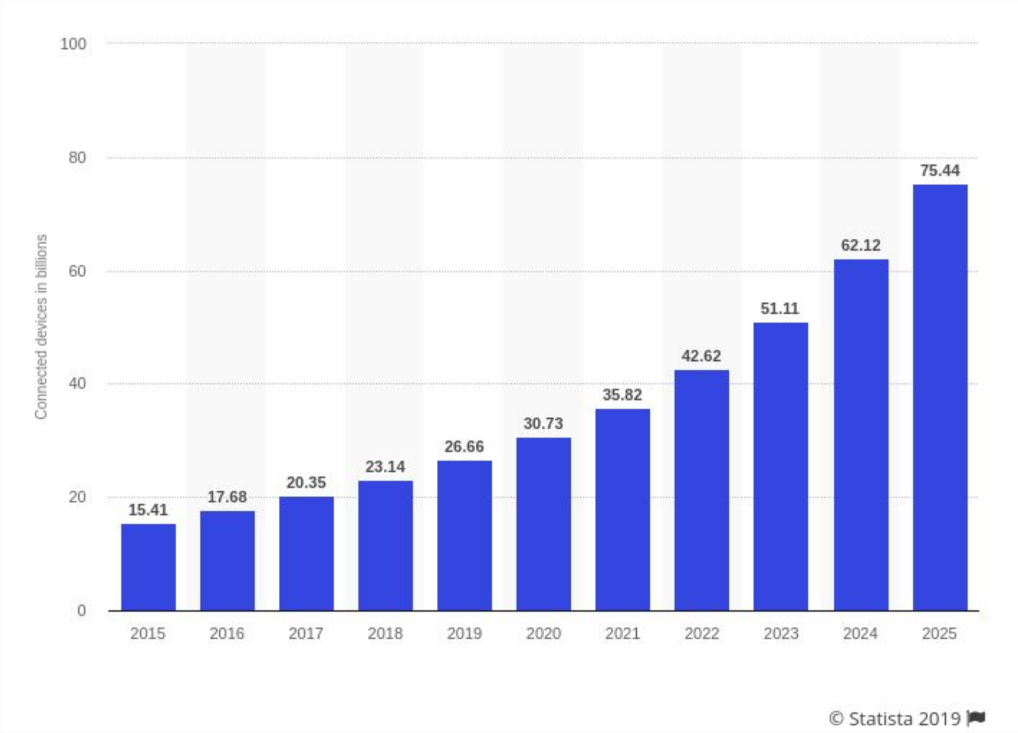
Machine Learning Techinques for Intrusion Detection in Internet Of Things Networks

Master Thesis - Federico Bacci

Why?

Internet of Things (IoT) connected devices installed base worldwide (in billions)



More than

70 000 000 000

IOT devices In 2025

Estimated IOT Market Share

3 trillions $

In 2026

(Business Insider Report)

Definiton

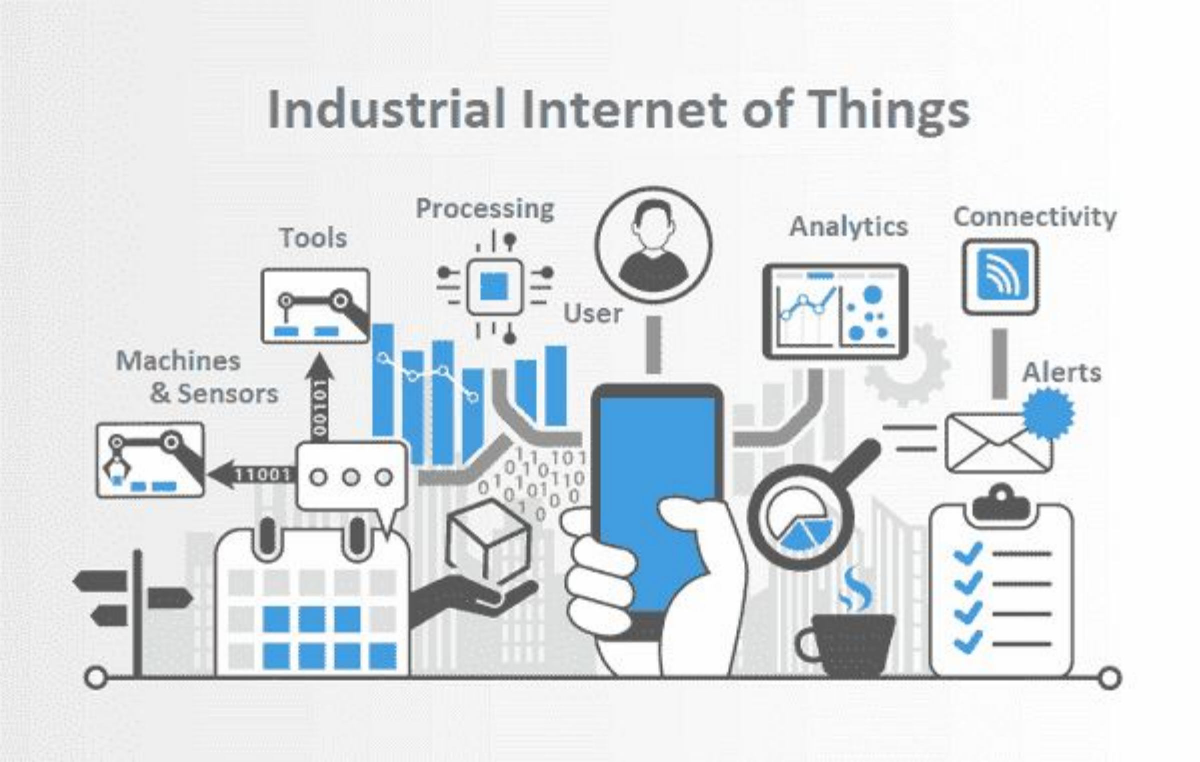
“The Internet of things (IoT) is the extension of Internet connectivity into physical devices and everyday objects.”

