Deploying Stateful Network Functions Efficiently using Large Language Models

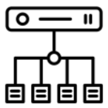
Hamid Ghasemirahni*, Alireza Farshin†, Mariano Scazzariello*,
Marco Chiesa*, Dejan Kostic*

* KTH Royal Institute of Technology
† NVIDIA

Stateful Network Functions

- Stateful Network Functions (NFs) are widely used in data centers
- Deploying NFs on commodity servers is a common use case
- Software Frameworks: **VPP** **FastClick** **BESS**
- A simple key-value store (i.e., hash table) is used to store per flow information

Firewall



Load Balancer

Rate Limiter

Deep Packet Inspector (DPI)

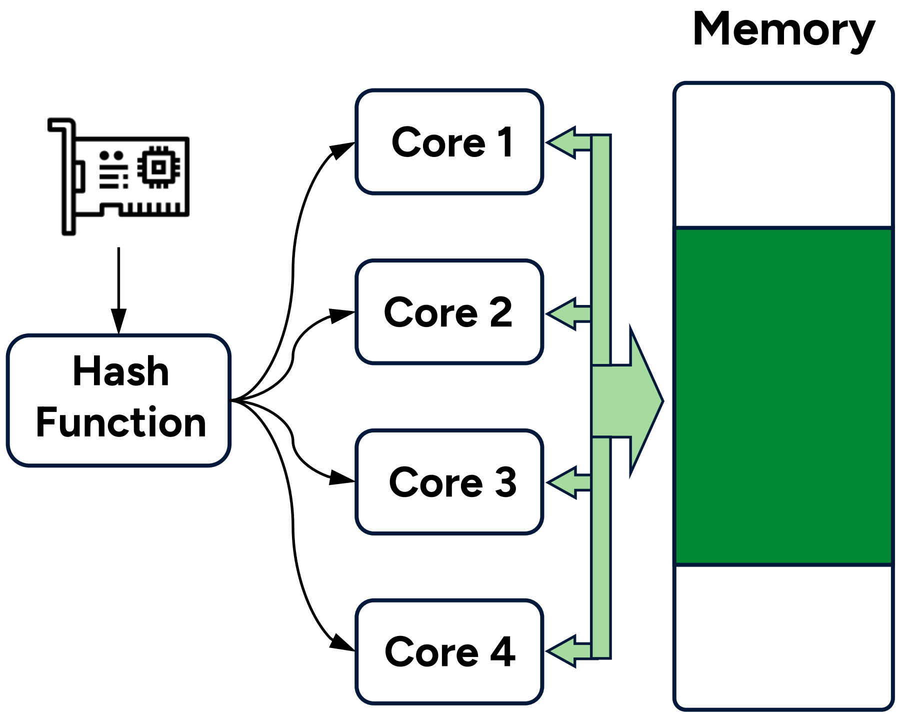
Access Control List (ACL)

Implementation Approaches

How to scale a chain of NFs to work on multiple cores?

Shared States

- ✅ Easy to implement
- ❌ Synchronization overhead



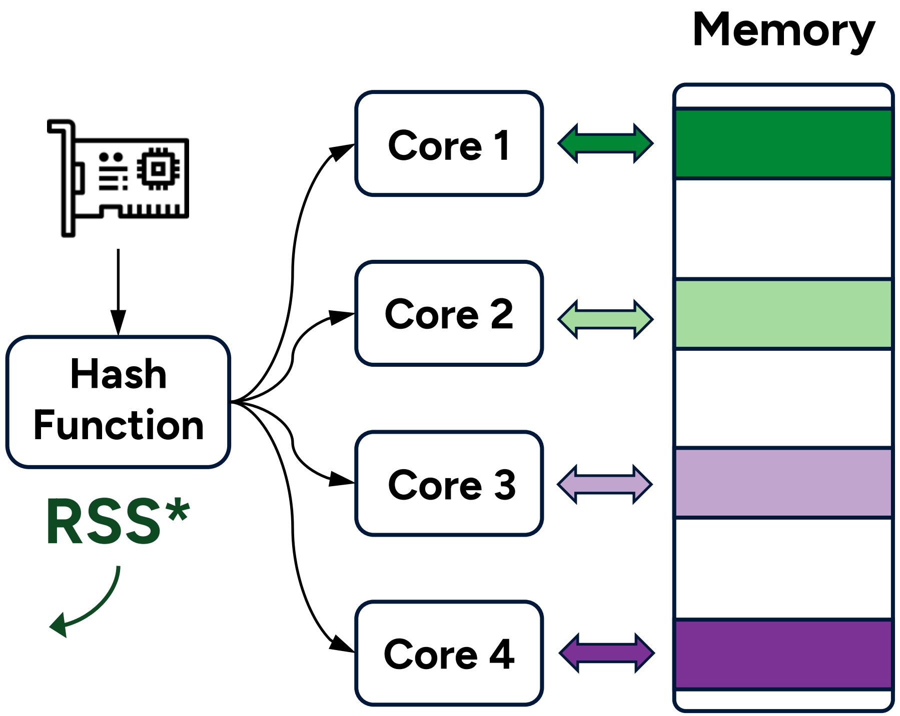
Implementation Approaches

How to scale a chain of NFs to work on multiple cores?

