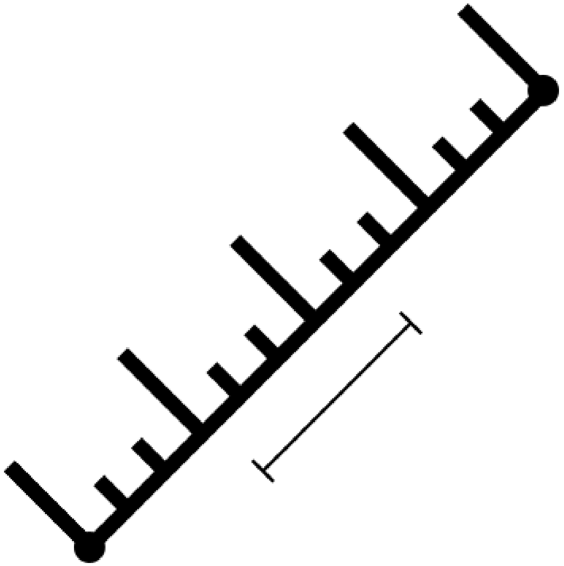
Measurements and analysis in high-speed software networks

Or... the ***Data Uncertainty Principle***

Leonardo Linguaglossa | *ANR 2023-2027*

Measurements are important

- Analyze a system
- Perform predictions
- Evaluate performance
- Detect anomalies
- Optimize resource usage
- …



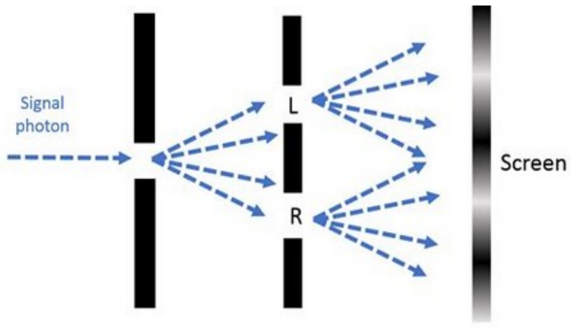
Any analysis is as good as the experimental observations

In physics, at very low level

- Fundamental limit of measurement accuracy of natural systems
- Complementary variables, Heisenberg inequality -> **native property**
- Inherent to all wave-like systems

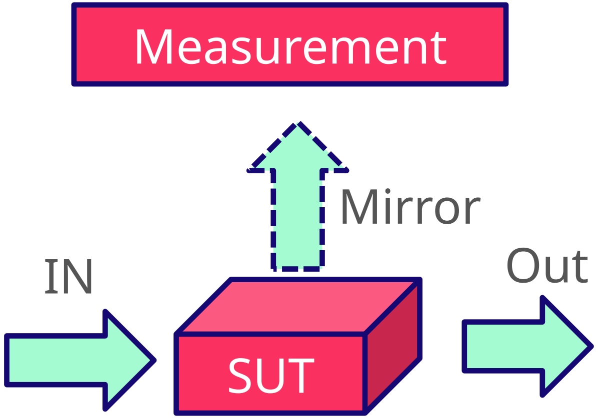
In physics, the observer effect

- A measured system is altered by the measurement itself
- It can be mitigated by technology or differential measurements
- Inherent to all *macro*scopic systems -> **behavioral alteration**
- Does not set a fundamental limit of the measurement that can be done



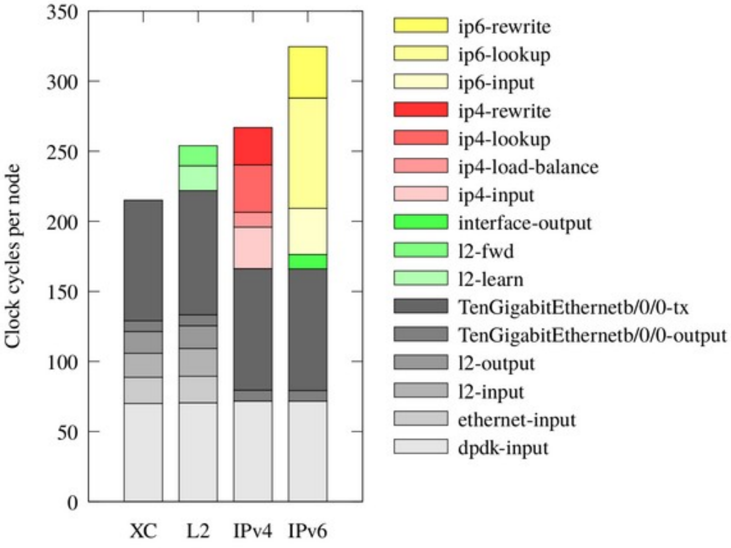
In networking

- Traffic measurements
- Monitoring performed directly on network devices
- Active or passive
- Direct or indirect
- With or without mirroring
- Deep (per-packet) or sampling (poisson, uniform)
- **Use cases**: anomaly detection, resource allocation, performance enhancement



In software networking

- Same as before, but all calculations are performed by one (or more) CPU(s)
- All « functions » are implemented as pieces of code… see the problem?



Bonus: **open the white-box!**

Fine-grained data previously not accessible

The IONOS project

- Measurement problem in high-speed software network: uncertainty/observer effects
- Exploratory project:
  - Limits of the uncertainty principle
  - Design of non-invasive measurement techniques

AGENDA

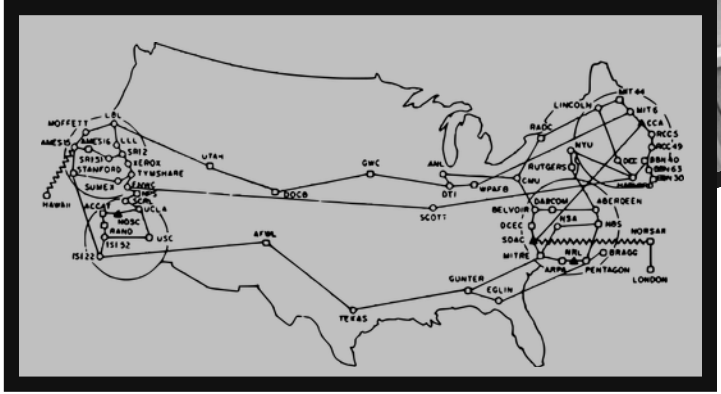
- High-speed software networks, COTS hardware and ML-enhanced functions
- Methodology: inference of VNF state using indirect non-invasive measurements
- Use cases and early results

Part I

Network softwarization

Evolution of network systems

Functionality tightly coupled with size



Topology tightly coupled with geography

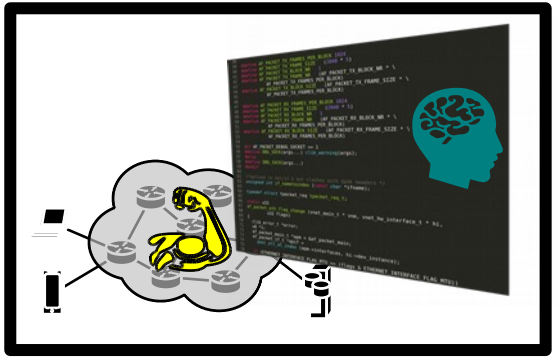
Maintenance tightly coupled with human interaction

Towards a steady softwarization of networks

- Virtualization: Hardware/software disaggregation
- NFV/SDN: Component/Function decoupling
- Automation : human/machine separation of tasks

Virtualization of devices, services, topologies, ...

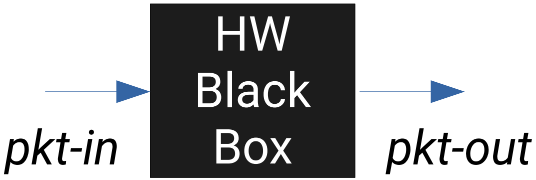
Miniaturization of network devices with same functionality

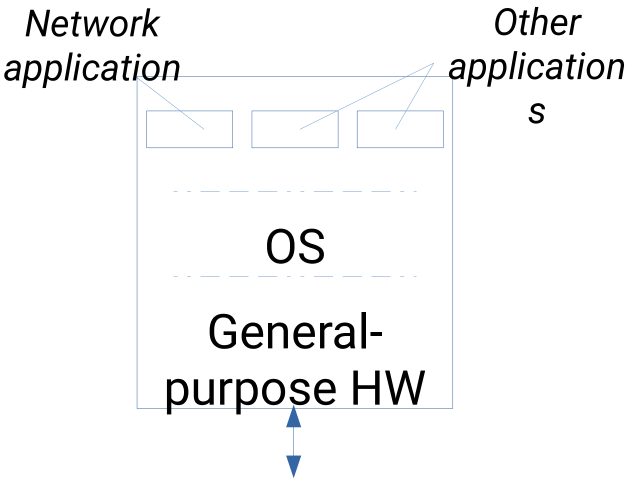


Automation and reduction of human intervention

Replacing middleboxes with SW equivalents

Software-based networking tradeoff
HW performance vs SW flexibility



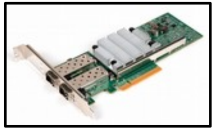


Acceleration techniques: reducing the HW/SW gap and provide high-speed

Sample function: software routing (L2/L3 forwarding)

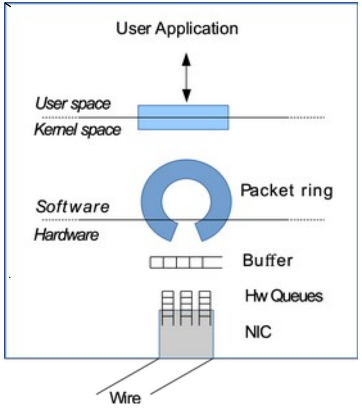
Commodity server (multicore)

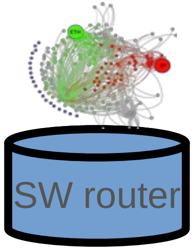
NIC

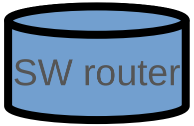


Wire + transceiver

Multi 10-Gbit per second packet processing capabilities







Program

```
While(true):
  batch = get_pkts(NIC)
  if (size(batch) > 0):
    do_processing(batch)
   continue
```