IOT Application

- Smart Home Devices (fridges, thermostats, lighting)
- Wearebles (smartbands, smartwatches)
- Smart Cars
- Smart grids (Tesla)
- Smart Cities

What if we use Internet of Things in the Industry?

Industry 4.0


Industrial Internet of Things
Processing
Tools
User
Analytics
Connectivity
Machines
& Sensors
Alerts
10100
11001

Industry 4.0

IoT intelligent systems enables:

- Rapid manufacturing of new products
- Dynamic response to product demands
- Real time optimization of manufacturing
- Predictive maintenance
- Statistical evaluation
- Maximize the reliability

Main Challenges

- **Security**
- Privacy
- Trust

Security Threats

- Wireless Threats
  - Eavesdropping (overhear information)
  - Data Alteration
  - Identity Theft
- **Routing Threats**
  - Spoofing
  - Selective Forwarding
  - Sinkhole Attacks
  - Sybil Attacks
  - **Wormholes**
  - Hello flood
  - Ackowledgement
- Denial of Service

Main concept

Can we use Machine Learning Techinques for Detect an Intrusion in Industrial Internet Of Things Networks?

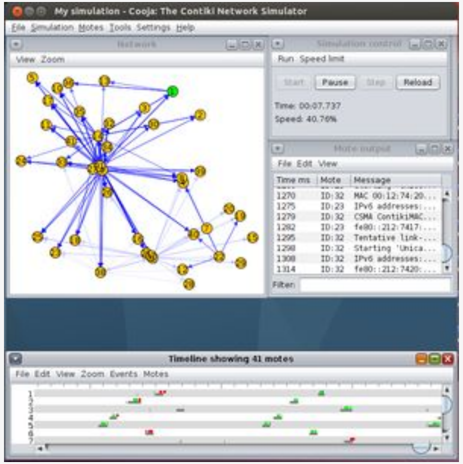
How ?

1. Use case
2. Data collection
3. Data Analysis
4. Machine Learning
5. Results

Use Case and Data creation

Real Use case:
IOT-LAB

Simulated Use case:
Cooja Simulator



Cooja Simulator

Advantages:

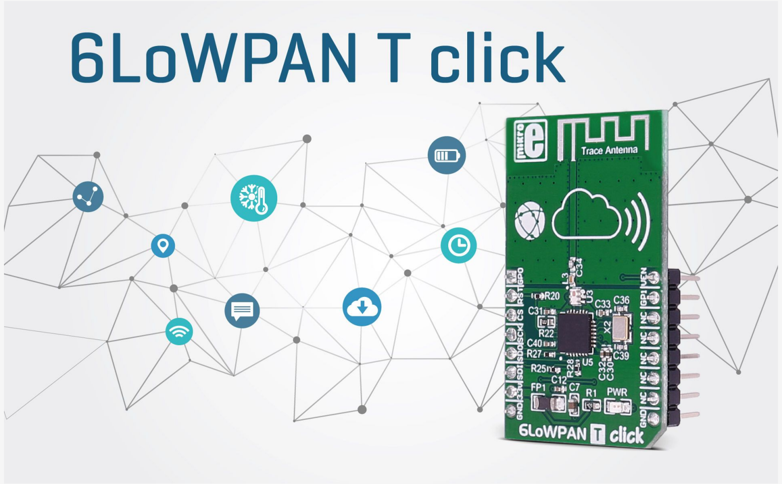
- Memory Allocation
- Full IP Networking
- Power Awareness
- 6lowPan, RPL

Cooja Simulator

Nodes and Router firmware created in the University of Patras and University of Athens

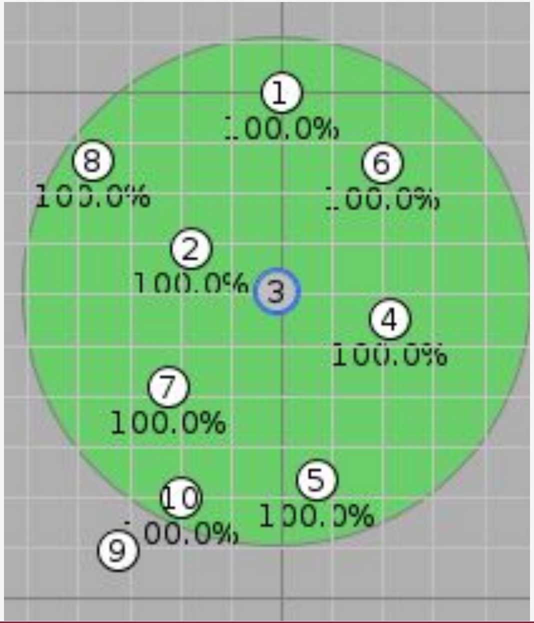
Network Specifications

- 9 or 16 nodes
- 6LowPAN-IPV6 (IEEE 802.15.4)
- RPL with DODAG root
- ICMPv6 Control Messages