Shared Nothing

- ✅ High Performance
- ❌ How to dispatch packets?

Configurable hashing parameters



* Receive Side Scaling

How to Dispatch Packets

Different combination of packets attributes can be used as key in various stateful NFs (subset of 5-tuples)

Load Balancer	**Src. IP**	**Dst. IP**	**Src. Port**	**Dst. Port**	**Protocol**
Policer	Src. IP	**Dst. IP**	Src. Port	Dst. Port	Protocol
Port Scan Detector	**Src. IP**	Dst. IP	Src. Port	Dst. Port	Protocol

Utilize **symbolic execution** or **annotations** to detect the key for each NF

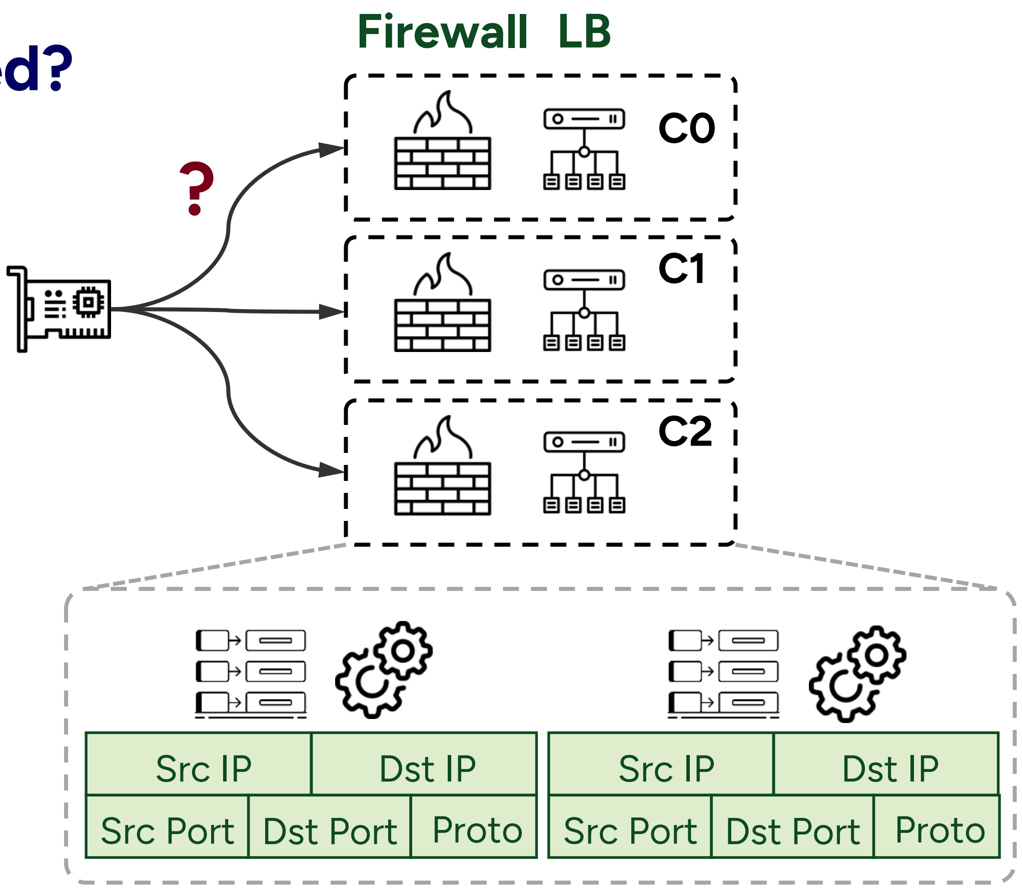
- ✗ Introduces coding limitations
- ✗ Extracted information is limited
- ✗ Depends on developers' knowledge

Existing Challenge

What if multiple stateful NFs are deployed?

Similar Flow Keys?

