Integration of Machine Learning and Networking Intents for the Orchestration of Distributed IIoT Devices

Speaker: Antonino Angi

Politecnico di Torino

Research challenges

Nowadays, the level of complexity of the network is growing

As we experience always more connected devices and a more need of data-intensive applications

Exploit advances in Machine Learning (ML)

to study the condition of the traffic

SDN and NFV approaches with data-plane programming languages have been adopted

to manipulate the traffic according to the network status

Software-Defined Networking

Software-Defined Networking (SDN) is a new approach to networking

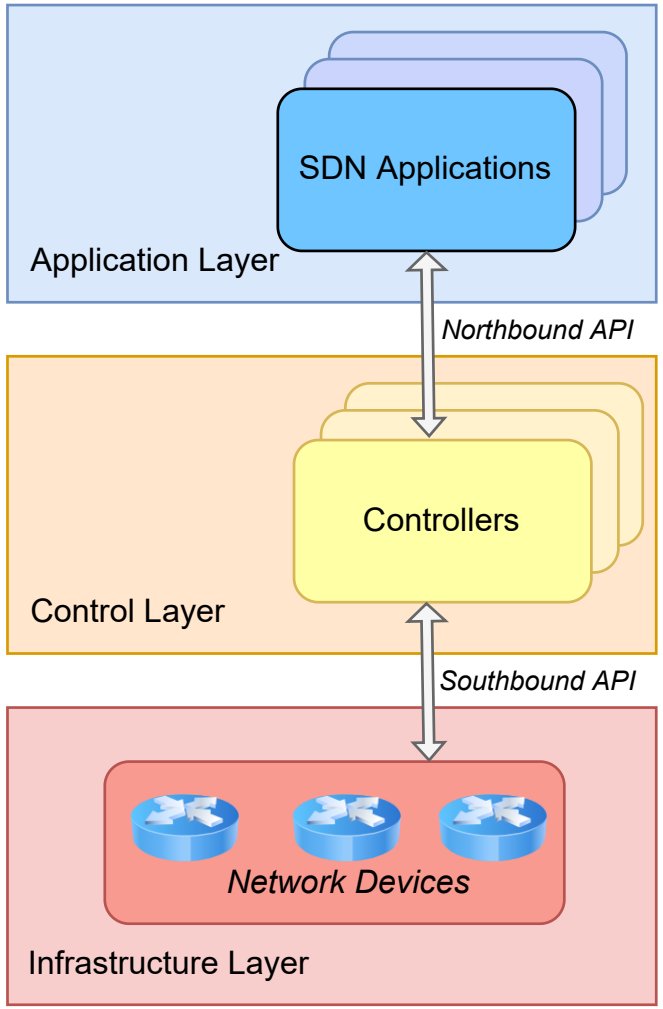
based on the design of the network control

Network Function Virtualization (NFV) complements SDN

Firewall, Load balancers: from hardware to software

+ Innovation feasibility
+ Simpler, cheaper equipment
+ Programmatically configured

- Virtualizing resources could increase latency
- Configuration is not easy

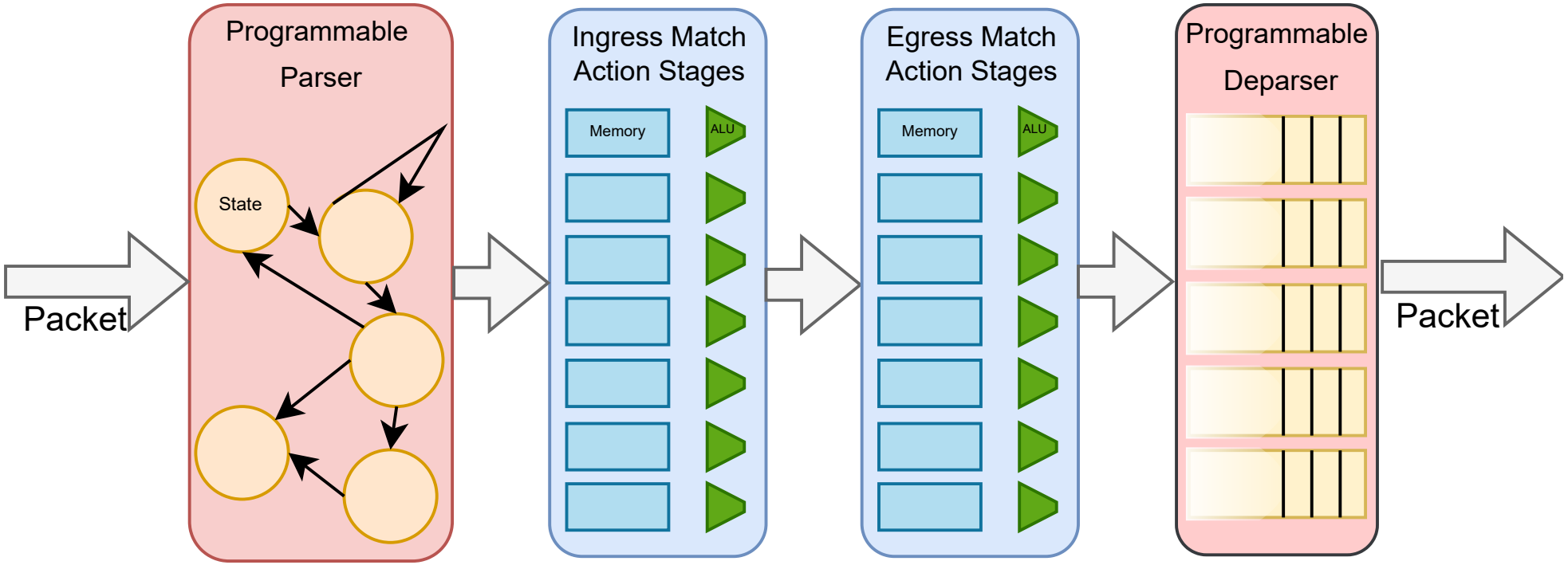


SDN architecture

Programming Protocol-independent Packet Processors (P4)

Programming Language to customize data-plane

based on the PISA architecture



PISA architecture in P4

Research challenges

In data-centers, network resources need load-balancing strategies

they require data at a faster speed but lower latency

Several strategies were proposed to handle the traffic load

but they

- Consider small flows (PRESTO);
- Only have local information (DRILL);
- Rely on hardware implementation (CONGA);
- Need specific overhead packets to get information on the network status (HULA + CONTRA)

Research question

How can we **balance network traffic** over paths **without elaborating protocols** for the exchange of information **between switches** in IIoT networks?