Assembler (ASM)

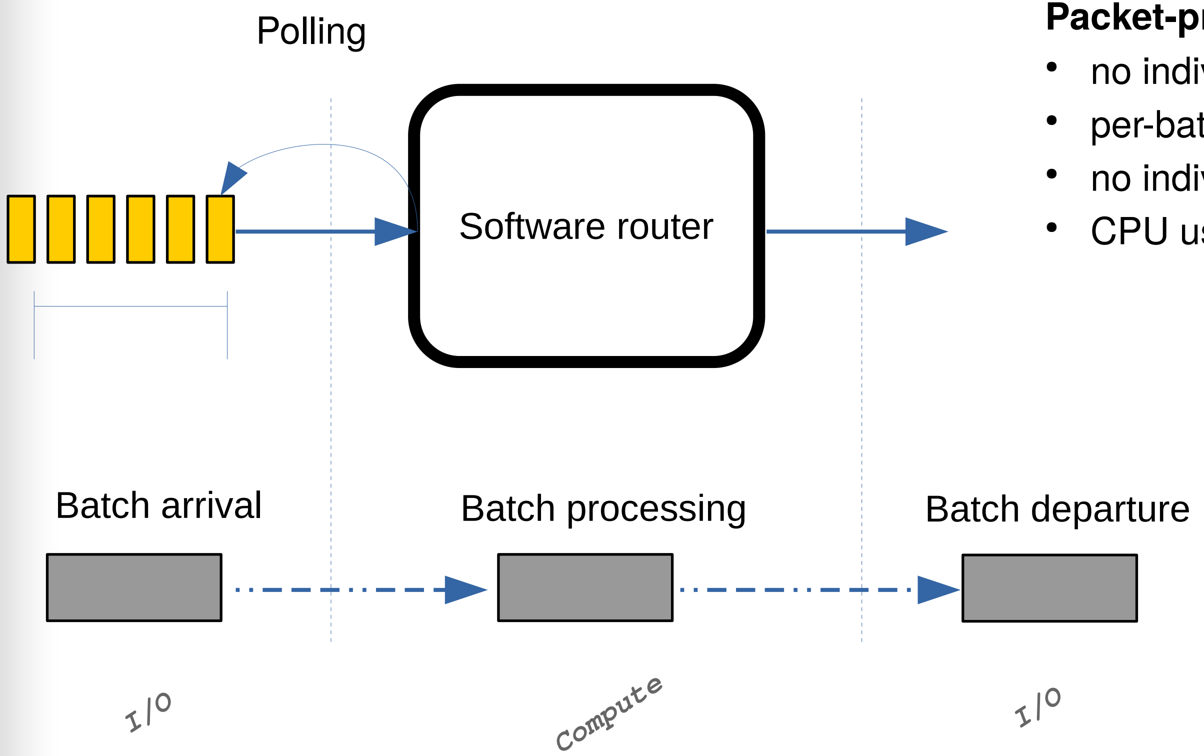
```
get_pkts:
                INSTR_1
                INSTR_2
                …
                INSTR_n

do_processing:
                INSTR_1
                …
                INSTR_m
```

Keeping up with speed: software acceleration techniques

	Poll	I/O Batch	Memory					Compute Batch	Threading		Coding		NIC-support			CPU-support	
			ZC	MP	HP	PF	CA		LFMT	LT	ML	BP	RSS	FH	SR-IOV	SIMD	DDIO
Reduce memory access			✓		✓	✓	✓										✓
Optimize memory allocation				✓	✓		✓										
Share overhead of processing								✓			✓						
Reduce interrupt pressure	✓	✓															
Horizontal scaling									✓	✓			✓		✓		
Exploit CPU cache locality							✓	✓	✓								✓
Reduce CPU context switches	✓	✓							✓	✓							
Fill CPU pipeline								✓			✓	✓				✓	
Exploit HW computation														✓	✓	✓	✓
Simplify thread scheduling	✓									✓							

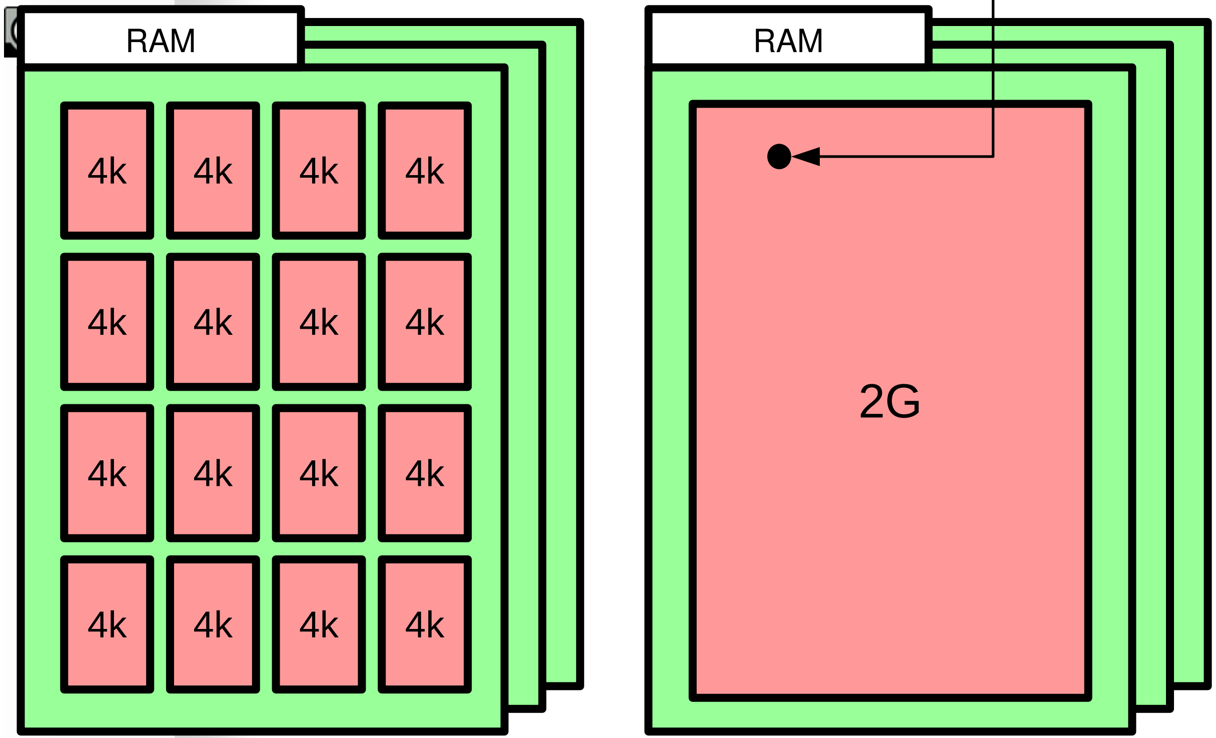
Acceleration technique: batching and polling

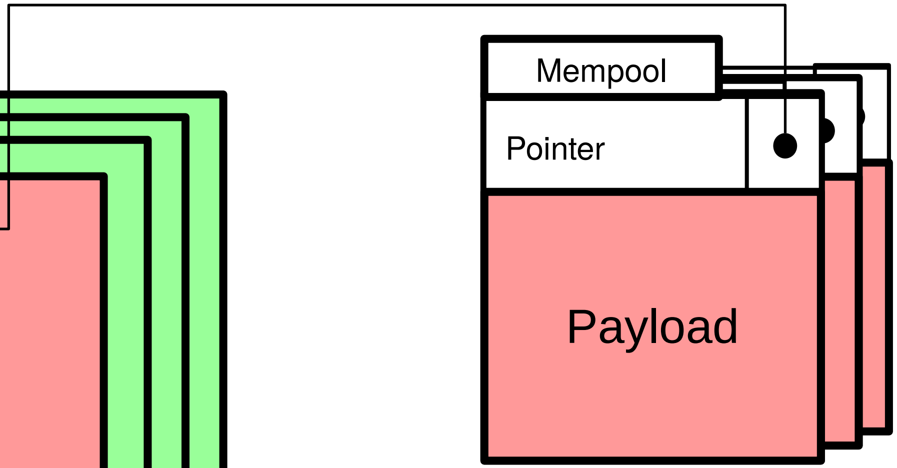


Packet-processing:

- no individual packet arrivals
- per-batch processing
- no individual packet departure
- CPU usage is optimized