Dispatch by the same attributes as NFs key



Existing Challenge

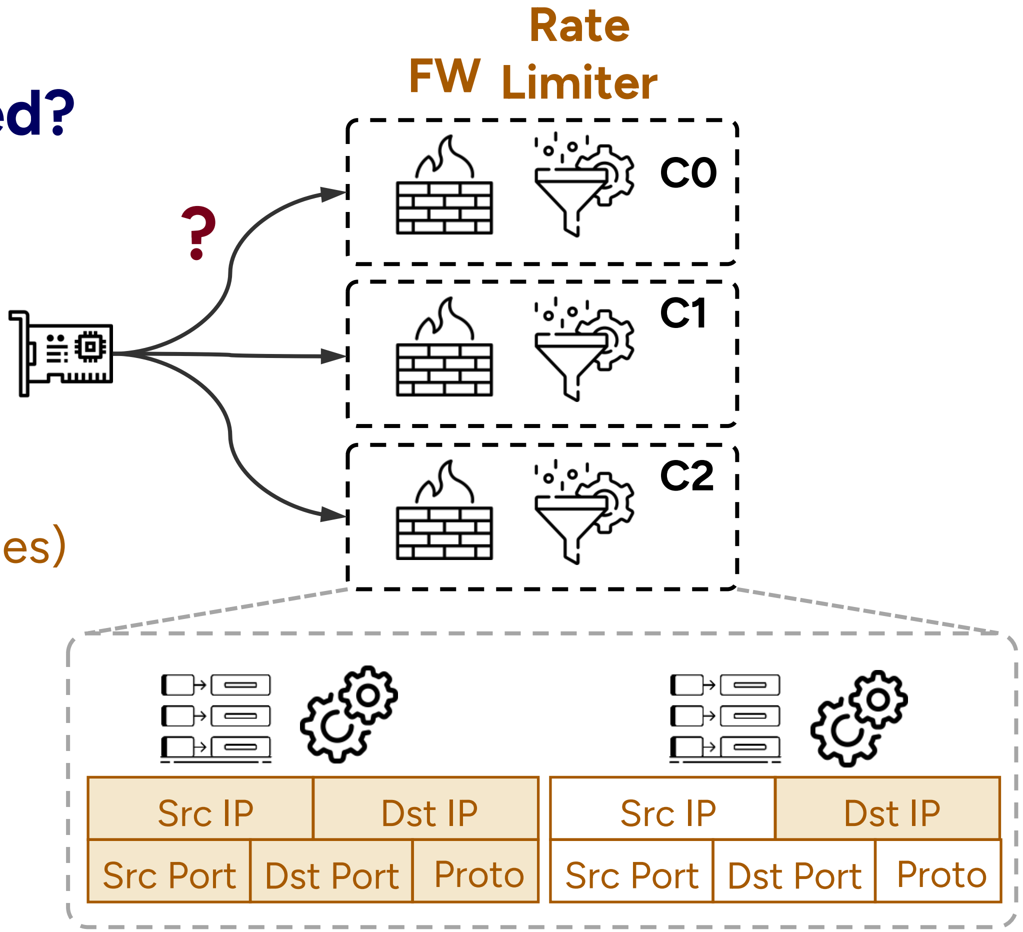
What if multiple stateful NFs are deployed?

Similar Flow Keys?

Dispatch by the same attributes as NFs key

None-Disjoint Flow Keys?

Dispatch by the intersection of attributes (common tuples)



Existing Challenge

What if multiple stateful NFs are deployed?

Similar Flow Keys?