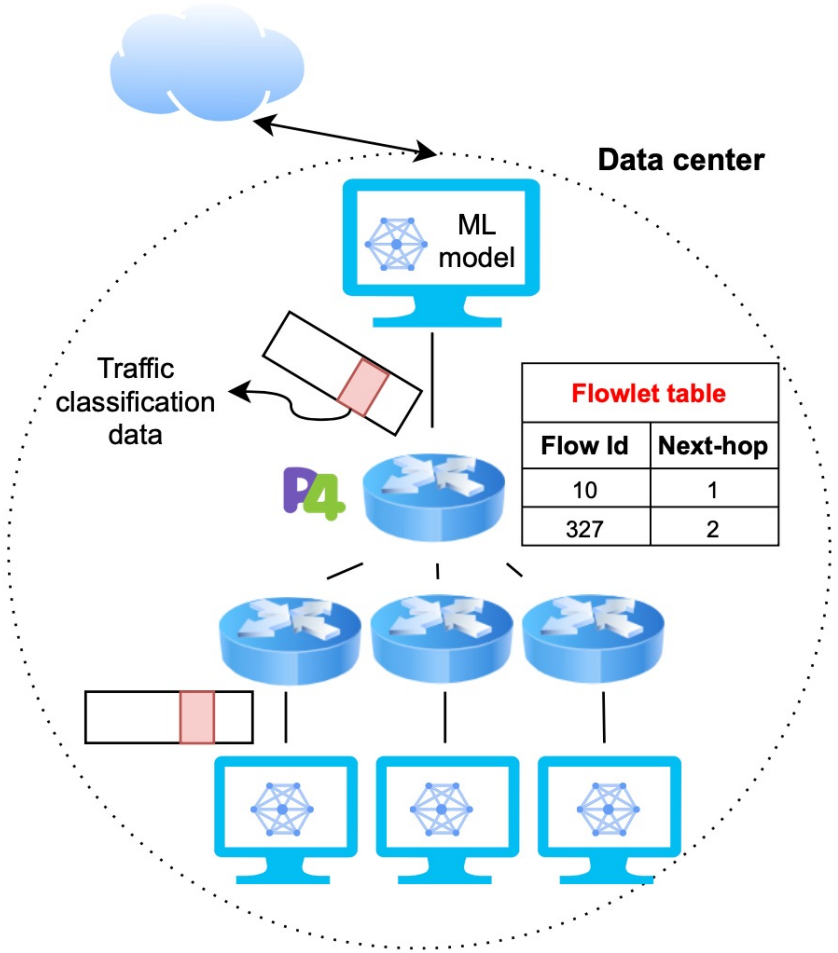
Research question

Howdah: Load Profiling via In-Band Flow Classification and P4

Howdah: Load Profiling via In-Band Flow Classification and P4

Host – switch cooperation based on the flowlet strategy

- ML on the host for traffic classification;
- P4-based switches for adopting forwarding decision according to the network load status.



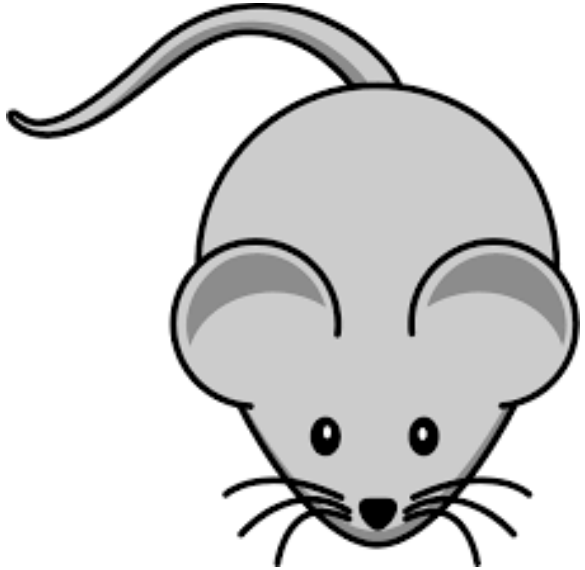
Howdah overview

Traffic classification in Howdah

Type of traffic → different actions

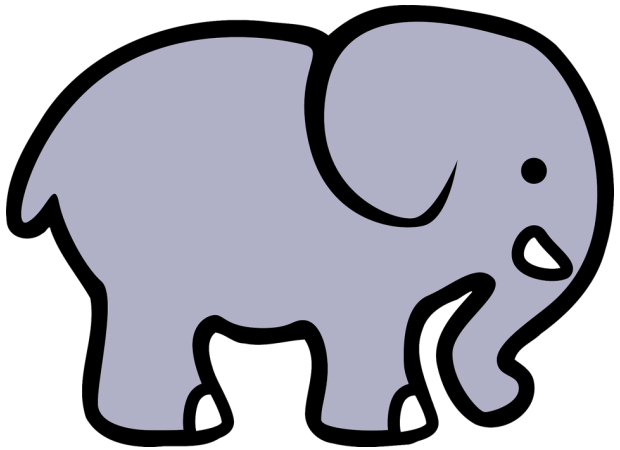
- If bytes < D
- Connection in seconds < L