Acceleration technique: memory management





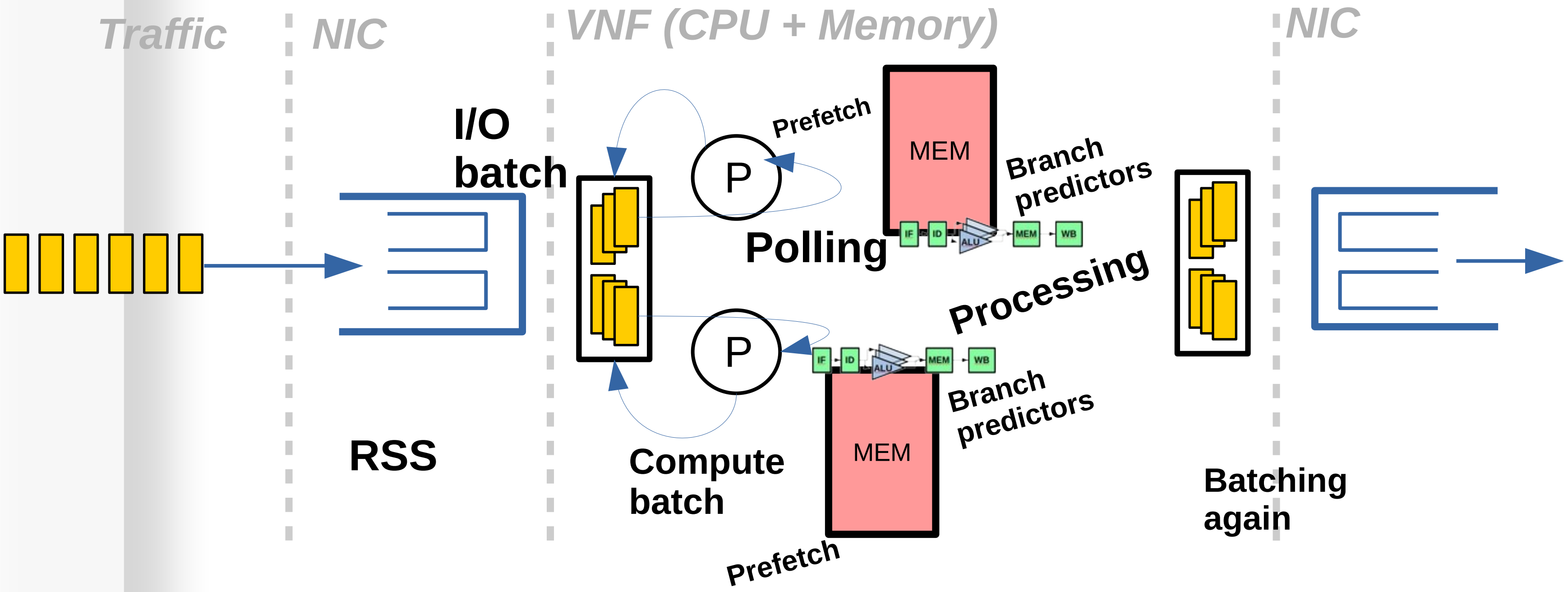
Hugepages

- Limiting page faults

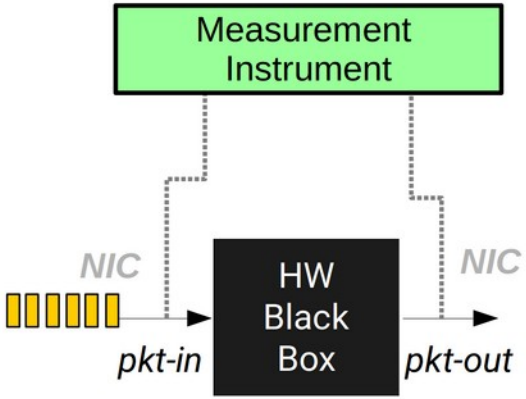
Mempool

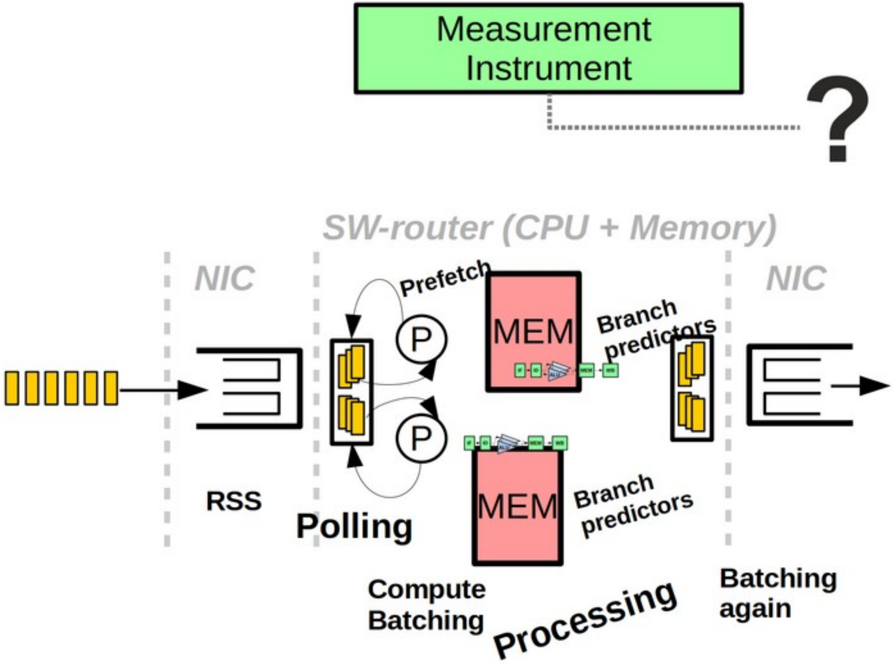
- Avoid run-time `malloc`

A simple model of a SW router

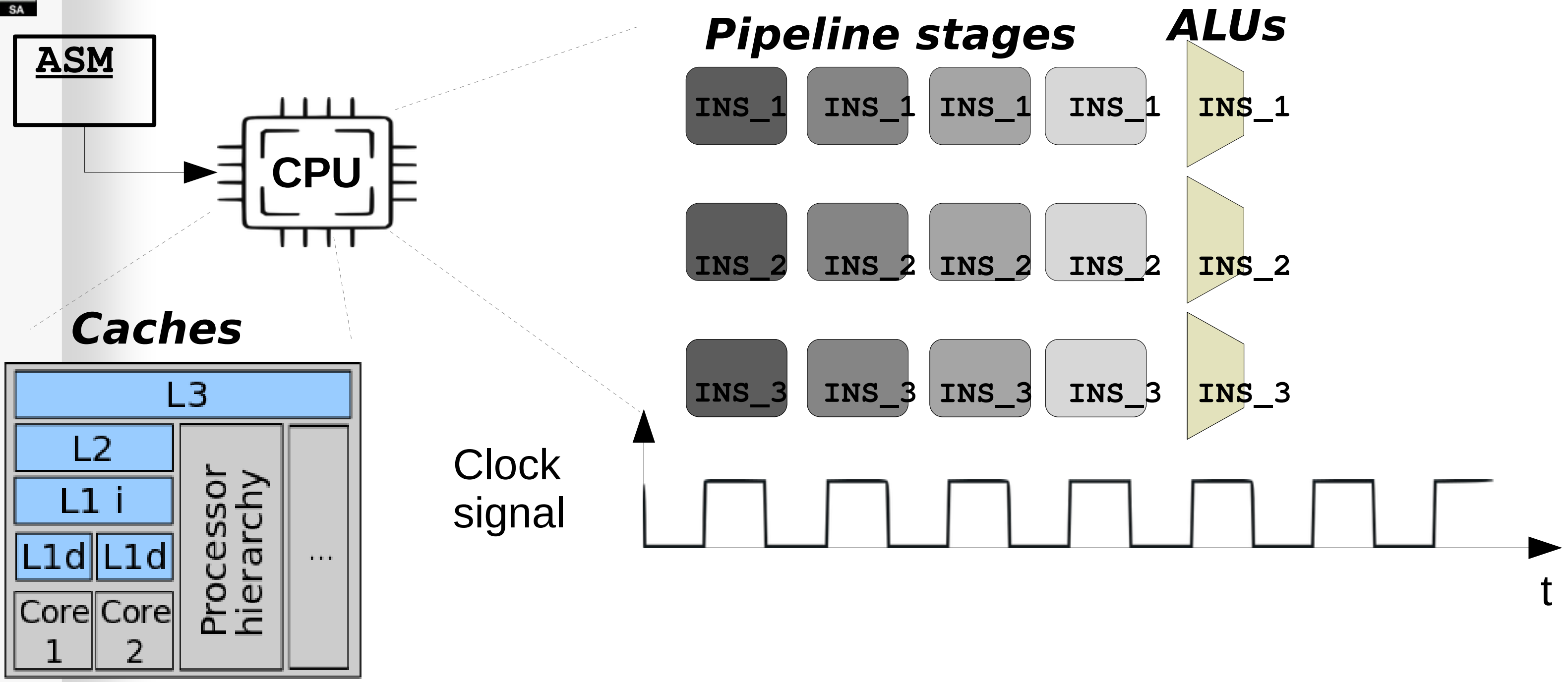


The measurement problem

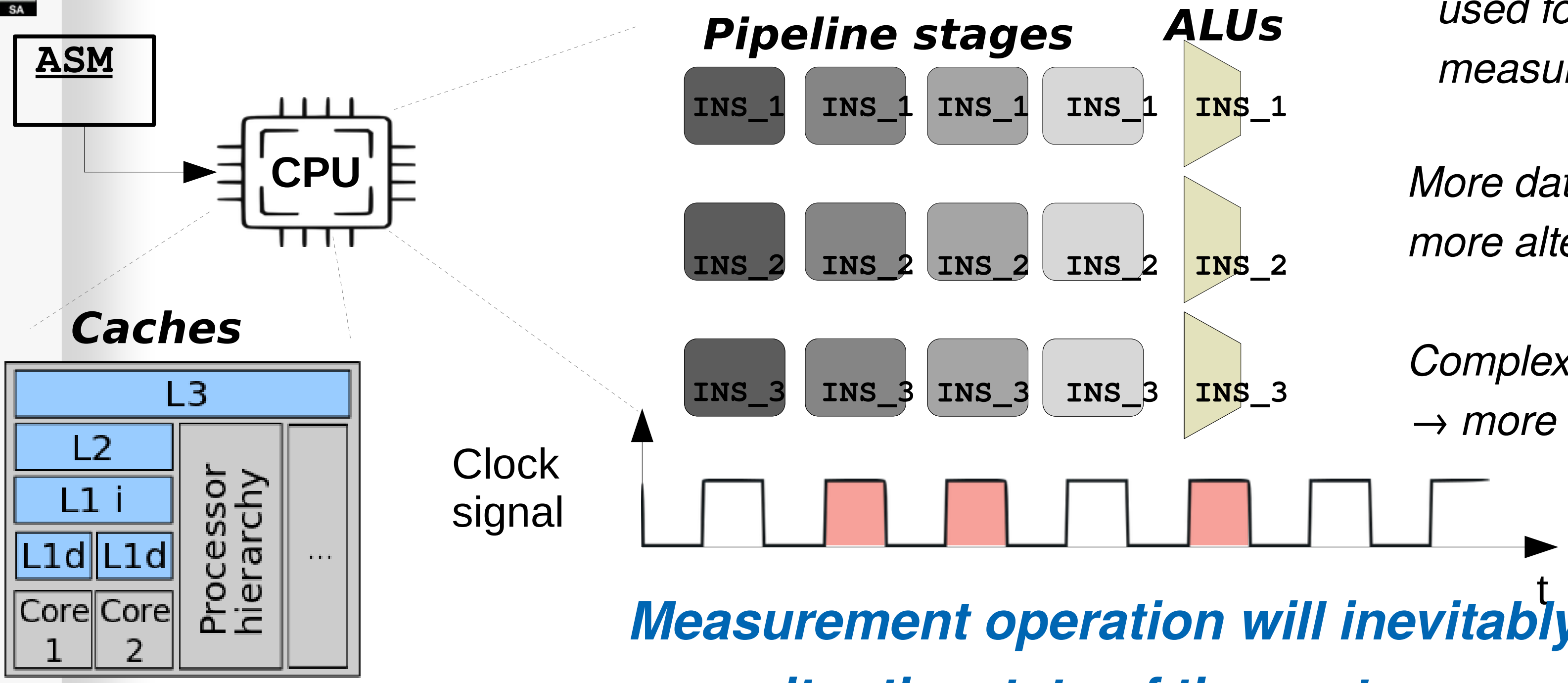