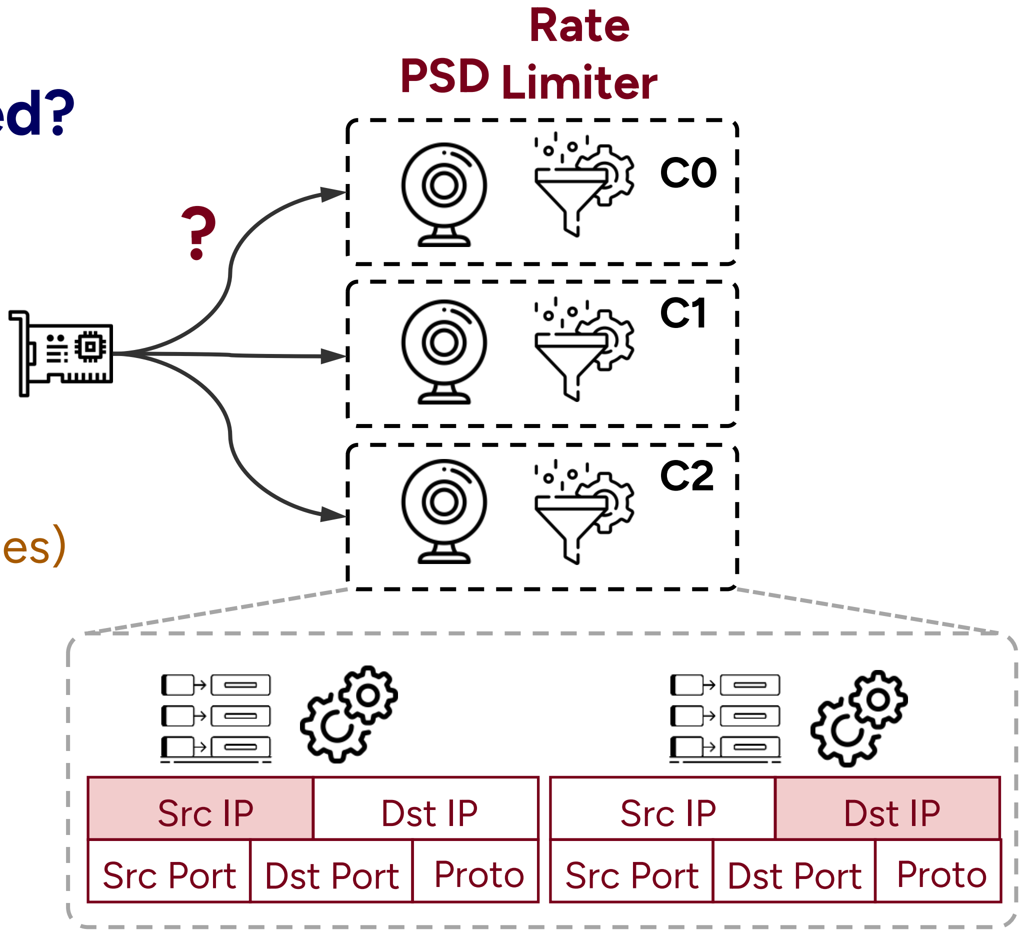
Dispatch by the same attributes as NFs key

None-Disjoint Flow Keys?

Dispatch by the intersection of attributes (common tuples)

Disjoint Flow Keys?

Impossible to achieve shared-nothing model



Existing Challenge

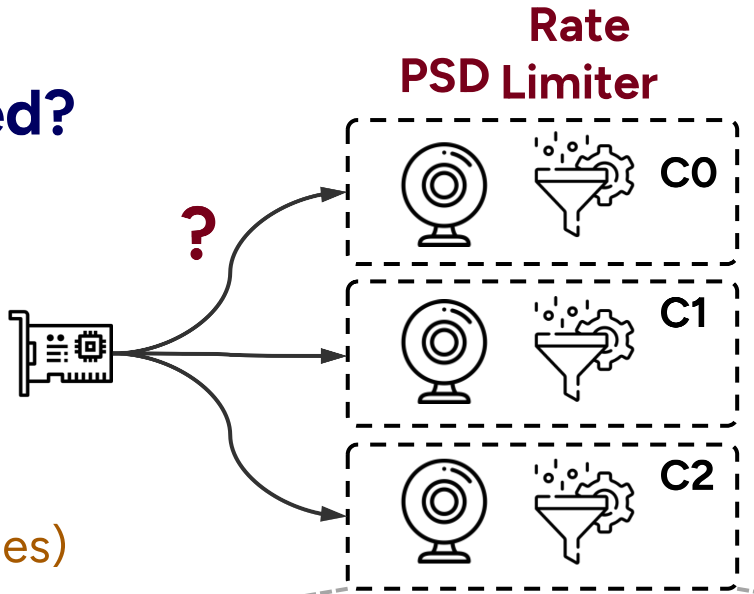
What if multiple stateful NFs are deployed?

Similar Flow Keys?

Dispatch by the same attributes as NFs key

None-Disjoint Flow Keys?

Dispatch by the intersection of attributes (common tuples)



Increasing the number of NFs in the chain makes the problem even more challenging