→



- If bytes >= D
- Connection in seconds >= L

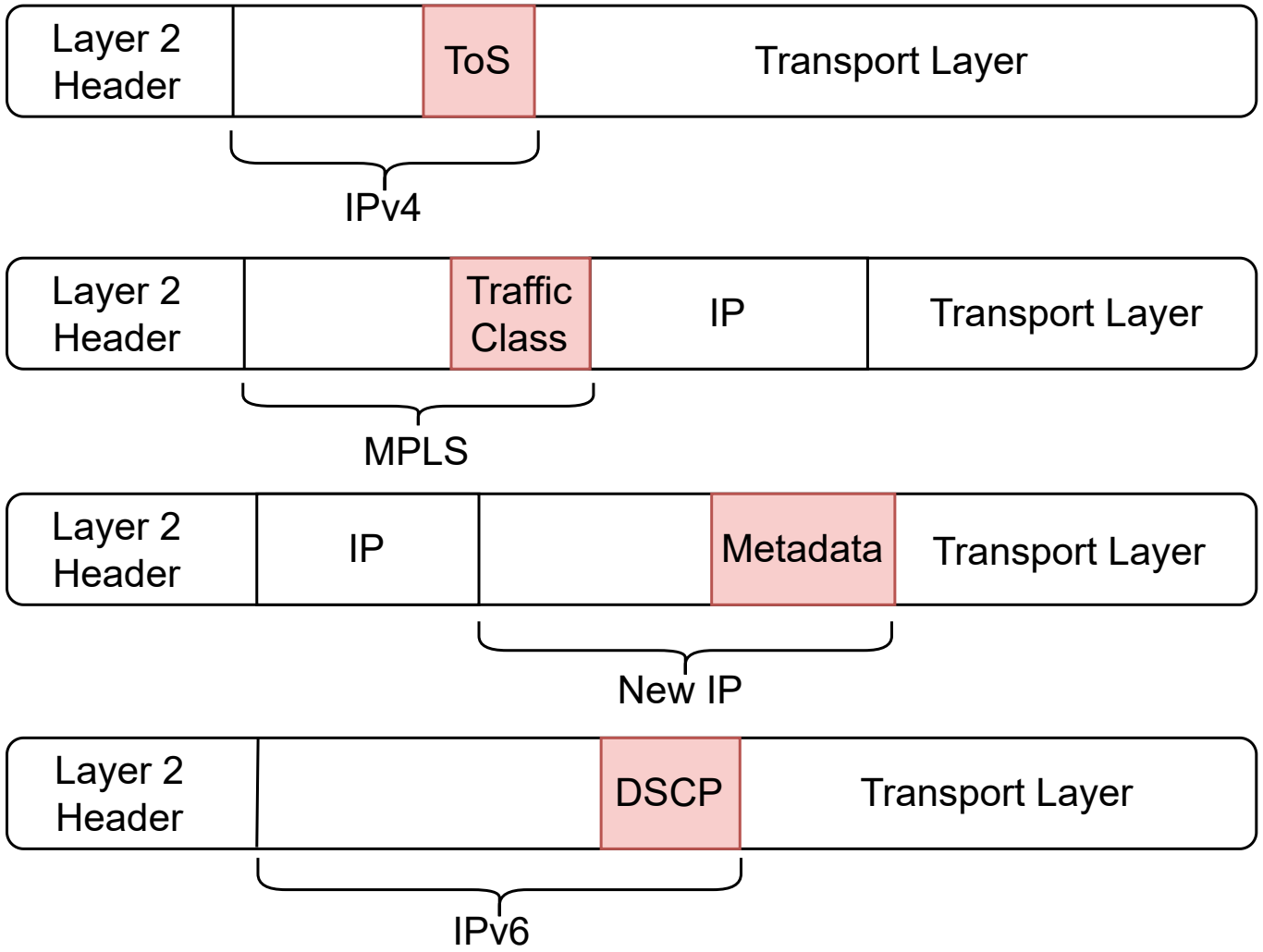
→



The host in Howdah

1. Classification with a Decision Tree model

2. Insertion of the label into the packet

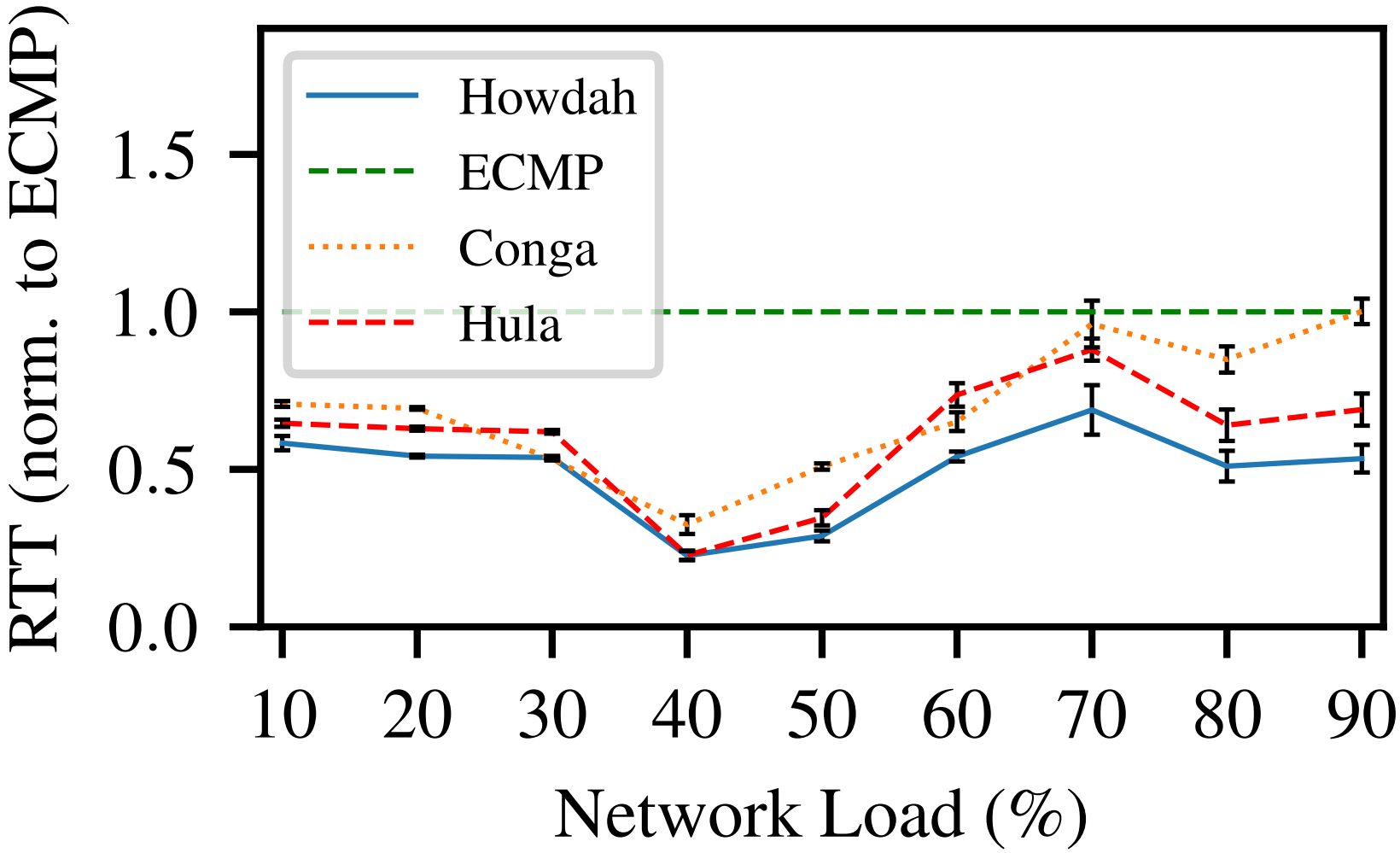


How it works: the P4 switch

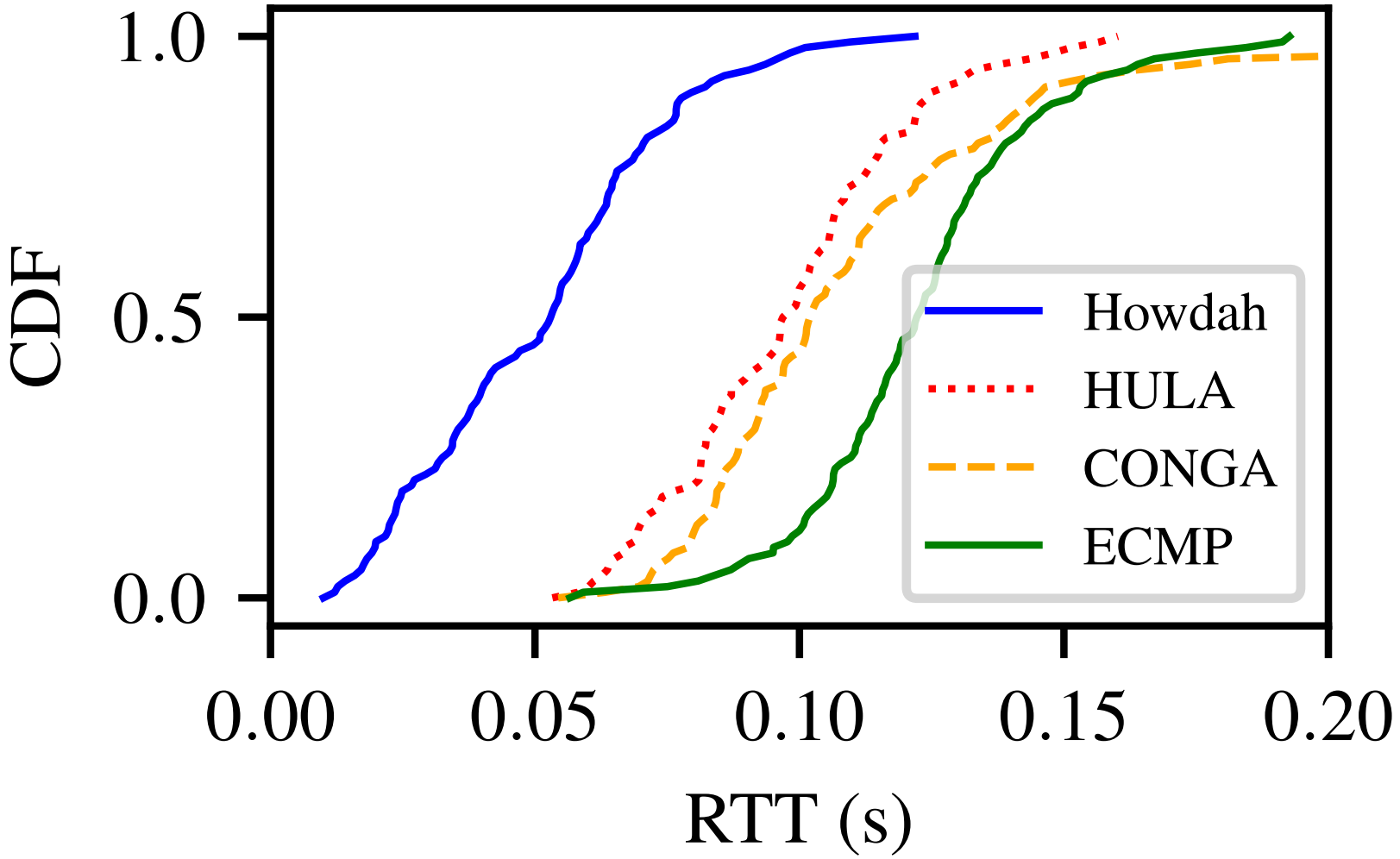
- If mice → next-hop in a flowlet-based ECMP
- If elephants
  - If the flowlet is known → $T = t_{arrived} - t_{stored}$
    - If <= *T* → forwards the flowlet using the stored next-hop value in the flowlet table;
  - If > T or flowlet unkown → hash(protocol, IP source & destination address, transport protocol source & destination port) and stores the next-hop with the LRU.

