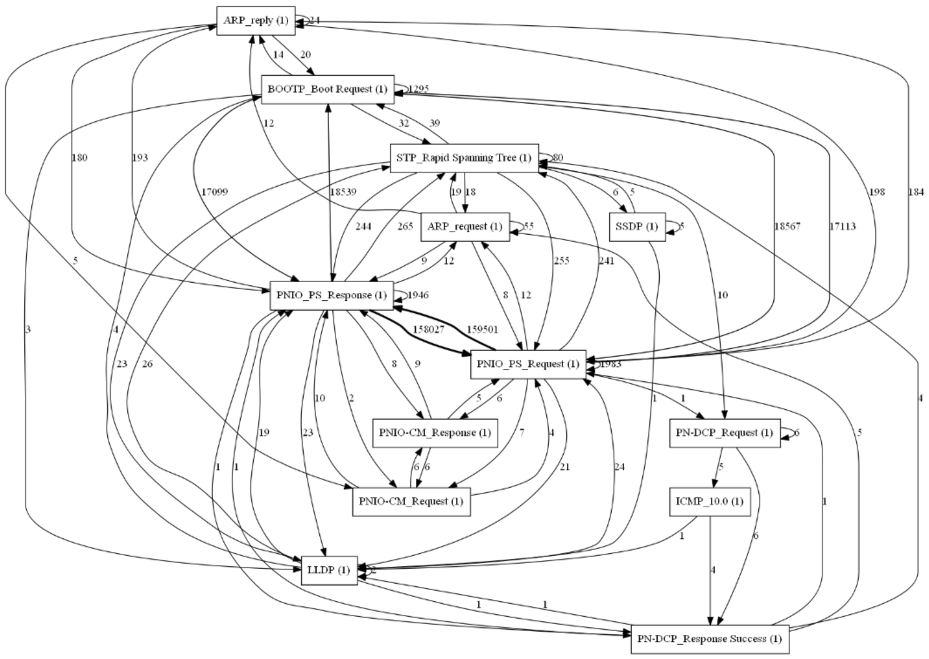


Excurse: Deep Packet Inspection

Normal



ARP CP



TECHNISCHE HOCHSCHULE
OSTWESTFALEN-LIPPE
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS

Thank you!