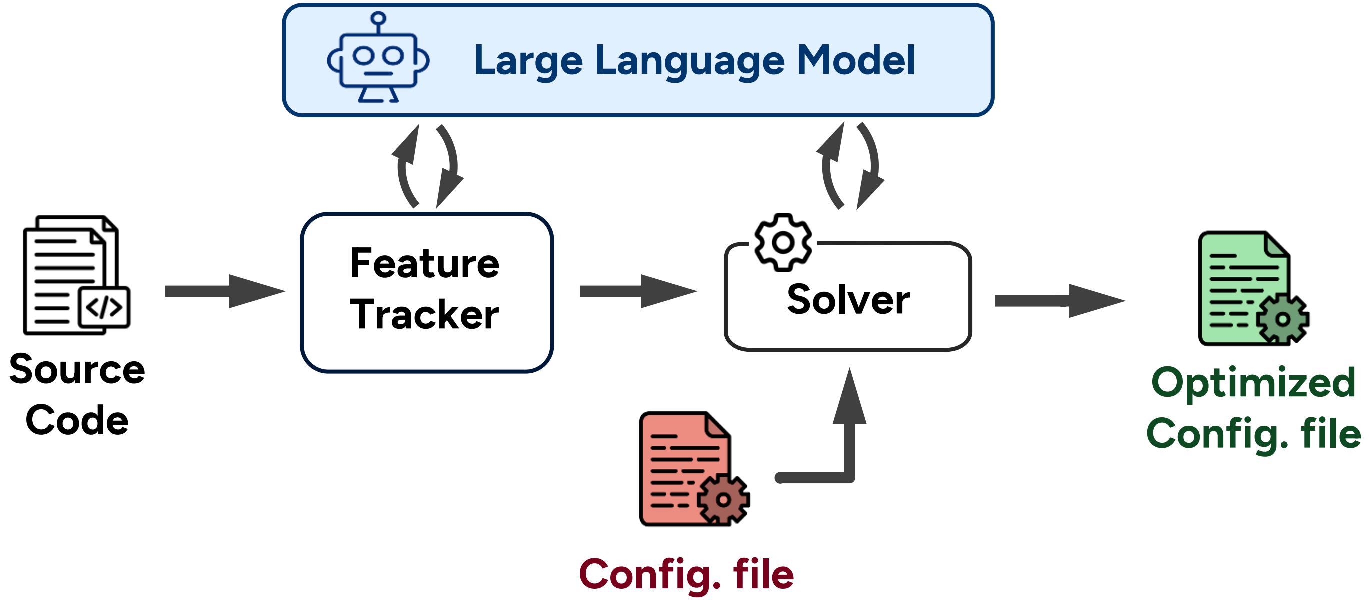
FlowMage

- LLMs have proven their ability in software engineering
- NFs' code bases are not large!

Leverage LLMs to deploy stateful NFs' chain efficiently!

- Framework agnostic and easy to integrate!
- Can extract several low-level and high-level features
- Low price

FlowMage



Feature Tracker

Triggers upon change in NFs' source code

Solver

Triggers on deploying a chain of NFs

FlowMage (Feature Tracker)

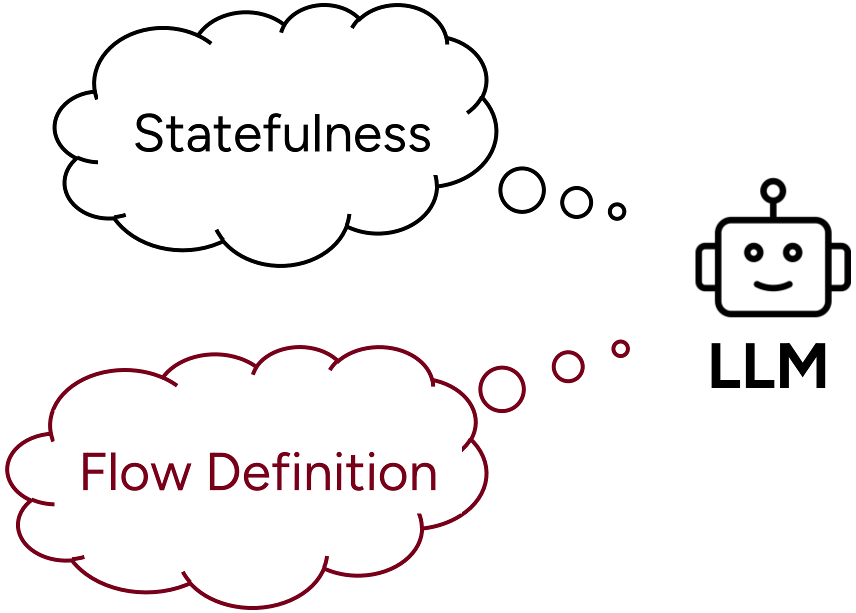
✅ **Leverage LLMs to extract high level features of NFs!**

1. Does a given NF store states per flow?

FlowMage (Feature Tracker)