How to collect network data



How to collect network data



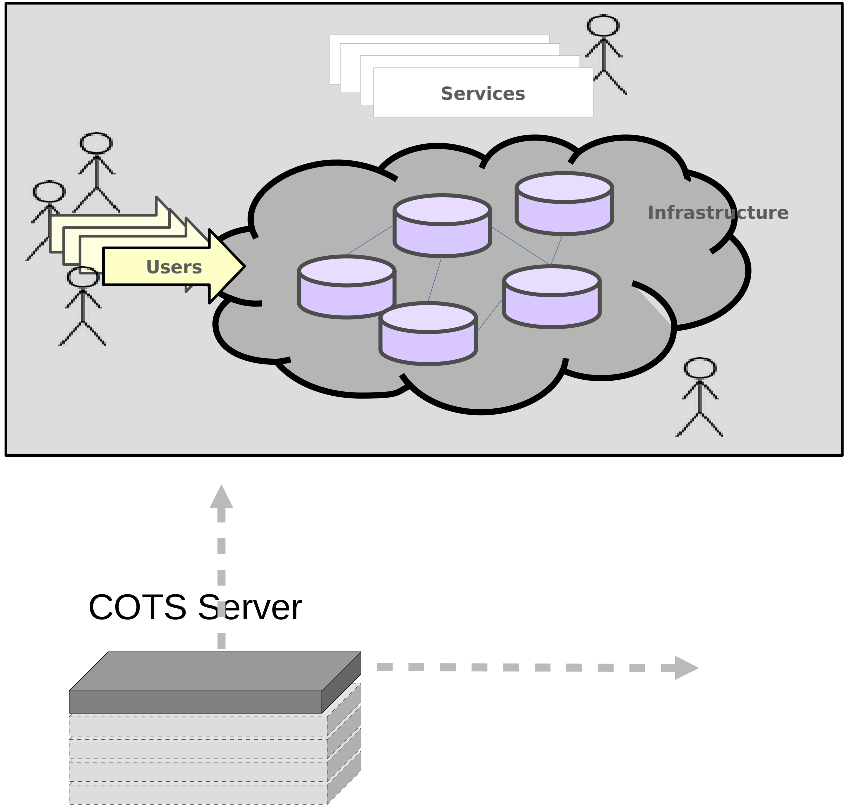
(Clock cycles used for measurement)

More data → more altering

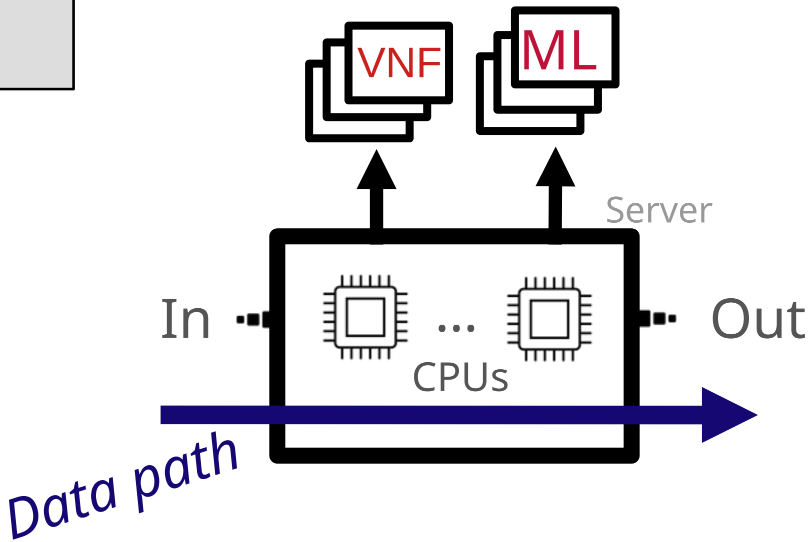
Complex analysis → more altering

Measurement operation will inevitably alter the state of the system.

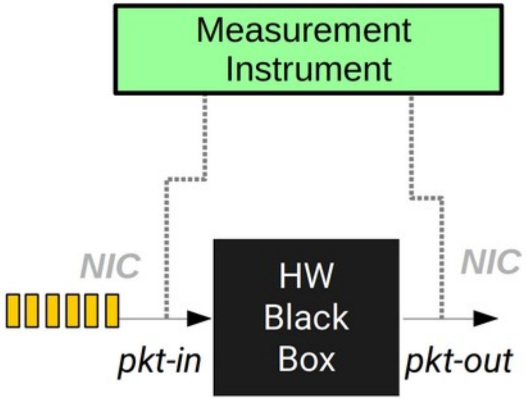
High-speed ML-enhanced functions on COTS servers