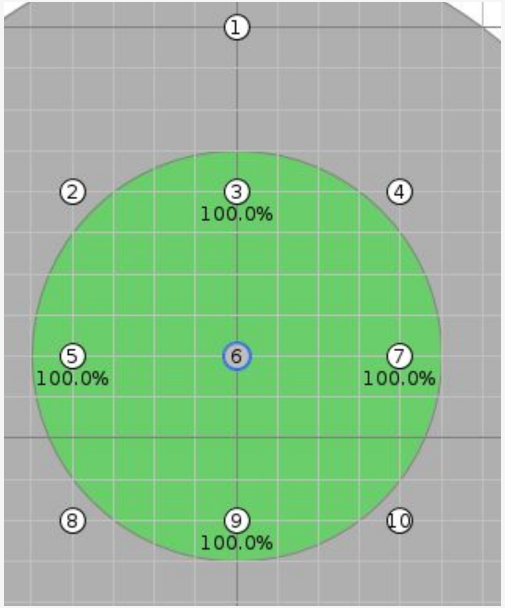


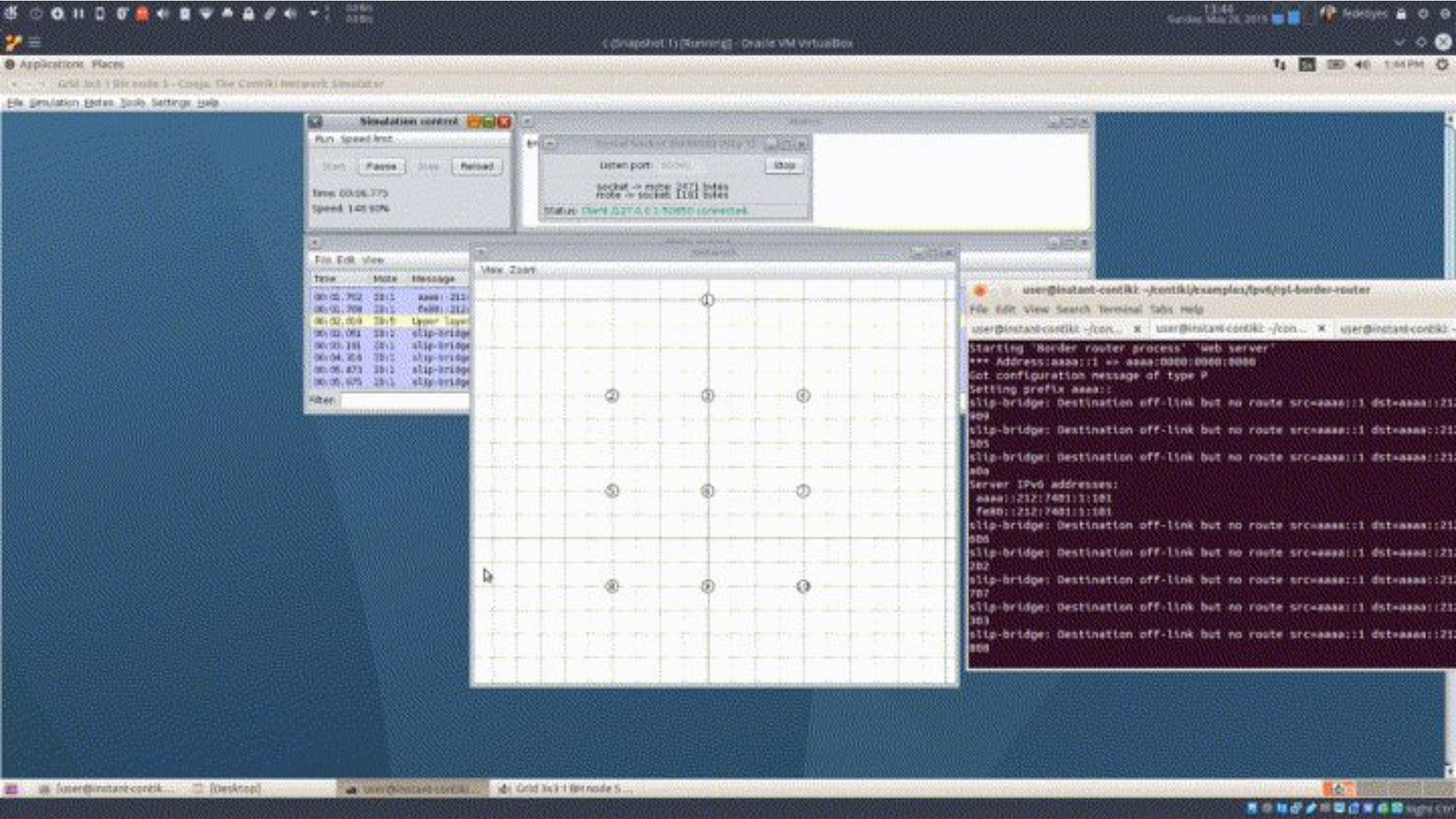
Network Specifications

Random Grid



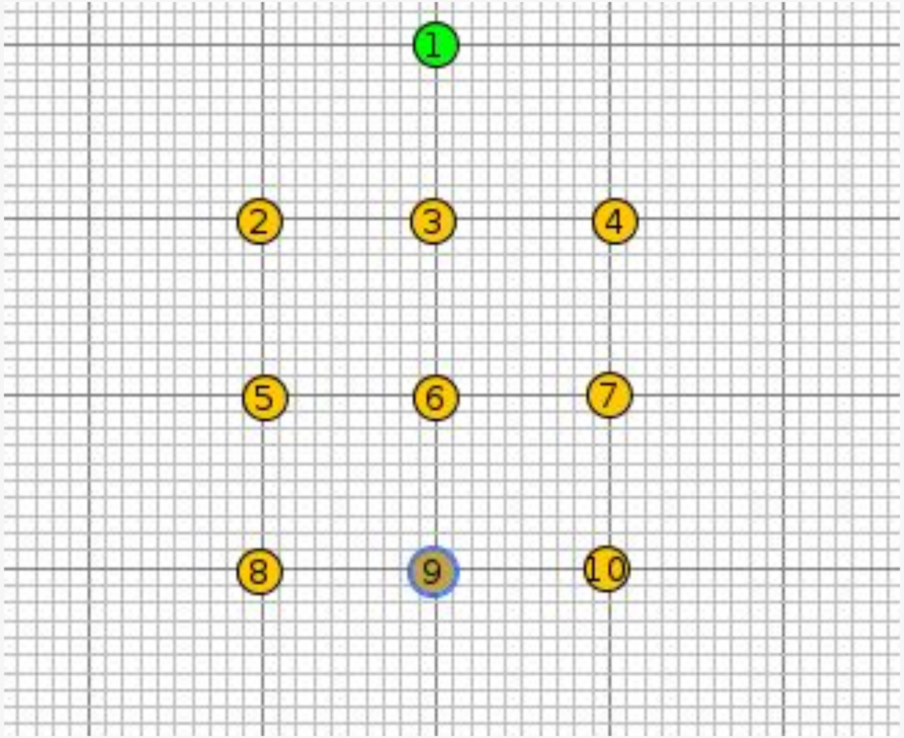
Square Grid

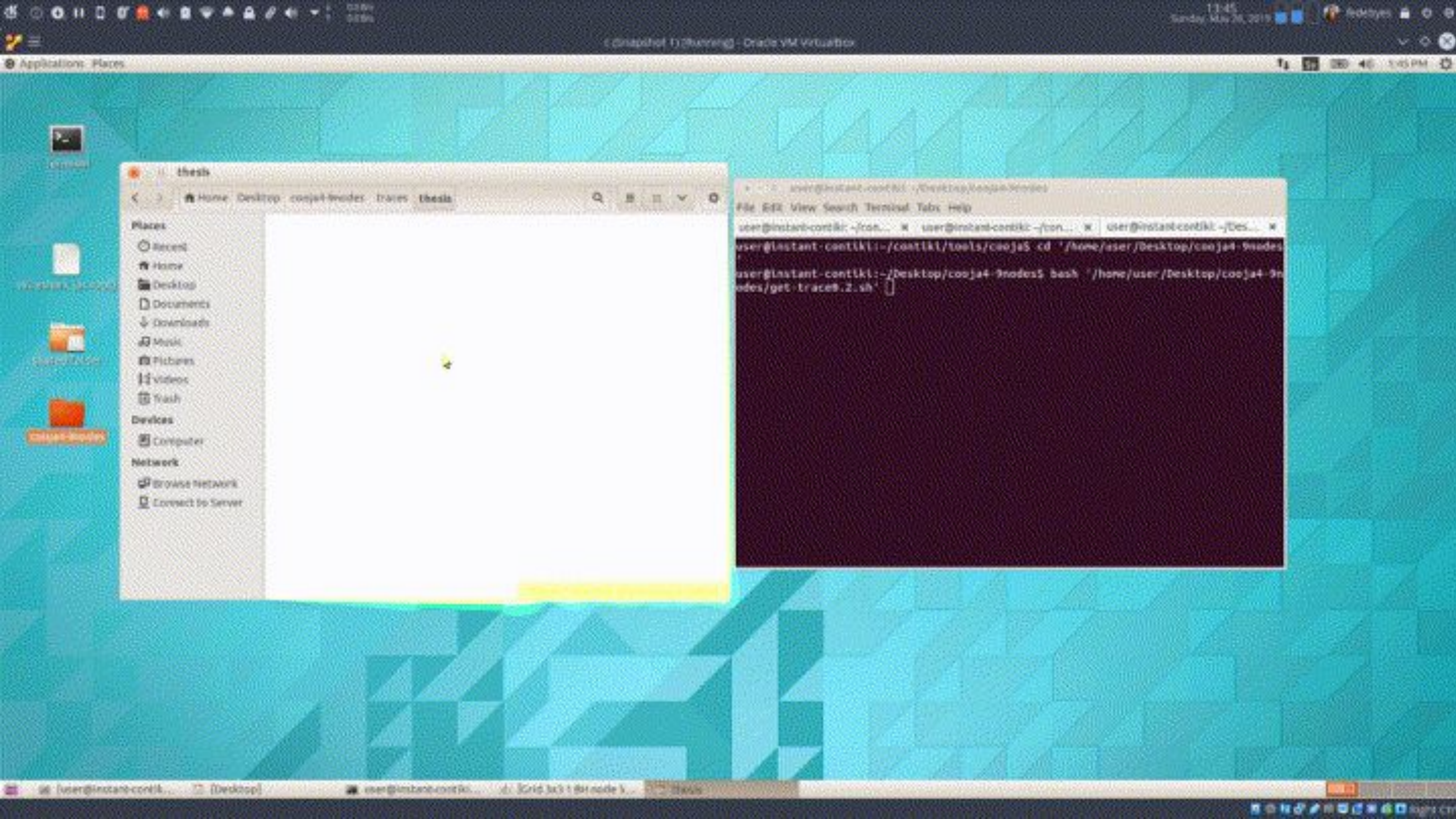



Applications Places
File Simulation Motes Tools Settings Help
Simulation control
Run Speed limit
Start Pause Step Reload
Listen port
Stop
File Edit View
Time Mote Message
Filter
View Zoom
user@instant-contiki: ~/contiki/examples/ipv6/rpl-border-router
File Edit View Search Terminal Tabs Help
Starting 'Border router process' 'Web server'
*** Address:aaaa::1 => aaaa:0000:0000:0000
Got configuration message of type P