✅ **Leverage LLMs to extract high level features of NFs!**

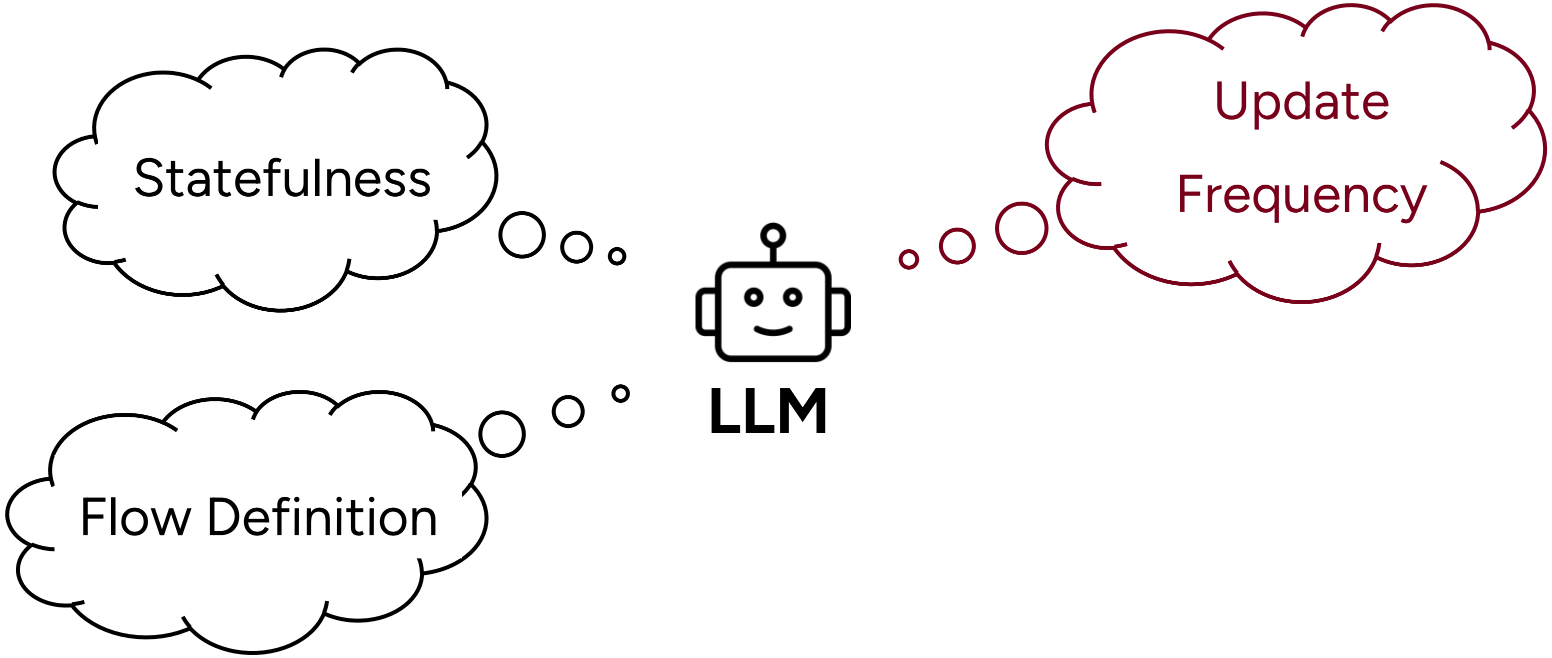
2. What are the packet attributes used for storing state?



FlowMage (Feature Tracker)