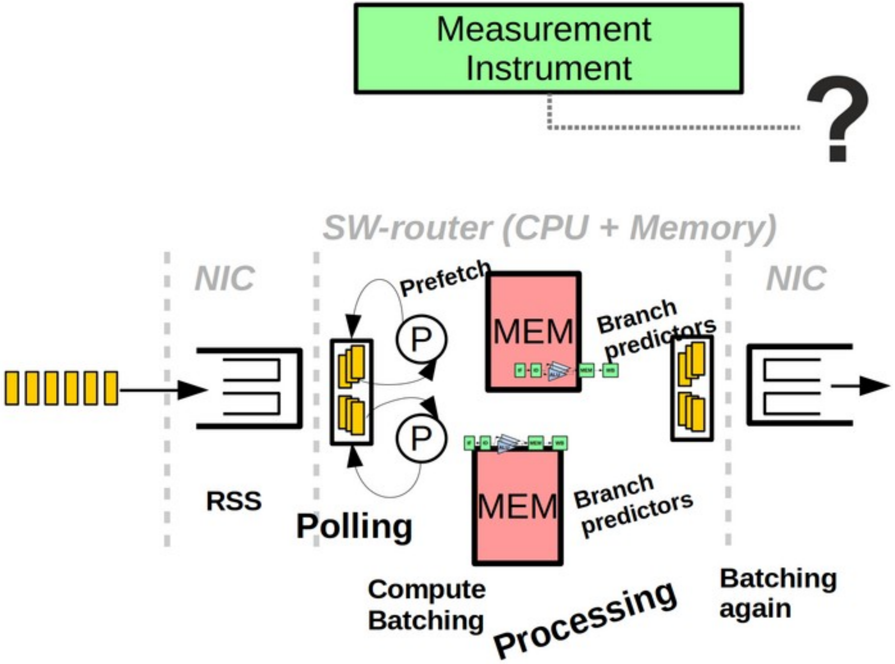


Note: accessing external devices for ML processing may not be possible



How to do measurements

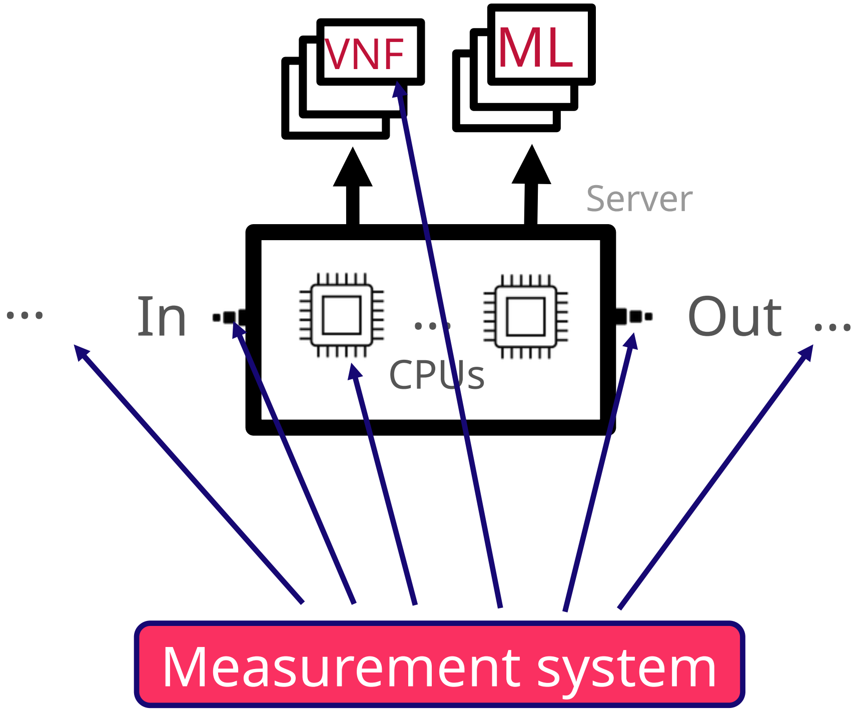




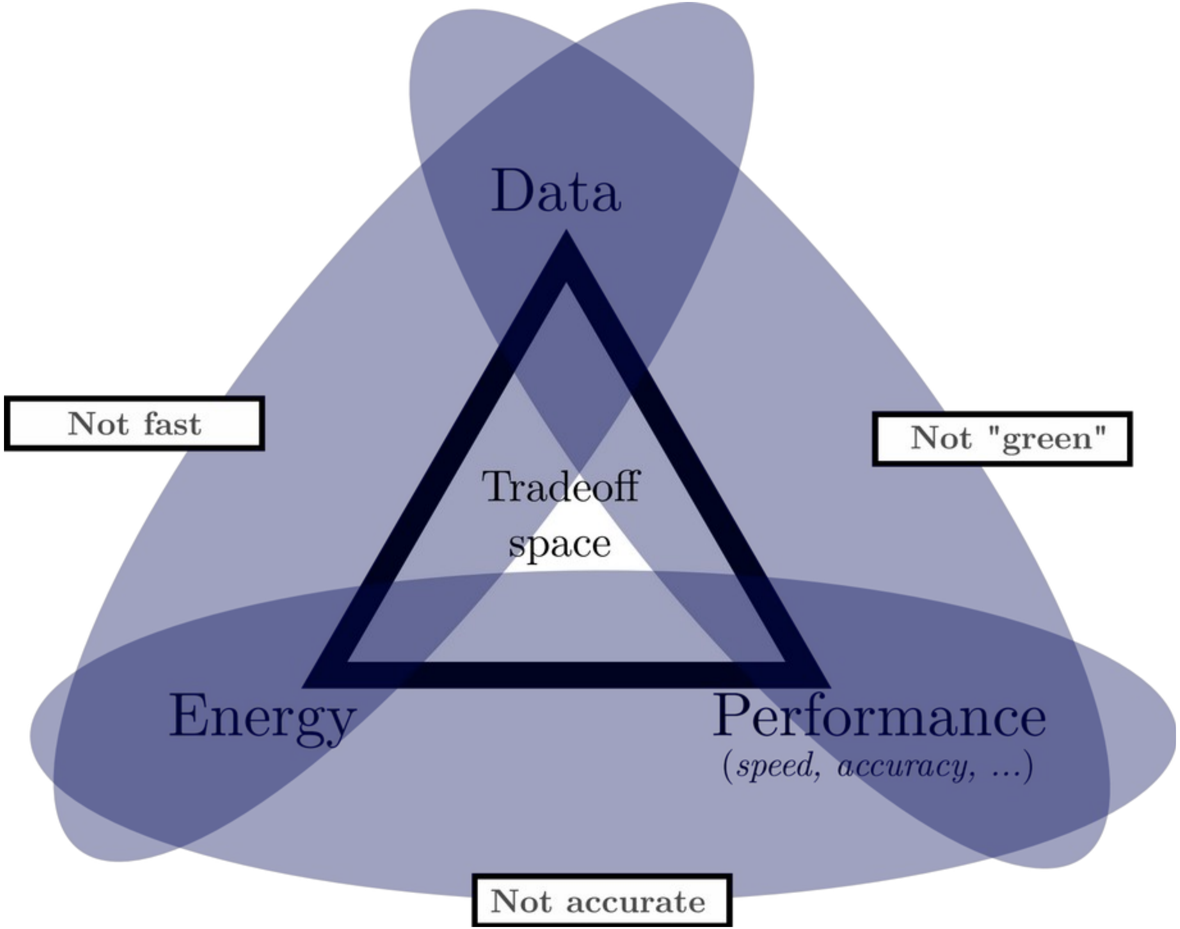
How to do measurements

- Pre-input (Data collector)
- Input NIC level (mirror, inspection)
- VNF level (pure software)
- CPU level (system level)
- Output NIC level (source demux)
- Post-output (Data collector)

In general: what should we do?



The main tradeoffs



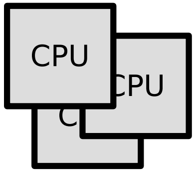
Part II

Methodology

Key idea : use **indirect** measurements...

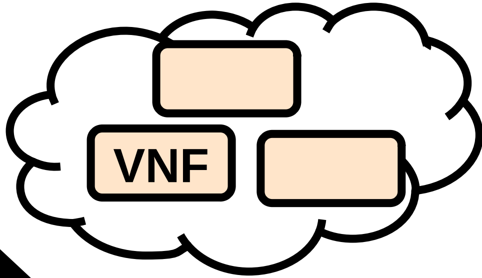
Low-level entities

e.g., `perf`, intel PCM



Get data from the bottom, analyze from the top

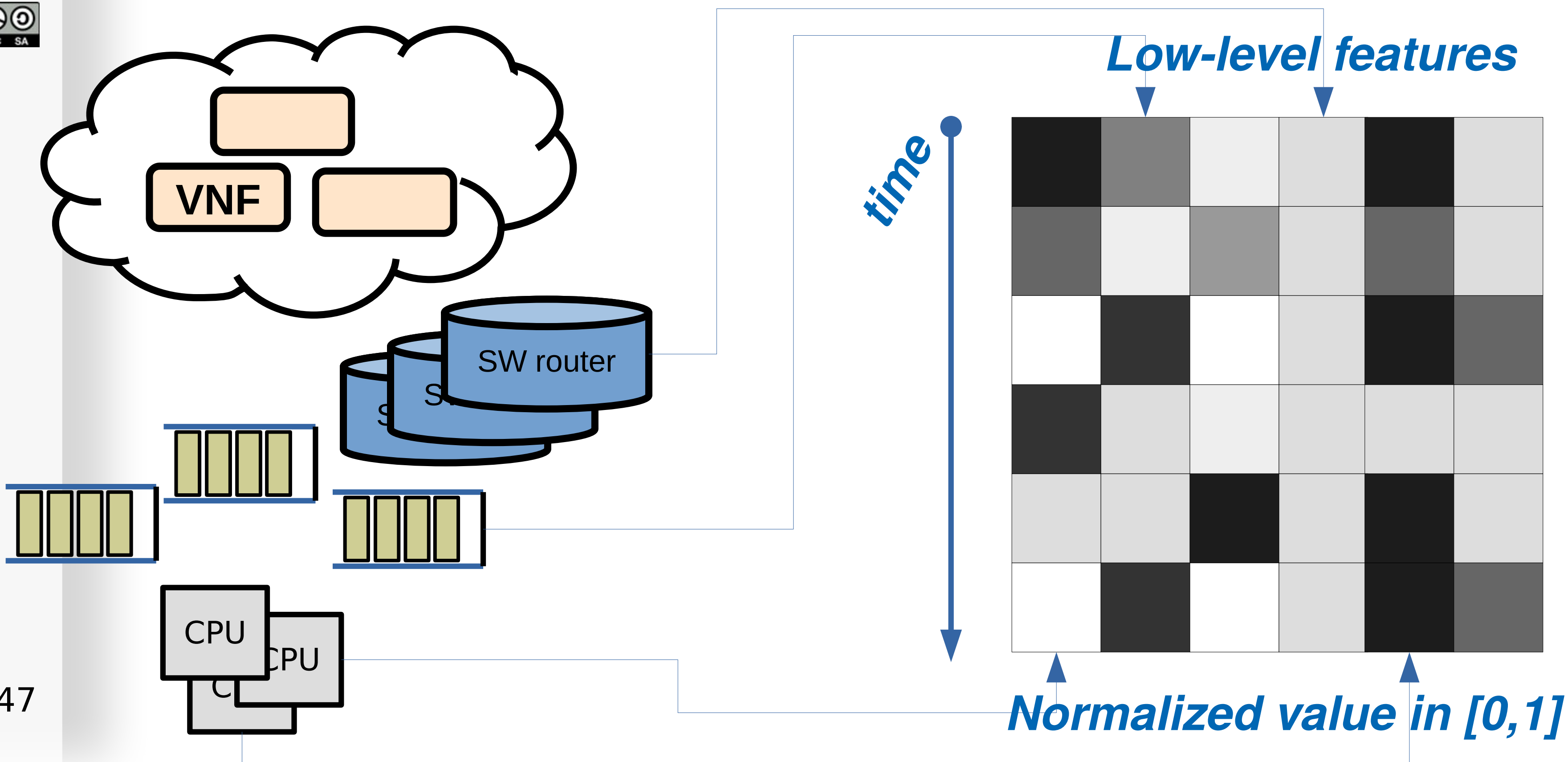
High-level entities



Easy to ***Monitor***
Hard to ***Interpret***
Data availability: ***Huge***

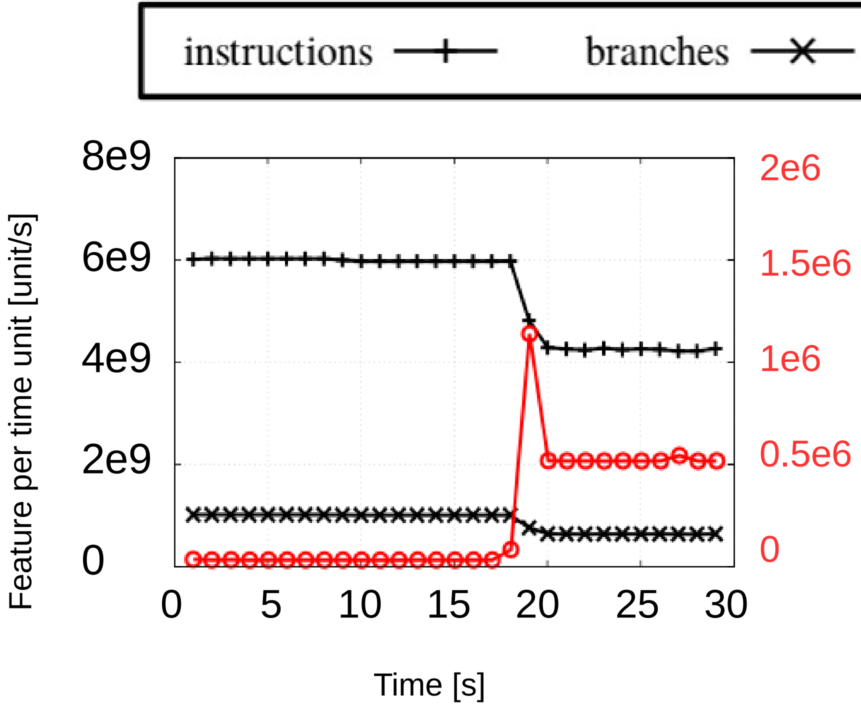
Hard to ***Monitor***
Easy to ***Interpret***
Data availability: ***Low-to-medium***

... to infer what is the high-level system state !



Are low-level features good predictors ?