Setting prefix aaaa::
slip-bridge: Destination off-link but no route src=aaaa::1 dst=aaaa::21
slip-bridge: Destination off-link but no route src=aaaa::1 dst=aaaa::21
slip-bridge: Destination off-link but no route src=aaaa::1 dst=aaaa::21
Server IPv6 addresses:
aaaa::212:7401:1:101
fe80::212:7401:1:101
slip-bridge: Destination off-link but no route src=aaaa::1 dst=aaaa::21
slip-bridge: Destination off-link but no route src=aaaa::1 dst=aaaa::21
slip-bridge: Destination off-link but no route src=aaaa::1 dst=aaaa::21
slip-bridge: Destination off-link but no route src=aaaa::1 dst=aaaa::21
slip-bridge: Destination off-link but no route src=aaaa::1 dst=aaaa::21

Cooja Simulator

- Router
- Black Hole
- Gray Hole (30, 50, 70)
- White Hole




Applications Places
thesis
Home Desktop cooja4-9nodes traces thesis
Places
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Devices
Computer
Network
Browse Network
Connect to Server
File Edit View Search Terminal Tabs Help
user@instant-contiki:~/contiki/tools/cooja$ cd '/home/user/Desktop/cooja4-9nodes
user@instant-contiki:~/Desktop/cooja4-9nodes$ bash '/home/user/Desktop/cooja4-9n
odes/get-trace0.2.sh'

Data analysis

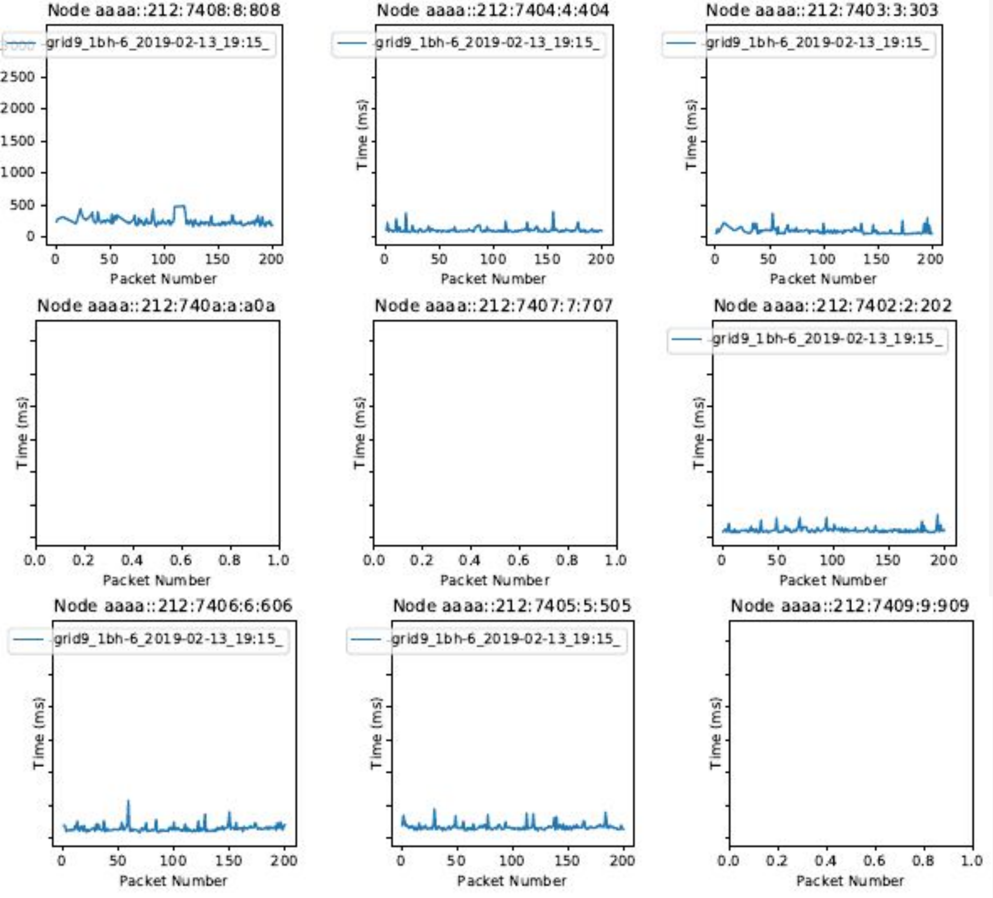
1. Find log files
2. Load ICMP
3. Create statistics