Flowlet table	
Flow Id	**Next-hop**
10	1
327	2

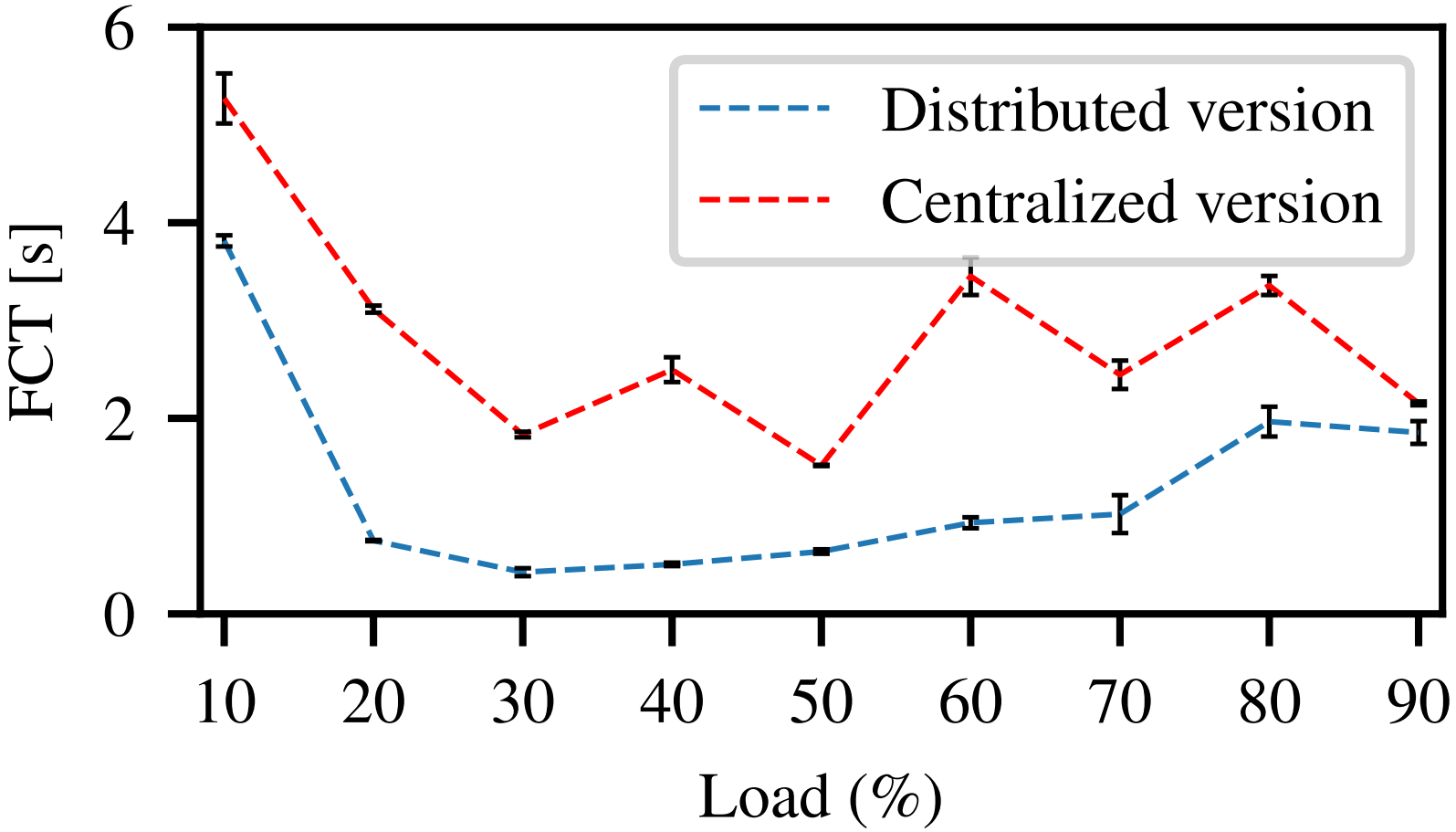
Howdah allows to achieve lower RTTs



All RTT responses are received in 0.12 *s*



Distributing the logic allows to achieve lower FCTs



Discussion on Howdah

Howdah shows that injecting information reacts well even in cases of a congested network

performing better than state-of-art solutions

Each switch locally decides on the best next-hop

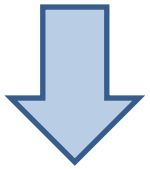
demonstrating even a link failure resistance and adaptability to topology changes

Howdah can be applied to different protocols by including in-band information about on-going traffic type

according to the different needs

P4 programming is problematic even for experts

- No loop cycles
- If-else conditions only in specific areas
- ‘’Trial-and-error’’ process
- Hundreds LoCs for simple forwarding



Intent-based Networking

Intent: High-level targets, written by humans, that a system should meet with no specification on HOW to reach them