✅ **Leverage LLMs to extract high level features of NFs!**

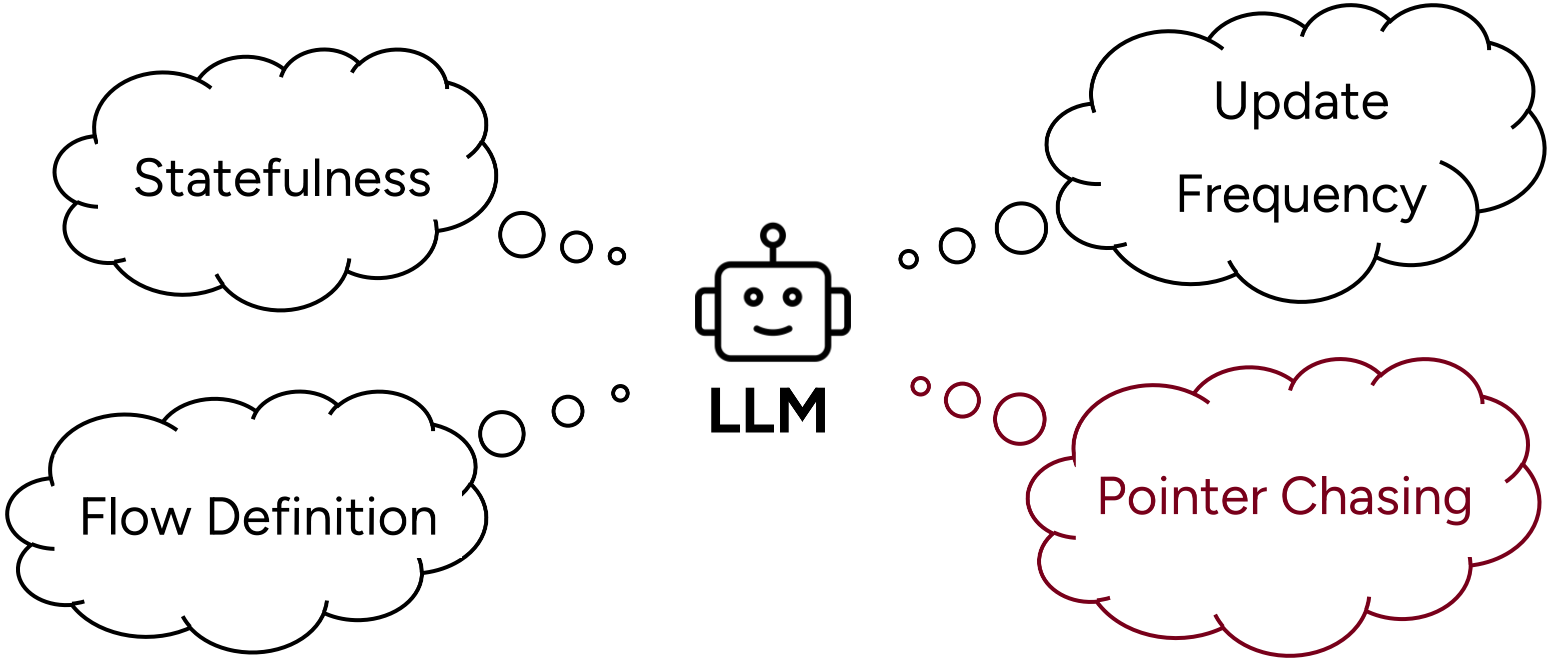
3. How often is state information for a flow being updated?



FlowMage (Feature Tracker)

Leverage LLMs to extract high level features of NFs!

4. Does the state information per flow contain pointers?



FlowMage (Feature Tracker)

System

Your task is to analyze C or C++ code of a network function provided by the user. For each network function, the user will ask: 1- if the NF is stateful. 2- how often the states are being updated. 3- the flow key of the NF. 4- If the state is stored in irregular memory.

User

Here is the C or C++ code of the network function to analyze:

```
class FlowIPNAT : public FlowStateElement<FlowIPNAT,NATEntryIN> , TCPHelper {
    public:
        const char *class_name() const override { return "FlowIPNAT"; }
        const char *port_count() const override { return "1/1"; }
        const char *processing() const override { return PUSH; }
```

Provide a JSON containing:

1. a key “statefulness” …

2. a key “intensity” …

3. a key “flow-key” …

4. a key “pointer” …

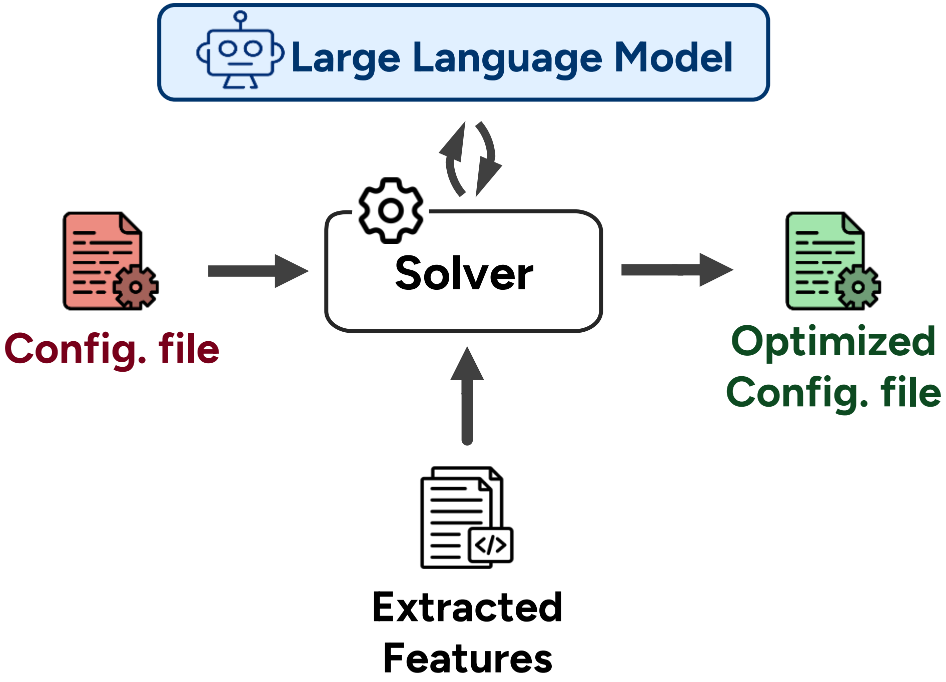
AI

```
"result": {
    "statefulness": “stateful”,
    "intensity": “per-packet”,
    "key":[“src_ip”, “dst_ip”,
           “src_port”, “dst_port”,
           “protocol”
    ],
    "pointer": “false”
}
```

FlowMage (Solver)

Transforms an input configuration file into its optimized counterpart!