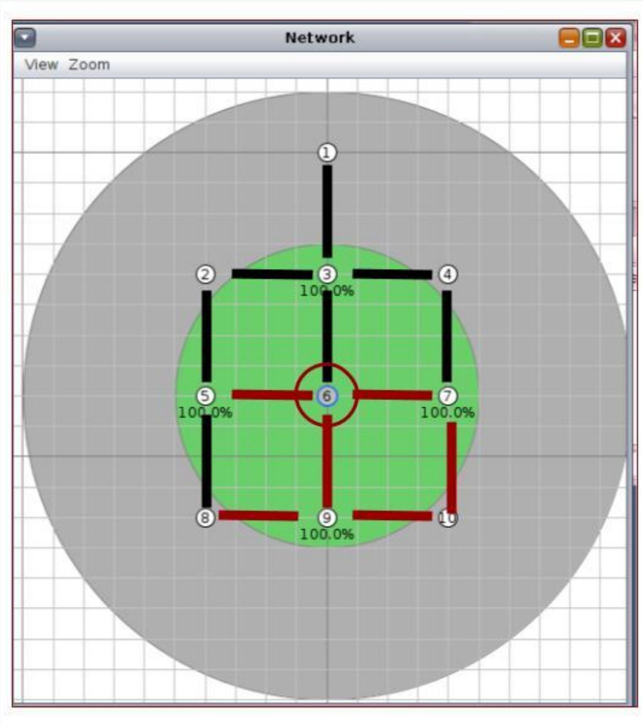
```python
def import_nodes_Cooja_2(directory,tracemask,node_defaults):

  files = []

  # load all files and extract IPs of nodes
  for file in os.listdir(directory):
      try:
          if file.startswith(tracemask) and file.index("routes"):
              continue
      except:
          files.append(file)

  nodes = pd.DataFrame(columns=['node_id', 'rank'])
  packets_node = {}

  # Load the ICMP traces
  for file in files:
      packets = pd.read_csv(directory + '/' + file,
                            sep=' |icmp_seq=|ttl=|time=',
                            na_filter=True,
                            header=None,
                            skiprows=1,
                            skipfooter=4,
                            usecols=[3, 5, 7, 9],
                            names=['node_id', 'seq', 'hop', 'rtt'],
                            engine='python').dropna().drop_duplicates()

      if len(packets) < 1:
          # Nodes affected by a black hole did not receive any packet
          node_id = file[-24:-4]
          if(node_id=="aa::212:7411:11:1111"): node_id="aaaa::212:7411:11:1111"
          packets = pd.DataFrame(columns=['node_id', 'seq', 'hop', 'rtt'],
                                 data=[[node_id, 1, node_defaults[node_id], 1]])

          nodes.loc[len(nodes)] = [file[-24:-4], node_defaults[node_id]]
          packets_node[file[-24:-4]] = packets

      else:
          #print("qui")
          packets['node_id'] = packets.apply(lambda row: row['node_id'][:-1], axis=1)

          packets = packets.sort_values(by=['node_id', 'seq'], ascending=True, na_position='first')
          packets = packets[packets['rtt'] > 1]
          packets["hop"]=  64-packets['hop']
          packets_node[packets['node_id'][0]] = packets

  nodes=nodes.sort_values(by=['rank', 'node_id'])

  #tranformation in node
  nodeList=[]

  for n in packets_node.keys():
      #print((packets_node[n]).head())
      pkts=packets_node[n].drop(["node_id","hop"],axis=1)
      #print(pkts)
      hop=int(packets_node[n]["hop"][0])
      ip=packets_node[n]["node_id"][0]
      #print(hop)
      n=node(ip,hop,pkts)
      nodeList.append(n)


  return nodeList
```

Data analysis

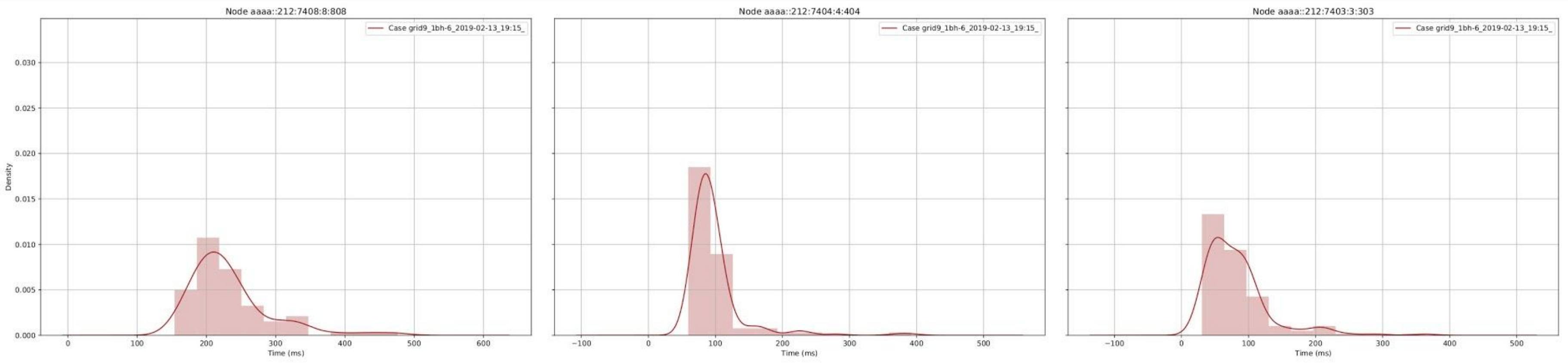
- Packet loss percentage
- Number of packets
- Standard deviation (rtt)
- Mean (rtt)
- Variance (rtt)
- Node hop
- Minimum (rtt)
- Maximum (rtt)
- Number of outliers
- Amplitude of the window