(Yes, *[ConextStu2019]*)



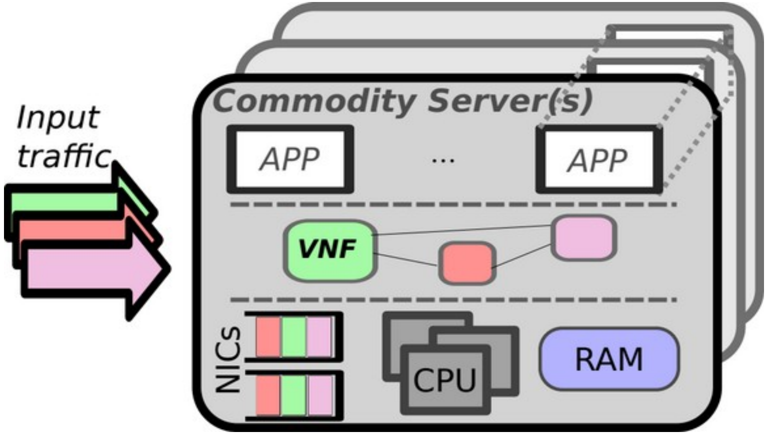
First half: low load

Second half: high load

Different features correlates to different behavior

Our methodology

- Sample application: detect VNF traffic/state anomalies with CPU measurements
- Precollect several CPU measurements and train very simple ML models
- Deploy the trained model in the data path (for instance, within the orchestrator)
- Access online CPU measurements to verify the current state



NOT everything

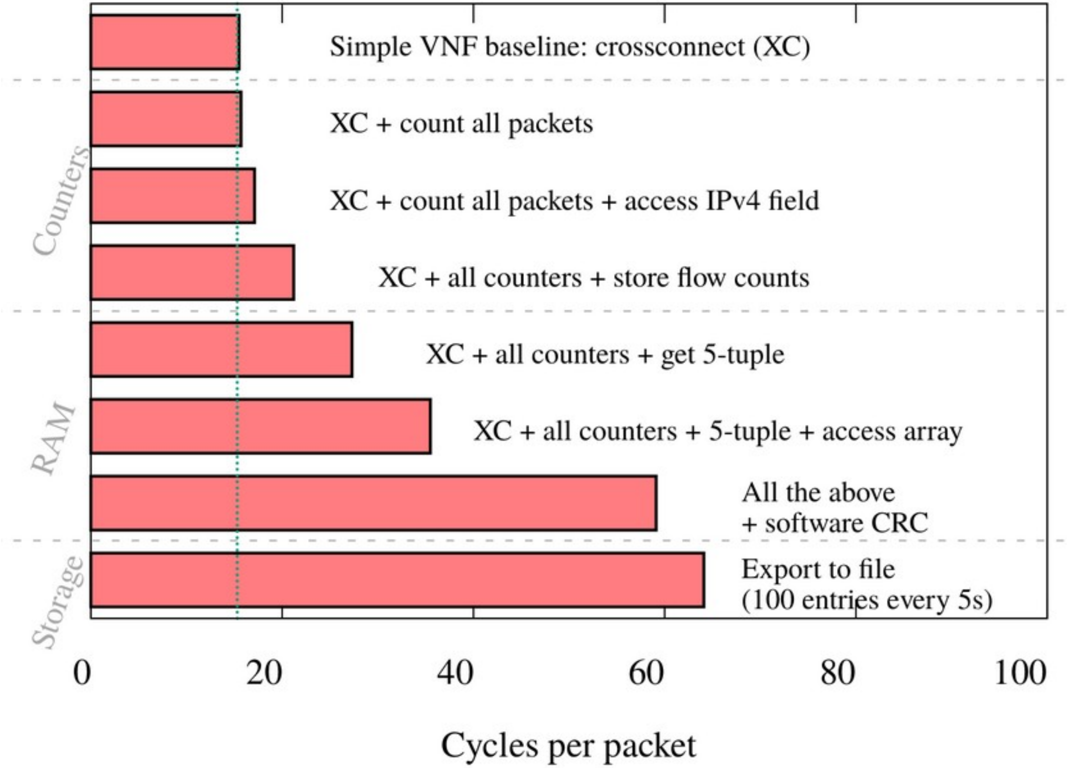
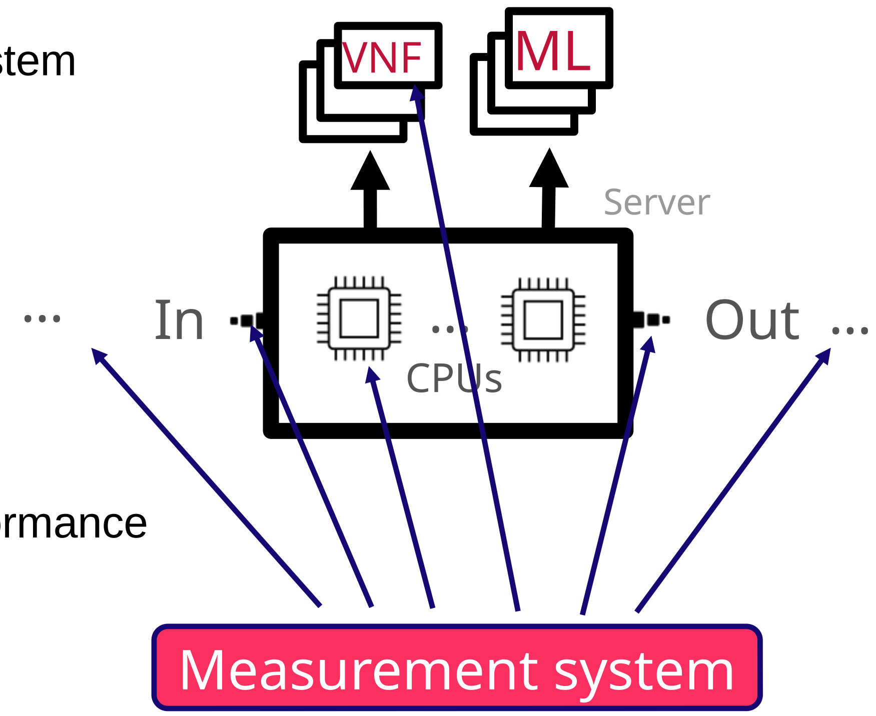


Figure from [TMA2019]

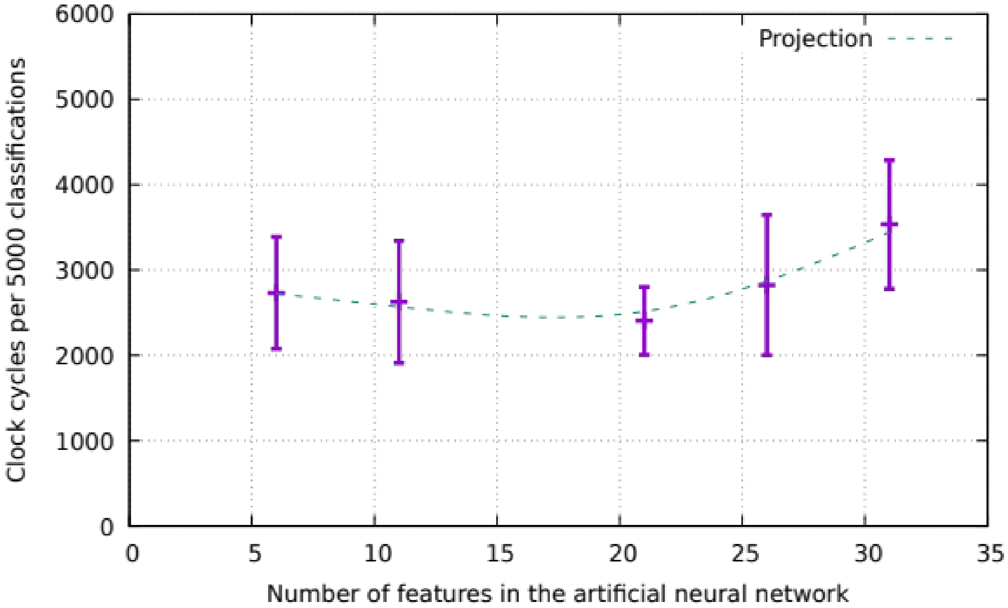
NOT everywhere

- Outside conditions affect the inner system
- Inner system affects the output
- CPUs already collect measurements
  - May be the best place
  - But cannot be done in real time
- Pure software collection has low performance
  - But external monitoring has costs



NOT all at once

- Pre-trained multilayer perceptron written in pure C and executed in pure SW
- 4 hidden layers, 10 neurons per layer
- The data structure used to store and run the model has an impact



*To be published.