- Formulates an optimization problem based on NFs feature.
- Utilizes LLMs to compare complexity of NFs' code base



Evaluation (LLMs Accuracy)

Examined 18 most frequent used NFs in FastClick and VPP

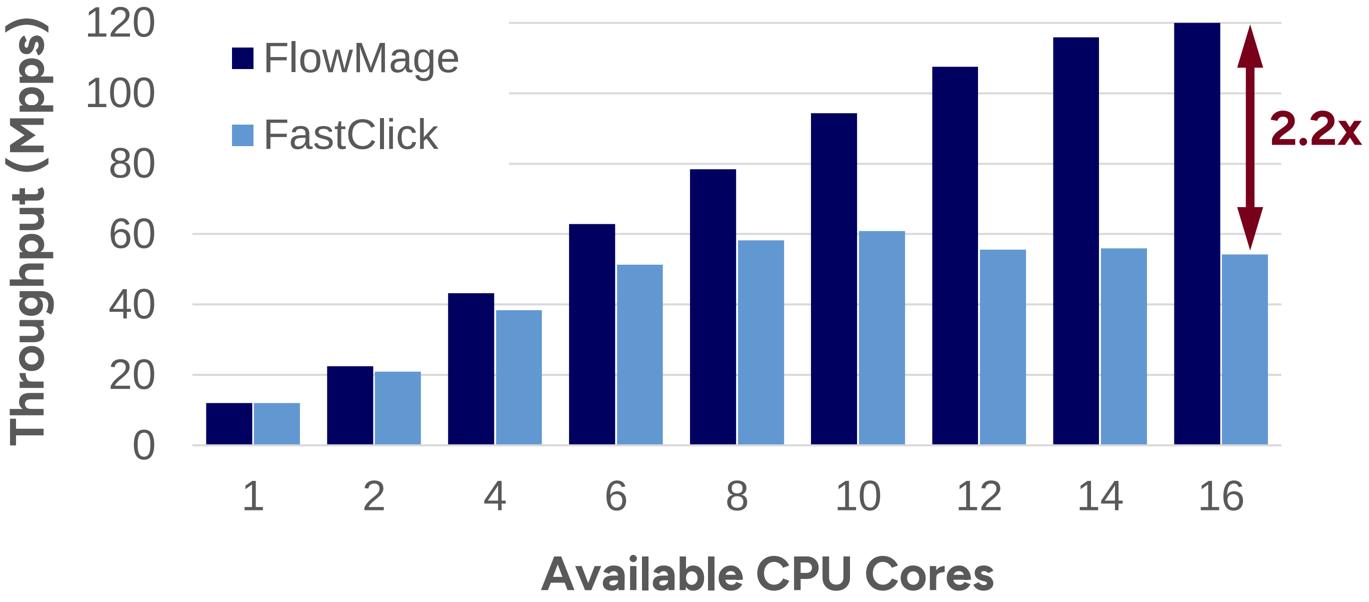
	Correct Assessments			
Feature	GPT-4 Turbo	GPT-3.5 Turbo	Gemini	Llama
Statefulness	18/18	15/18	15/18	11/18
Flow definition	10/10	8/10	8/10	4/10
Update Frequency	9/10	7/10	6/10	3/10
Pointer Chasing	9/10	6/10	6/10	4/10

Avg. token used per prompt: **7164 [min: 1168 , max: 31570]**

[Check the paper for the detailed report]

Evaluation (System Performance)

A Chain consisting of a *Policer* and a *Source IP Tracker*



Performance gain increases in more complex scenarios! *(check more evaluations in the paper)*

Conclusion

FlowMage: Leveraging LLMs to efficiently deploy a chain of stateful NFs

- Easy to integrate into the existing frameworks
- **2.2x** higher throughput deploying a simple NFs chain scenario

Future Work:

- In-context learning or fine-tuning of LLMs to:
  - Improve accuracy
  - Extract more detailed information from NFs source code
- LLMs understand low level syntax such as LLVM IR Bitcode
  - Estimate system level performance metrics for NFs

Check Paper Here

hamidgh09/FlowMage

European Research Council
Established by the European Commission
Horizon 2020
European Union Funding
for Research & Innovation