label	typ	type_coı	type_cor	pa	node	coun	std	mean	var	hop	o
grid_1bh-9_2019-02-20_00	BH	normal	normal	2	aaaa::212:7403:3:303	98	76.05565228	120.8887755	5784.462244	1	4
grid_1gh30-9_2019-02-20_	BH	normal	normal	3	aaaa::212:7402:2:202	97	102.0977565	152.8463918	10423.95189	2	3
grid_1bh-9_2019-02-20_00	BH	normal	normal	4	aaaa::212:7405:5:505	96	109.6936084	186.6666667	12032.68772	2	6
grid_1bh-9_2019-02-20_00	BH	normal	normal	4	aaaa::212:740e:e:e0e	96	104.7791155	383.3229167	10978.66305	5	4
grid_1bh-9_2019-02-20_00	BH	normal	normal	4	aaaa::212:7407:7:707	96	119.1651879	182.0729167	14200.342	2	3
grid_1gh50-9_2019-02-19_	BH	normal	normal	5	aaaa::212:740b:b:b0b	95	404.6223683	373.6842105	163719.2609	3	7
grid_1gh50-9_2019-02-19_	BH	normal	normal	5	aaaa::212:7404:4:404	95	81.87068502	98.68631579	6702.809066	1	4
grid_1gh70-7_2019-02-19_	BH	normal	normal	5	aaaa::212:740e:e:e0e	95	142.484334	399.6421053	20301.78544	5	6
grid_1bh-9_2019-02-20_00	BH	normal	normal	6	aaaa::212:740a:a:a0a	94	84.21884656	278.1170213	7092.814116	4	8
grid_1gh30-9_2019-02-20_	BH	normal	normal	6	aaaa::212:7403:3:303	94	30.98190749	75.20319149	959.8785919	1	3
grid_1bh-7_2019-02-19_22	norma	normal	normal	7	aaaa::212:740e:e:e0e	93	253.9215146	561.4086022	64476.13558	5	5
grid_1gh30-9_2019-02-20_	BH	normal	normal	7	aaaa::212:7404:4:404	93	83.04367269	93.59247312	6896.251573	1	4
grid_1gh30-9_2019-02-20_	BH	normal	normal	8	aaaa::212:740f:f:f0f	92	151.9781285	323.0108696	23097.35153	4	6
grid_1gh50-9_2019-02-19_	BH	normal	normal	8	aaaa::212:740e:e:e0e	92	216.0088642	418.1956522	46659.82943	5	4
grid_normal_2019-02-26_1	norma	normal	normal	8	aaaa::212:7403:3:303	92	124.7764051	143.0804348	15569.15126	1	2
grid_1bh-9_2019-02-20_00	BH	normal	normal	9	aaaa::212:7406:6:606	91	146.1388429	325.1648352	21356.56142	3	5
grid_1gh50-9_2019-02-19_	BH	normal	normal	9	aaaa::212:7407:7:707	91	248.2625506	304.9230769	61634.29402	2	5
grid_1bh-7_2019-02-19_22	BH	normal	normal	10	aaaa::212:740a:a:a0a	90	166.3555452	405.9666667	27674.16742	4	4
grid_1bh-9_2019-02-20_00	BH	normal	normal	11	aaaa::212:740f:f:f0f	89	170.8553386	388.4606742	29191.54673	4	3
grid_1gh70-7_2019-02-19_	BH	normal	normal	11	aaaa::212:7407:7:707	89	128.0426315	245.247191	16394.91547	2	4
grid_normal_2019-02-26_1	norma	normal	normal	11	aaaa::212:7406:6:606	89	348.924725	621.3033708	121748.4637	3	5
grid_normal_2019-02-26_1	norma	normal	normal	11	aaaa::212:7402:2:202	89	260.1215954	310.7752809	67663.24438	2	3
grid_1bh-7_2019-02-19_22	norma	normal	normal	12	aaaa::212:7402:2:202	88	160.0573535	187.4215909	25618.35643	2	4
grid_1bh-9_2019-02-20_00	BH	normal	normal	12	aaaa::212:7402:2:202	88	93.42733082	171.7727273	8728.666144	2	4
grid_1gh30-9_2019-02-20_	BH	normal	normal	12	aaaa::212:7406:6:606	88	82.53669508	224.625	6812.306034	3	5
grid_1bh-9_2019-02-20_00	BH	normal	normal	13	aaaa::212:740c:c:c0c	87	155.8397229	325.6551724	24286.01925	3	4
grid_normal_2019-02-26_1	norma	normal	normal	13	aaaa::212:7406:6:606	87	317.5891025	596.1034483	100862.838	3	4
grid_normal_2019-02-26_1	norma	normal	normal	13	aaaa::212:7403:3:303	87	119.0138971	183.8793103	14164.30771	1	4
grid_normal_2019-02-26_1	norma	normal	normal	13	aaaa::212:740e:e:e0e	87	197.3916663	625.1724138	38963.46993	5	4
grid_normal_2019-02-26_1	norma	normal	normal	13	aaaa::212:740a:a:a0a	87	278.40635	509.2988506	77510.0957	4	5
grid_1bh-9_2019-02-20_00	BH	normal	normal	15	aaaa::212:7404:4:404	85	46.26066597	104.9352941	2140.049216	1	4
grid_1gh70-7_2019-02-19_	BH	normal	normal	15	aaaa::212:7405:5:505	85	170.1480567	246.4894118	28950.3612	2	4


