Automating Network Configuration With Natural Language Intents

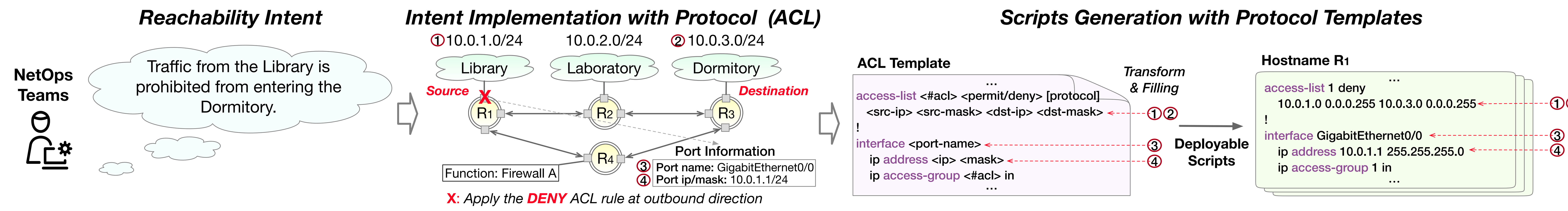
Wenlong Ding[1], Jianqiang Li[1], Zhixiong Niu[2], Huangxun Chen[3], Hong Xu[1]
[1]CUHK, [2]Microsoft Research, [3]HKUST (Guangzhou)

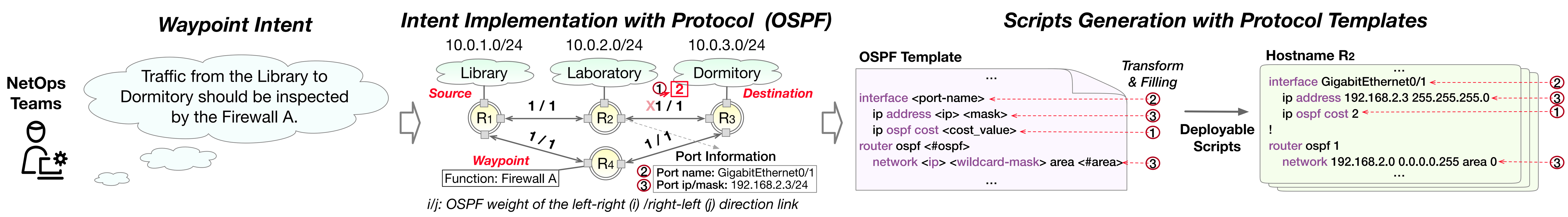
The Chinese University of Hong Kong

SIGCOMM 2024 — SYDNEY —

Introduction

[Our Goal] Automate three tasks to generate deployable configurations directly from natural language (NL) intents.





1. *Intent Understanding*: Interpret NL inputs to identify necessary network specifics and policies for configurations (e.g. source prefix or ACL action).

2. *Intent Implementation*: Solve parameter settings in network topology with proper protocols (e.g. OSPF link weights or ACL configuration ports).

3. *Scripts Generation*: Generate deployable configuration scripts by filling identified network specifics and protocol settings in templates.

Despite the widespread exploration of template-filling-based methods for Script Generation, unresolved challenges remain in the other two tasks.

Intent Understanding

[Overview] Use *LLMs* with *Prompt Engineering* to identify all necessary configuration specifics.

Why LLM

Related works

Named-Entity Recognition (*NER*) with traditional language models (e.g. BERT), such as identifying “Library” as source endpoint.

Limitations: Low generality

Related works requires training different models for different network corpora and specific retraining for entity synonyms (e.g. “Lib”, “Library” and “10.0.1.0/24”).

LLM: Suitable to solve expression variations

LLMs are pre-trained models with strong text processing abilities for various corpora, capable of solving numerous text-based tasks via prompting instead of repeated training.

Challenges with LLM

Handle implicit network information

Certain network-specific information needed for configuration (e.g. prefixes) may not be explicitly stated in NL and varies between networks.

Specify key identification elements

LLMs should understand which elements are necessary to identify for the current configuration, such as "source prefixes" and “policy” to allow or deny flows for ACL intents configuration.

Design: Prompt Engineering

Implicit information database

We map NL names to corresponding network-specific information and supply it to LLMs, using prompts like “*Library” -> “10.0.1.0/24*”.

Key identification elements description

We provide key identification elements with explanations to LLMs. Our prompts for reachability intent can be: "*Policy: <permit/deny intent flow>*".