What about prediction performance

- Expected traffic VS predicted traffic
- At high-rates the model struggles -> performance/accuracy tradeoff

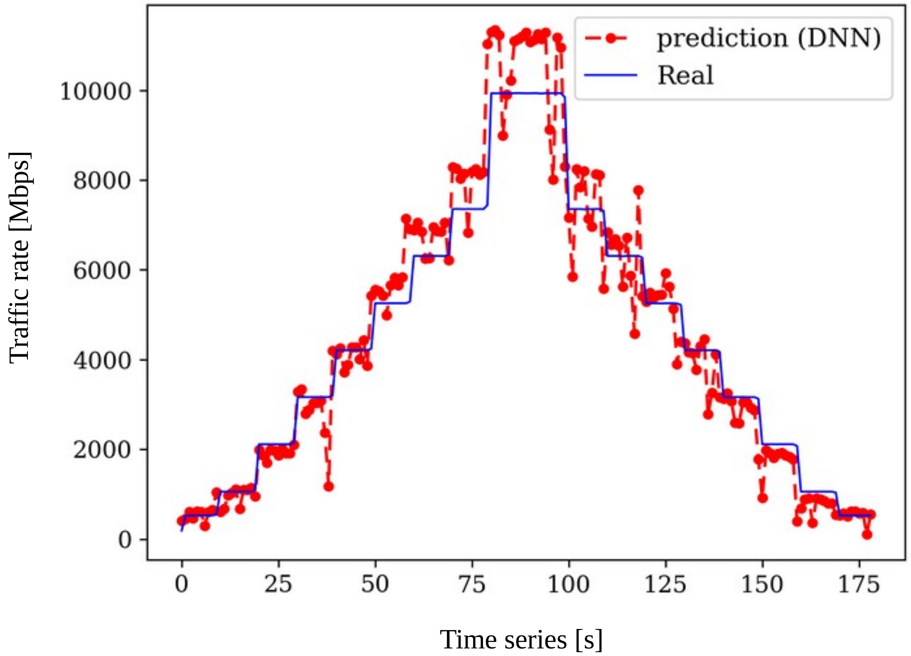


Figure from [INFOCOM2024]

In a nutshell

- Measurements in software components can alter the state of the system
  - Offline measurements : difficult to react to current conditions
  - Online measurements : difficult to collect data at runtime
- Analysis may require a large amount of data
  - Complex analysis → complex data → complex interactions
- The measurement problem must be mitigated
  - Less measurements → lose accuracy in predictions
  - More measurements → lose speed and alter the analysis
  - Pre-collected measurements → no reaction to real-time changes
  - **Indirect measurements :** correlate different types of measurements

Takeaways and future plan

- High-speed software networks come as a **high-cost / high-gain** problem :

  - lots of collectible data, but collection alters the system to be measured

- Study the **fundamental limits** of software measurements

- Propose **new methodologies** for network managers, operators and users

- **Key ideas:** (i) indirect measurements, (ii) simplify the input space, (iii) distribute the knowledge sources

- **ANR-funded for 4 years**

  - **1 Ph.D. student June 2023**

  - **3 Internship positions + 1 Post-doc TBA**

  - Collaborations with other institutions in both academia and industry

References

[IEEEProc2019] Linguaglossa L., Lange S., Pontarelli S., Retvari G., Rossi D., Zinner T., Bifulco R., Jarschel M., Bianchi G., "Survey of Performance Acceleration Techniques for Network Function Virtualization", **Proceedings of IEEE**, 2019

[ConextStu2019] Shelbourne C., Linguaglossa L., Lipani A., Zhang T., Geyer F., "On the Learnability of Software Router Performance via CPU Measurements", **ACM CoNEXT Student Workshop**, 2019

[ITC33] Shelbourne C., Linguaglossa L., Zhang T., Lipani A., "Inference of virtual network functions' state via analysis of the CPU behavior", **ITC 33**, 2021

[INFOCOM2024] Liu Q., Zhang T., Linguaglossa L., "Non-invasive performance prediction of high-speed softwarized network services with limited knowledge", **IEEE INFOCOM**, 2024

The NMS team: Network, mobility, services

- Team leader: Laurent Decreusefond
- **Projects and national/international grants**
  - PEPR « réseaux du futur » (5G+, 6G) (Daniel Kofman)
  - PIIEC (Philippe Martins)
  - Grants ANR (Leonardo Linguaglossa), ERC (François Baccelli)
  - Alexandre Pacaud, awarded the “Orange” thesis prix
- **Industrial chairs and common labs:**
  - Cisco (Jean-Louis Rougier)
  - SEIDO (EDF) (Jean Leneutre, Leonardo Linguaglossa)
  - LINCS (Nokia, SystemX, INRIA, Sorbonne, IMT) (Daniel Kofman)
- **Research/experimental platforms**
  - 5G platform (Core network, RAN) Philippe Martins
  - Low-tech compute (Minicluster Kubernetes) Leonardo Linguaglossa (E4C)

The PEPR project

- **Réseaux du futur, *“Networks of the future”***
  - Around 40 M€ for 10 target macroprojects
  - Architectures, building blocks, end-to-end systems, prototypes
- TP10 **SLICES-FR**
  - Initial deployment of national B5G/6G sites
  - ORAN, Core network
- TP2 **NAI**
  - Network architectures and infrastructures
  - Communications of next generation fixed/mobile networks
  - Network/cloud convergence
  - Self-deployable networks

Thank you for your attention!

Thanks a lot to all my collaborators!

Interested? `linguaglossa [at] telecom-paris [dot] fr`

Not interested? Feel free to tell me why ^

Part III

Use case

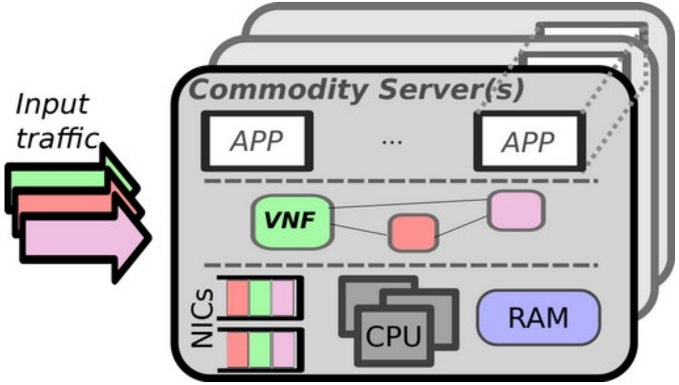
Inferring the system load via indirect measurements

A typical scenario

- Server owner
  - Rents her resources to Clients
  - Clients deploy their VNFs

- VNFs are linked to provide Services (APPs) using the low-level resources (NICs, CPU, RAM)

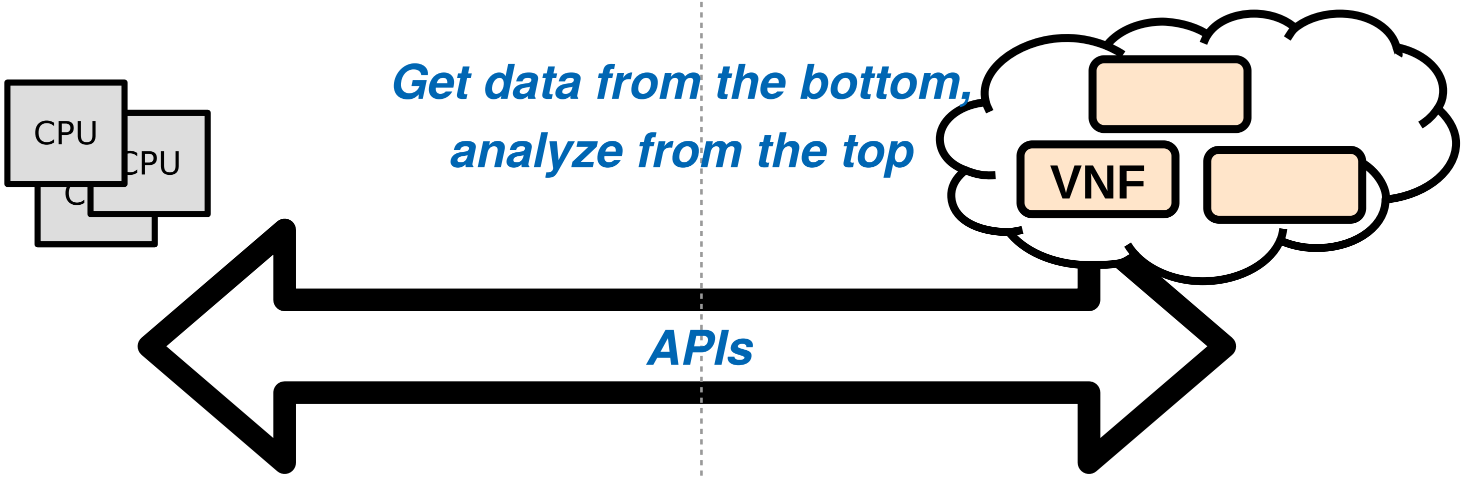
- Server cannot access the VNFs or the Services



Exploit the different expressiveness

Low-level entities

High-level entities

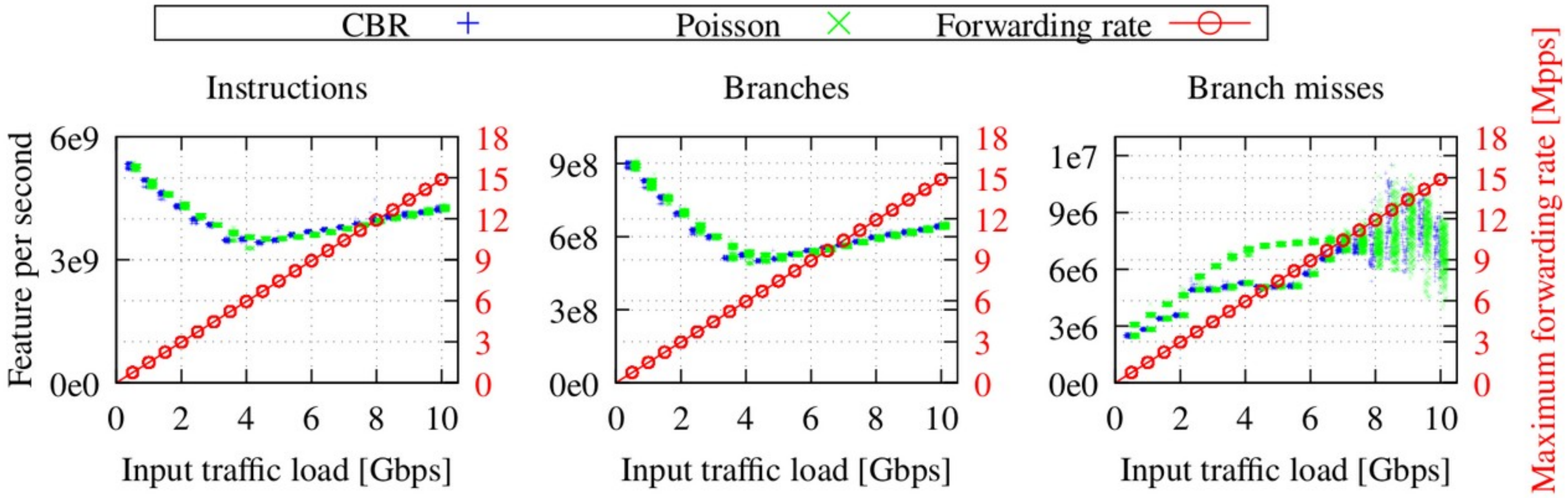


Easy to ***Monitor***
Hard to ***Interpret***
Data availability: ***Huge***

Hard to ***Monitor***
Easy to ***Interpret***
Data availability: ***Low-to-medium***