Node aaaa::212:7408:8:808
Node aaaa::212:7404:4:404
Node aaaa::212:7403:3:303
Node aaaa::212:740a:a:a0a
Node aaaa::212:7407:7:707
Node aaaa::212:7402:2:202
Node aaaa::212:7406:6:606
Node aaaa::212:7405:5:505
Node aaaa::212:7409:9:909
grid9_1bh-6_2019-02-13_19:15_
Time (ms)
Packet Number


Network
View Zoom
1
2
3
4
5
6
7
8
9
10
100.0%

Data analysis



Machine Learning Techniques

Supervised Learning

- Random Forest
- KNN
- SVM

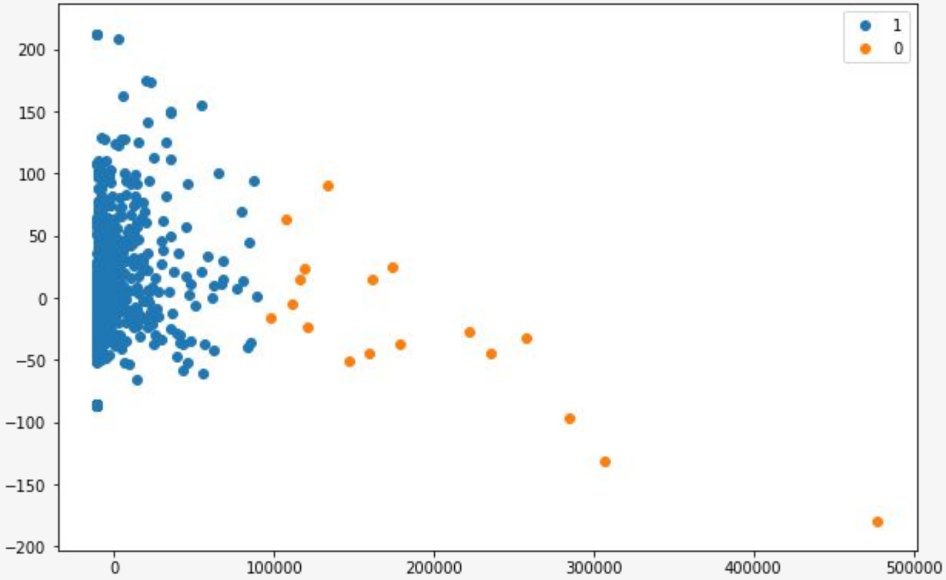
Unsupervised Learning

- K-Means

K-Means

```
randomly chose k examples as initial centroids
while true:
   create k clusters by assigning each
       example to closest centroid
   compute k new centroids by averaging
       examples in each cluster
   if centroids don't change:
       break
```

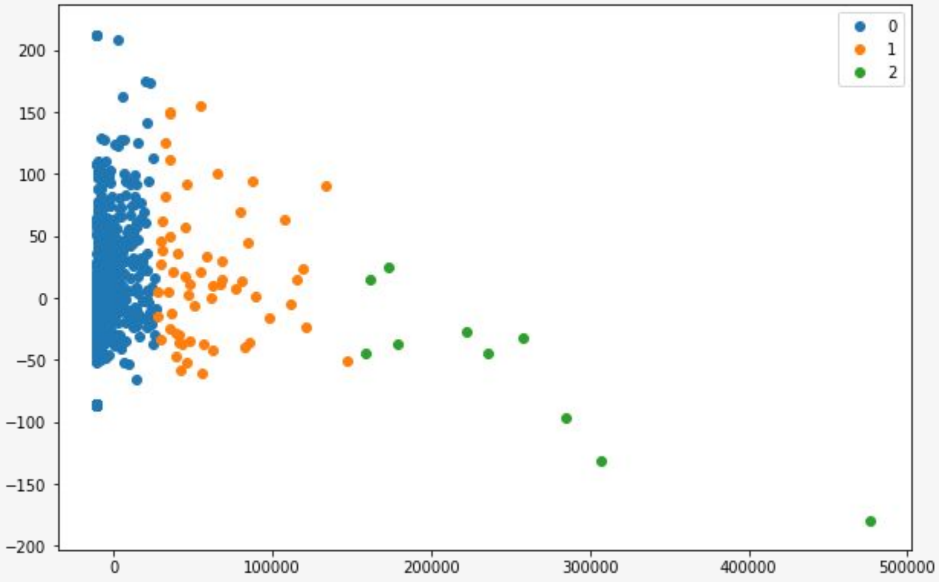
Results Unsupervised Learning



2 Cluster division by Kmeans

Blue: Normal Behaviour
Orange: Affected by a Malicious Node

Results Unsupervised Learning



3 Cluster division by Kmeans

Blue: Normal Behaving Node
Orange: Affected from Gray Hole
Green: Affected from Black Hole

Accuracy Supervised Learning

Machine Learning Tecnique	Mean Accuracy
Random Forest	99%
SVM	87%
KNN	84.2%

Conclusion

Can we use Machine Learning Techinques for Detect an Intrusion in Industrial Internet Of Things Networks?

Yes!

Future Work

- Try in a real world Industry 4.0
- Different types of attacks

Thanks

Questions

Federico Bacci

Master Research Thesis in
Engineering in Computer Science

github.com/fedebyes/iot-netprofiler

fedeb703@gmail.com

Machine Learning Techinques for Intrusion Detection in Internet Of Things Networks

Master Thesis - Federico Bacci