A Solution for Goal-oriented Policy Refinement in NFV-MANO Systems

Michel Bonfim, Fred Freitas, Stênio Fernandes

Federal University of Pernambuco (UFPE)

NMRG 57th meeting
IETF 106, Singapore
Introduction

• **Policy Refinement**: the process of transforming high-level policies into directly enforceable, low-level policies.

• **Problem Statement**: A fully automated refinement process in NFV systems is still an open issue.

• **NSPlanner**: A goal-oriented policy refinement procedure for NFV-MANO systems.
  – It uses a well-funded HTN planner to perform Goal-oriented policy refinement procedures.
  – It proposes the use of one ontology in OWL 2, called **Onto-Planner.**
NSPlanner Architecture

Operator
- Thresholds

NFV Orchestrator
- NS goals

NSPlanner

Threshold API

Threshold Management Module

Goal API

Goal-oriented Policy Management Module

Conflict Detection Module

Planner API

Planner Generation Module

Planning Problem

Planning Domain

Onto-Planner

Onto-Planner (In-Memory)
High-level Goal Language

1 Language -> <Elements> must receive <Level> <Attributes>
   Elements -> <Element> | <Element><Connective><Elements>
3 Element -> vnf-member-index
   Level -> high | medium | low
5 Attributes -> <Attribute> | <Attribute><Connective><Attributes>
   Attribute -> resiliency | manageability | security | performance
7 Connective -> and

EXAMPLE:

1 and 2 must receive high performance and resilience
Describing Goals
Policy Refinement Procedure

1. It instantiates a copy of Onto-Planner in memory;
2. It extracts the current state of the domain (actions, metrics, alarms) and requested goal information;
3. It builds the planning problem document from above data;
4. It performs the policy refinement;
5. It creates the enforceable policies (ECA rules) and enforceable alarms descriptions into the Onto-Planner copy;
6. It performs DL reasoning to provide inconsistency verification in Onto-Planner copy.
Describing Policy Rules
Managing Policy Conflicts

Operator
- Thresholds

NFV Orchestrator
- NS goals

NFV Orchestrator
- NS goals

Threshold API

Goal API

Planner API

Threshold Management Module

Goal-oriented Policy Management Module

Conflict Detection Module

Planner Generation Module

Planning Problem

Planning Domain

Onto-Planner

Onto-Planner (In-Memory)
Managing Policy Conflicts

- **Goal Conflicts:**

\[\text{Goal}(g1) \land \text{Goal}(g2) \land \text{differentFrom}(g1, g2)\]
\[\land \text{hasNS}(g1, ns1) \land \text{hasNS}(g2, ns2) \land \text{id}(ns1, id1) \land \text{id}(ns2, id2)\]
\[\land \text{hasVNF}(g1, vnf1) \land \text{hasVNF}(g2, vnf2)\]
\[\land \text{hasMemberVNFIndex}(vnf1, index) \land \text{hasMemberVNFIndex}(vnf2, index)\]
\[\land \text{Attribute}(a) \land \text{hasAttribute}(g1, a) \land \text{hasAttribute}(g2, a)\]
\[\rightarrow \text{intersectsWith}(g1, g2)\]

- **ECA Rule Conflicts:**

\[\text{PolicyRule}(r1) \land \text{PolicyRule}(r2)\]
\[\land \text{hasPolicyEvent}(r1, pe1) \land \text{hasPolicyEvent}(r2, pe2)\]
\[\land \text{hasEvent}(pe1, e) \land \text{hasEvent}(pe2, e)\]
\[\land \text{hasPolicyCondition}(r1, pc1) \land \text{hasPolicyCondition}(r2, pc2)\]
\[\land \text{hasParameterKey}(pc1, key) \land \text{hasParameterKey}(pc2, key)\]
\[\land \text{hasParameterValue}(pc1, value) \land \text{hasParameterValue}(pc2, value)\]
\[\land \text{hasPolicyAction}(r1, pa1) \land \text{hasPolicyAction}(r2, pa2)\]
\[\land \text{hasAction}(pa1, a1) \land \text{hasAction}(pa2, a2) \land \text{conflictsWith}(a1, a2)\]
\[\rightarrow \text{conflictsWith}(a2, a1)\]
We developed an NSPlanner prototype in Java:
- We use Spring Boot to implement the RESTful APIs;
- We use OWL API.

The Conflict Detection Module uses Hermit Reasoner;
The Planner Management Module uses SHOP2 as HTN Planner:

Use case: we implemented a planning domain model for the Resilience Attribute
NSPlanner performs and scales well.
Contributions to NMRG

1. Enables goal-oriented policy management:
   - NSPlanner focus on intent-* specific aspects, since goal policies describe desired states in an environment, not a sequence of actions.

2. Provides a decomposition logic to resolve intent to relevant service and select appropriate mgmt function(s).
Thank you! Questions?

Michel Bonfim
<msb6@cin.ufpe.br>