Examples

- *Enable firewall in all switches*
- *Block all traffic from GroupA to GroupB*

NLP4: An Architecture for Intent-Driven Data Plane Programmability

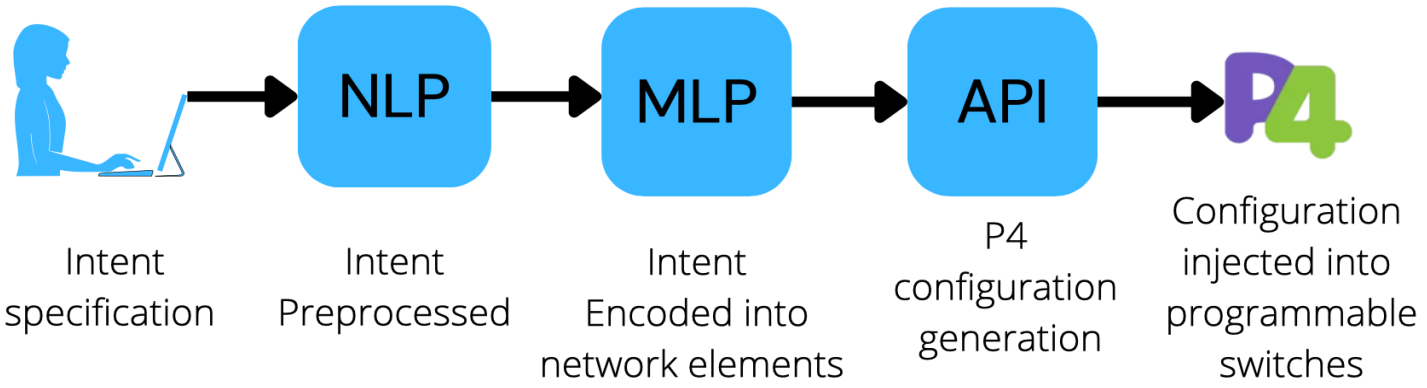
Translate intents, human language, into data-plane programs, P4 rules

using NLP for the interpretation

Designing an API

that can be generalized over different languages

Final goal: let users customize the network as they want



Natural Language Processing (NLP)

1. Text cleaning

   text -> tokens

2. Normalization with stemming technique

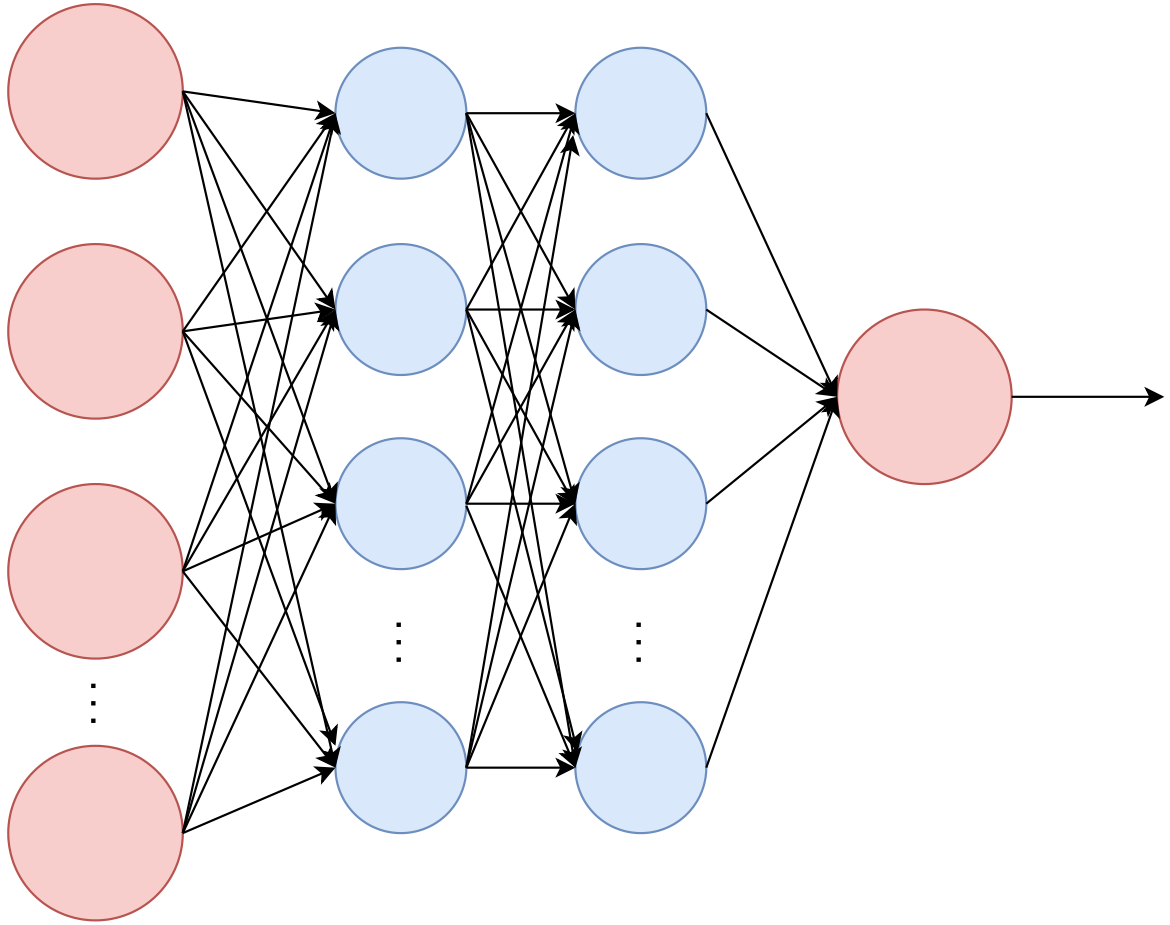
   tokens -> root form

3. Encoding

   root form -> vector

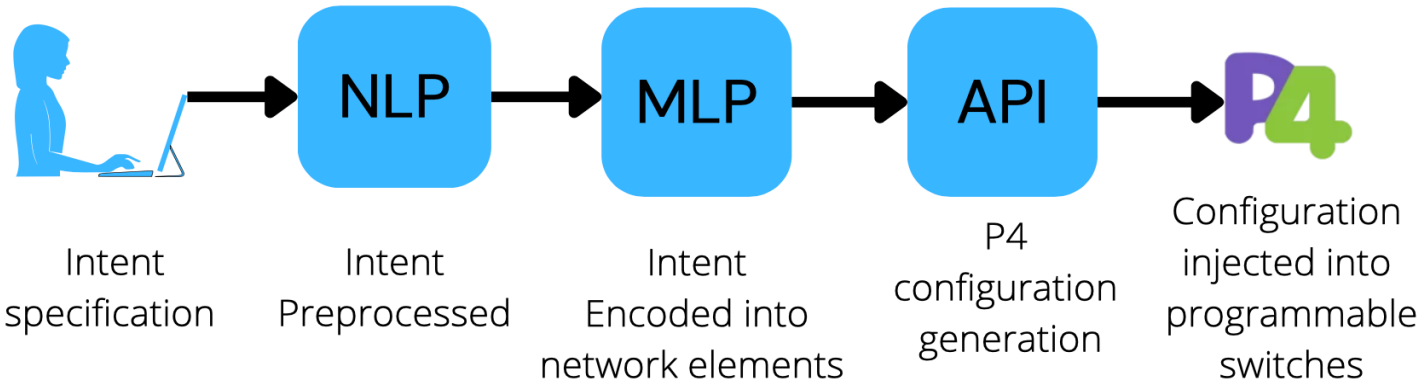
MultiLayer Perceptron (MLP)