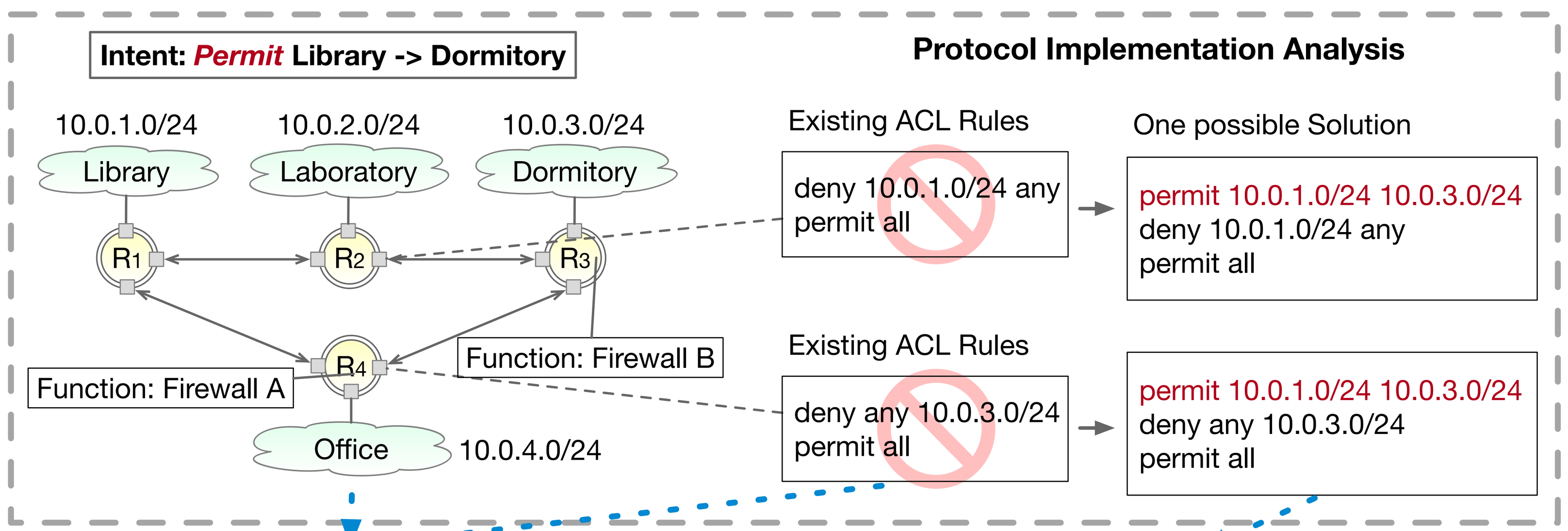
Understanding process guidance

We describe the details about how to extract key identification elements using NL and implicit information databases.

Intent Implementation

[Overview] Use *priority-based framework* to reconcile *configuration conflicts* in *Intent Implementation*.

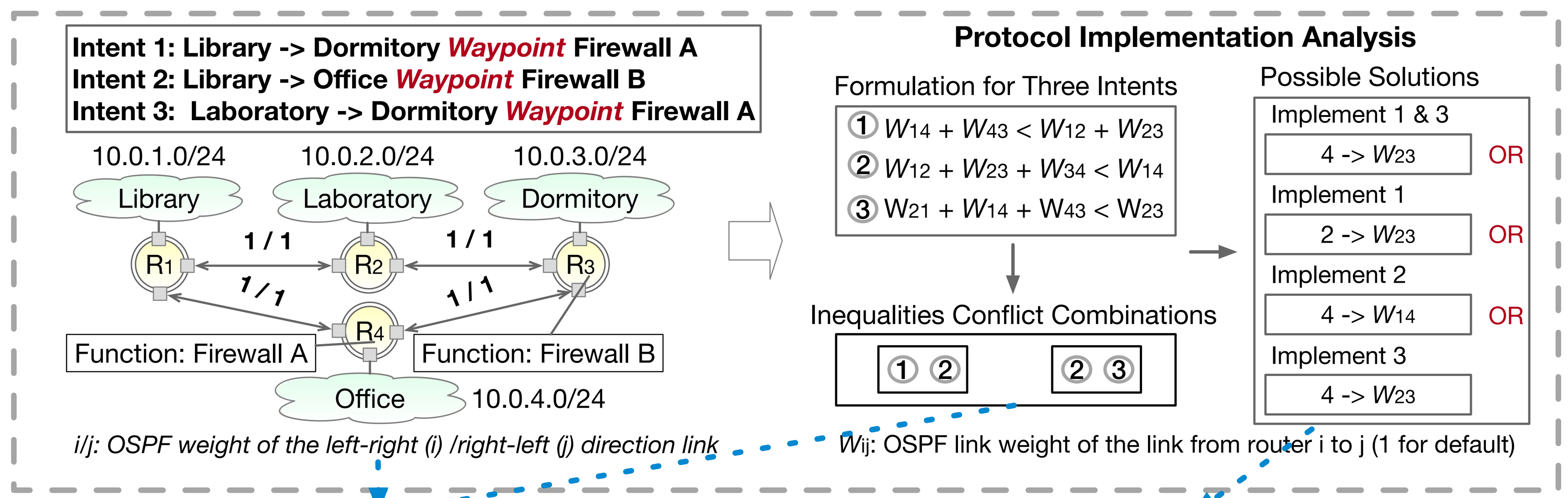
Conflict (a): New Intent and Existing Configuration



Intent and existing ACL rules manage overlapping flows with opposite action.

If achieving new intent, partial existing old settings will be overwritten.

Conflicts (b): Between New Intents



Three new intents cannot be achieved simultaneously with formulation.

Only partial new intents can be achieved and there are many partial solutions.

Since some configurations cannot be achieved simultaneously, NetOps teams should reconcile these conflicts.

Design: Priority-Based Framework

NetOps teams set a *priority* to *each* new intent and existing setting.

Satisfy intents with higher priorities when conflicts happen by *maximizing priority sum*.

Two Examples (In Conflict (b) Situation)

Priority of three intents: 1, 4, and 1. We satisfy intent 2 with priority sum equal to 4.

Priority of three intents: 1, 1, and 1. We satisfy intent 1 and 3 with priority sum equal to 2.

Preliminary Results

[Overview] Our *Intent Understanding* achieves high accuracy with fast inference time in mere seconds.

Results

Dataset	Dataset Information		Evaluation Results				
	# Intents	# Chars Per Intent	Metric	Llama 2	Mistral	GPT-3.5	GPT-4
Hand-crafted	150	81.3	Accuracy	91.3%	86.7%	98.7%	100%
			Time (s)	2.25	2.59	1.38	3.98
AI-generated	300	81.9	Accuracy	88.0%	88.7%	98.3%	100%
			Time (s)	2.89	2.33	1.34	4.13

Conclusion

1. Advanced models (GPTs) achieve over 95% accuracy, with GPT-4 even reaching 100%.
2. Inference time range from a few seconds for all models.
3. Two datasets yield similar accuracy and inference time results.