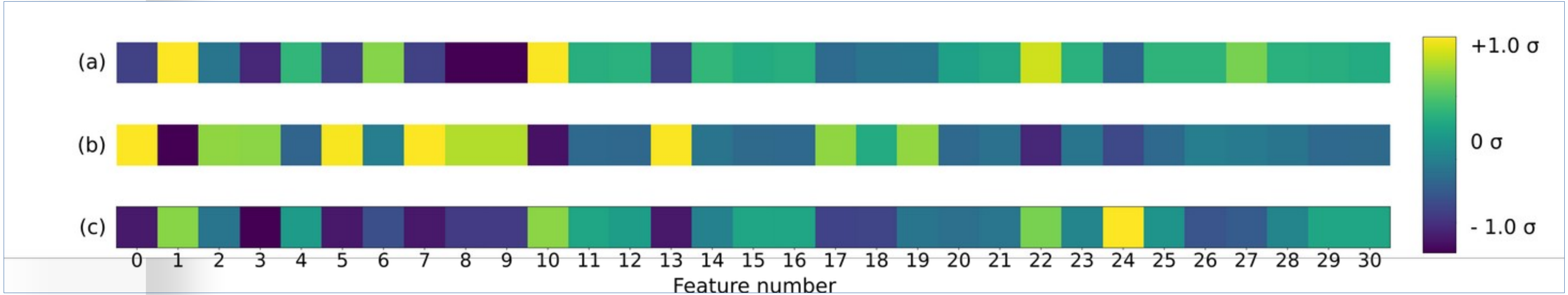
1) Adapt CPU utilization w.r.t. load

Analyze the CPU data (number of instructions, number of branches, …) to detect the Input traffic load and adapt the resource allocation



2) Detect different load/configuration conditions

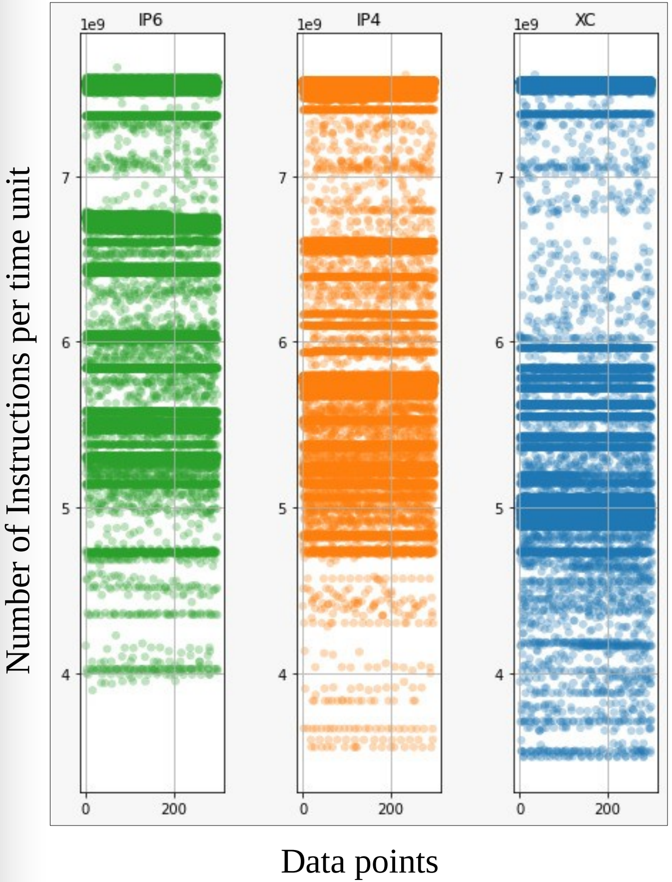
Analyze the CPU fingerprint and infer what is the possible underlying processing



Normalized value, measured by **perf** command, measured in units of standard deviations from the global average

(a) Poisson traffic at 3.5 Gbps with 256B packets; (b) Poisson traffic at 7.5 Gbps with 64B packets; (c) CBR trafficat 5.0 Gbps with IMIX packets.

3) Detect misbehavior/misconfiguration



Analyze the CPU data (**number of instructions,** in this figure) for a frequency analysis to understand what is the expected pattern of execution

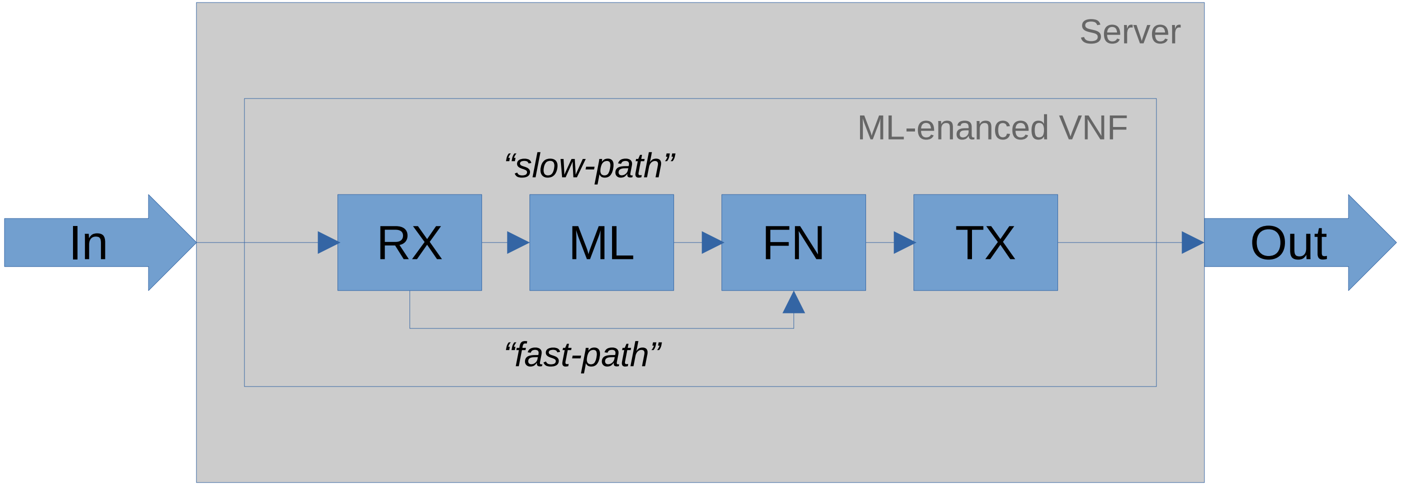
- Detect if the VNF is **optimally placed** (and adapt otherwise)

- Detect if a tenant is performing **unauthorized processing** with the allocated resources

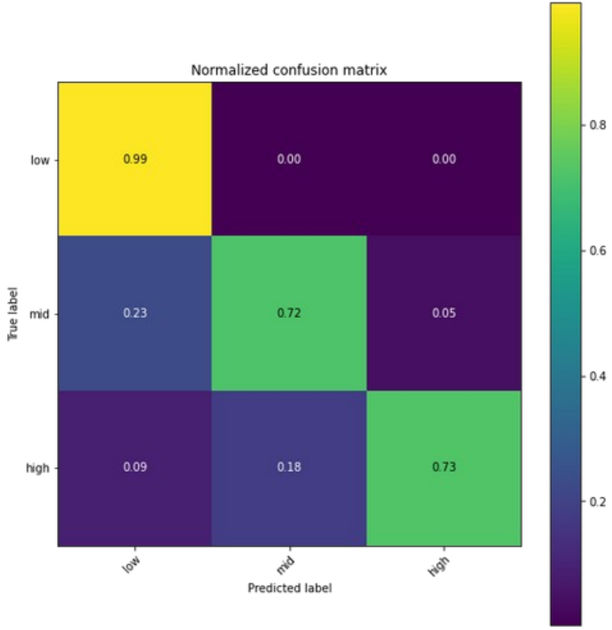
- Quickly react to potential attacks or threats

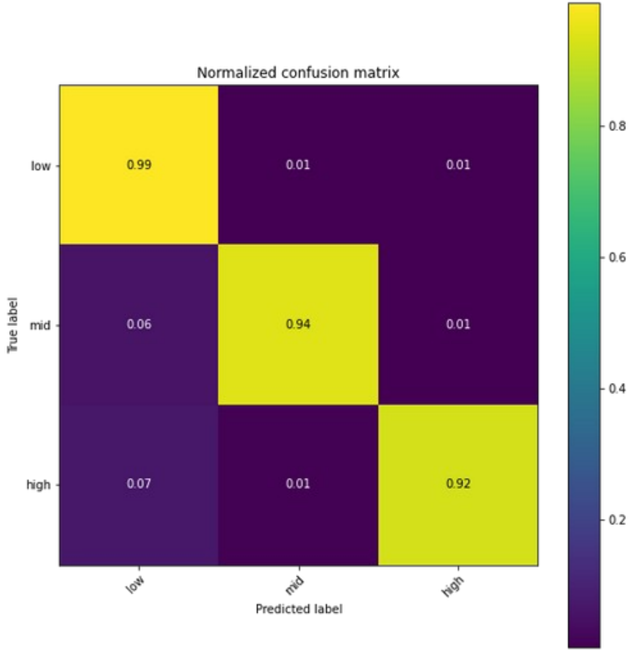
In-network ML and software packet processing

- Assume a ML-enhanced virtual network function (VNF)
- Model is pre-trained, deployment is in data-path



Model performance, AI context





- Objective: detect an anomaly in the network processing load
- Left: logistic regression
- Right: multilayer perceptron | 5 layers, 10 neurons per layer

VNF performance, network context

- We run a 20s emulation in python
- The model that performed the worst, has 0% packet loss
  - In the data path, there is not a **visible alteration**
  - It depends on the initial load

- The model that performed the best, has ~45% packet loss
  - For every packet received, a packet is not transmitted
  - The model is altering the state of the system
  - The “anomalies” cannot be detected
  - The system is not the same system to be monitored

- A real-life scenario will have unknown/unquantifiable interactions