- Flexibility when applied to different contexts and types of data
- Creates a mid-level representation that our API can use
- Maps preprocessed intents to the network elements

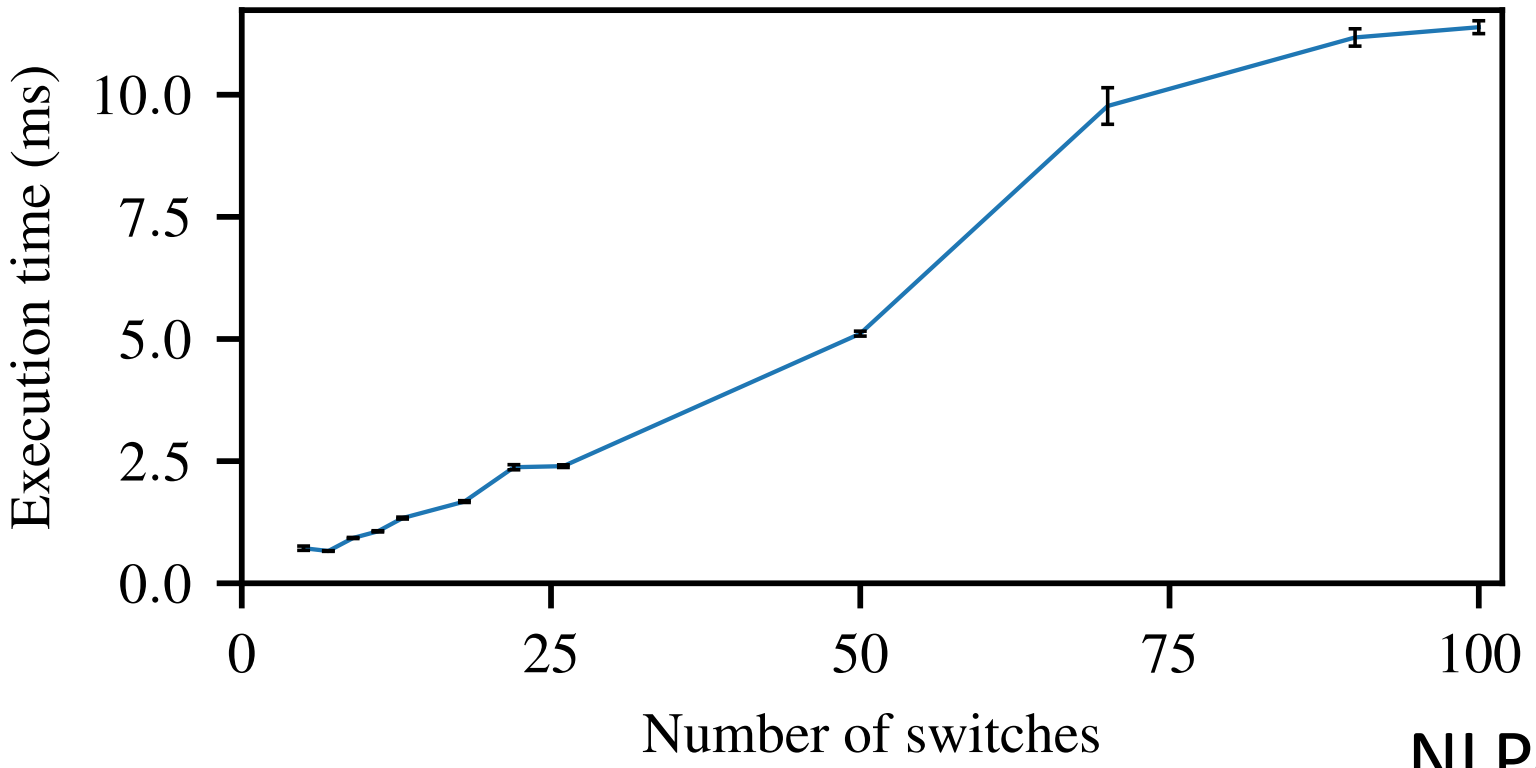


Intent-Driven API

- To convert the encoded array into P4-enabled switches' configuration files
- Following the idea of *behave*, a Behaviour Driven Development (BDD) framework
- *Scenario-Given-When-Then* formula

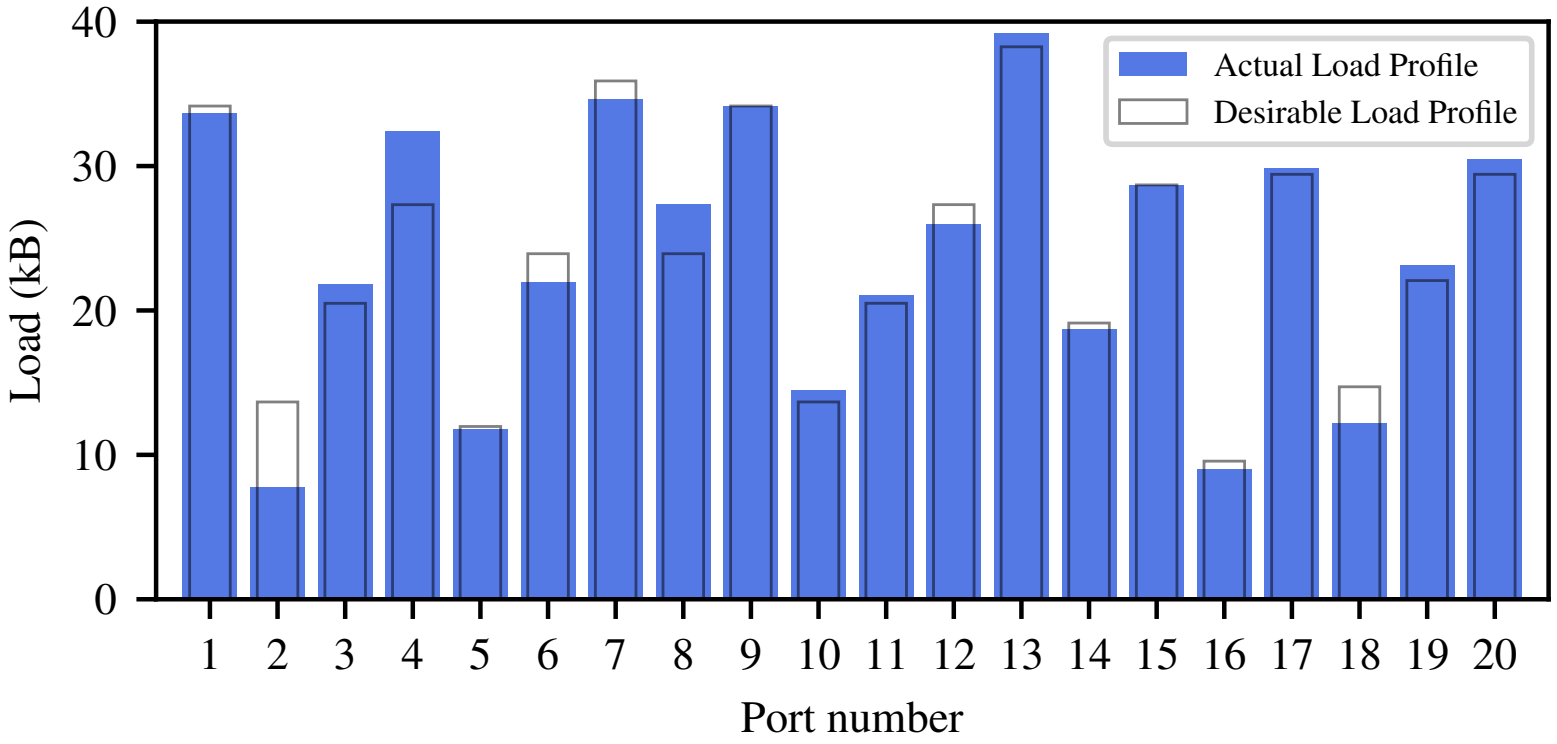


Evaluation Results: Running NLP4



NLP4 maintains a limited execution time as the network grows

Evaluation Results: Efficacy in Profiling the load



NLP4 allows to obtain the desirable load profile for each port

Discussion on NLP4

IBN facilitates the spread out of SDN

Policies and intents open new research opportunities

NLP4 is our preliminary attempt to create an easy-to-use IBN interface

We are currently working on expanding this work using a more interactive API and facilitating the network administrator experience and usability

THANK YOU

Antonino Angi

Politecnico
di Torino

Backup

Example of input

Input: ”Block traffic from GroupA to GroupB”:

1. Tokenization: the intent is converted into tokens resulting in “['block', 'traffic', 'from', 'groupa', 'to', 'groupb’]”

2. Stopword removal: the deleted tokens are those belonging to the English stopwords dictionary. In this case, the deleted tokens are “from” and “to” resulting in “['block', 'traffic', 'groupa', 'groupb’]”

3. Stemming technique: we use the Porter stemmer, one of the most accurate and used stemming techniques. In this case, our tokens are not modified as they already are in their canonical form

Example of input

According to the content of the tokens, our MLP matches with the word “block” and understands that the wanted action is to **apply a firewall from the first group**, i.e., “**groupa**” to the second one, i.e., “**groupb**”.

Finally, using the API, our solution takes a source template P4 code, with minimal functionalities (i.e., IPv4 forwarding) and for each involved switch, writes its configuration file.

Why was a DT chosen?

	US-UNV-1				US-UNV-2				US-UNV-3			
	Accuracy	*Precision*	*Recall*	*F1-score*	*Accuracy*	*Precision*	*Recall*	*F1-score*	*Accuracy*	*Precision*	*Recall*	*F1-score*
SVM	0.94	0.96	0.94	0.97	0.94	0.96	0.94	0.96	0.99	0.99	0.99	0.99
NN	0.93	0.92	0.99	0.96	0.93	0.93	0.99	0.95	0.99	0.99	0.99	0.99
k-means	0.83	0.99	0.83	0.91	0.84	0.99	0.84	0.91	0.97	0.99	0.98	0.99
RF	0.99	0.99	0.99	0.99	0.93	0.93	0.93	0.93	0.97	0.97	0.97	0.97
D-Tree	0.99	0.98	0.99	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.97	0.98

	Tr. time [s]	**Class. time [μs]**	**CO_2 (T) [mg]**	**CO_2 (C) [mg]**	**RAM (T) [%]**	**RAM (C) [%]**	**CPU (T) [%]**	**CPU (C) [%]**
SVM	0.927	0.0264	1.933	0.0417	2.013	2.022	15.554	14.523
NN	170.869	4.081	416.6	1.8391	2.766	2.758	18.328	18.207
k-means	0.647	0.07974	2.013	0.0728	1.959	1.954	16.621	15.583
RF	83.439	10.562	192.4	3.1312	1.872	1.851	16.58	16.983
D-Tree	1.0202	0.107	1.733	0.0481	2.969	2.93	16.619	16.506

Why was a DT chosen?