ETSI GS ENI 034 V4.1.1 (2024-07)

Experiential Networked Intelligence (ENI); Requirements and Detailed Procedure of Network Policy Conflict Detection

Disclaimer

The present document has been produced and approved by the Experiential Networked Intelligence (ENI) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
DGS/ENI-0034v411_Confli_Detect

Keywords
conflict detection, OAM, policy management

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Experiential Networked Intelligence (ENI).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Executive summary

The present document specifies a high-level functional abstraction of the process of ENI Intent policy Multi-Stage translating in an ENI system in terms of Functional Modules, Internal Reference Points and working pipelines.

Introduction

The present document defines a high-level functional abstraction of Network Policy Conflict Detection for ENI Intent Policies. The organization of the present document is as follows. Clause 1 defines the scope of the present document. Clauses 2 and 3 provide normative and informative references and definition of terms, respectively. Clause 4 provides an informative overview of Network Policy Conflict Detection, including its motivation, benefits, important concepts and an overview of its Functional Modules. Clause 5 defines important design principles of the processing. Clause 6 provides some use cases of Network Policy Conflict Detection. Clause 7 describes areas of future work.

1 Scope

The present document provides additional information concerning Network Policy local conflict detection for ENI Intent Policies. The present document expands on the work done in ETSI GS ENI 005 [i.2], clause 6.3.9.6.3, to provide additional requirements and procedures to ensure that a new network policy will not conflict with any currently deployed network policies in the same administrative domain. The present document is only intended for Network Policies that are structured as ENI Intent Policies and which meet the requirements defined in clause 4.3.1.

The present document also describes the input(s), output(s), Internal Reference Points, and functionality of every step in the Network Policy local conflict detection process.

If network policies with potential risks are dispatched, they may lead to various errors in the network and cause instability and harm. However, as the scale of the network increases, the difficulty and cost of detecting and correcting network errors also increases. Therefore, Network Policy Conflict Detection is implemented after Policy Validation and Policy Rewriting. This will potentially save time and reduce misconfigurations. Consequently, the stability and availability of the system will be increased.

The present document will encompass research and investigation activities that will address network policy conflict in IP networks at the first stage. Subsequent efforts may extend the work into telecommunication networks.

2 References

2.1 Normative references

Not applicable.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI GR ENI 004: "Experiential Networked Intelligence (ENI); Terminology".

[i.2] ETSI GS ENI 005 (V3.1.1): "Experiential Networked Intelligence (ENI); System Architecture".

[i.3] ETSI GR ENI 010: "Experiential Networked Intelligence (ENI); Evaluation of categories for AI application to Networks".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GR ENI 004 [i.1], ETSI GS ENI 005 [i.2] and the following apply:

black hole: place in the network where incoming or outgoing traffic is unexcepted discarded, so that the data did not reach its intended recipient

forwarding loop: abnormal phenomenon in which a packet reaches the same device twice during the forwarding process

forwarding model: edge-labelled directed graph for representing network forwarding behaviour

hit domain: set of packets that satisfy the Action rules

match domain: set of packets that match the routing table, ACL or NAT

network invariant: packet forwarding constraint that needs to be satisfied in any computer network

network policy: specific type of policy that affects network behaviour and can be directly understood and executed by network devices within the Assisted System

self-change: unpredictable changes due to Network Events such as node faults

snapshot translation: kind of data translation in which the raw routing table (also known as flow table), ACL and NAT of each device in the AS are translated into FW, ACL and NAT rules, respectively

User-Defined Network Policy: user-defined packet forwarding constraint that needs to be satisfied in a particular computer network to comply with applicable network invariants

waypoint: device that the packets need to pass through during the forwarding process in addition to the source and destination

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GR ENI 004 [i.1], ETSI GS ENI 005 [i.2] and ETSI GR ENI 010 [i.3] apply.

4 Overview of Network Policy Conflict Detection

4.1 Introduction

This clause provides an informative introduction to Network Policy Conflict Detection in the ENI System Architecture. Clause 4.2 describes the background and motivation of Network Policy Conflict Detection, and then provides a high-level description of Network Policy Conflict Detection in the ENI System, including which ENI Functional Block it is deployed in and what kind of policy it can and cannot validate. Clause 4.3 describes the external requirements for Network Policy Conflict Detection, including the requirements for the network policy, the maximum input frequency of Network Policy Conflict Detection, and the architecture of the Assisted System. Clause 4.4 describes the functional architecture of Network Policy Conflict Detection in terms of Conflict Detection Functional Modules.

4.2 Description of Network Policy Conflict Detection

In an increasingly interconnected world, network traffic is increasingly diverse and demanding, whether it is communication between small everyday devices on LANs or communication between large global data centres on the Internet. This diversity in network traffic has driven the design and widespread adoption of a new open network architecture called Software-Defined Networking (SDN). SDN is built upon programmable network switches, which enable the separation of the network control plane from the data plane. This separation allows the control plane to customize the data plane with User-Defined Policies that users want the network forwarding behaviour to meet.

However, as network background traffic continues to evolve and new User-Defined Policies emerge, some of the User-Defined Policies are likely to be violated after some forwarding rule changes due to rule conflicts or information isolation between different applications. In addition to User-Defined Policies, network policies also can contain Network Invariants that cannot be violated, including not having network Forwarding Loops and Black Holes. Network invariance is one of the most easily violated network policies in complex computer networks, and it can easily be caused by misconfiguration, hardware or software problems. Network Policy Conflict Detection is part of Intent Processing in the Policy Management Functional Block, which aims to automatically detect network policy correctness through telemetry information. More specifically, the Network Policy Conflict Detection with the forwarding rules issued by the ENI System as input, maintains a network forwarding model to validate whether the installed forwarding rules would violate Network Invariants or User-Defined Policies by extracting a specific forwarding slice from the forwarding model.

In this context, the Network Policy Conflict Detection needs to process network invariants and user-defined network policies. More specifically:

- Network Invariant: a property inherent to an object (e.g. a router interface) in a computer network that cannot be violated. Examples include network Forwarding Loops and Black Holes.
- User-Defined Network Policy: this kind of policy represents additional user requirements for network forwarding behaviour, including reachability, isolation and so on.

4.3 External Requirements for Network Policy Conflict Detection

4.3.1 Network Policy

A detailed description of a User-Defined Network Policy is as follows:

For the purposes of the present document, a network forwarding model is used to represent the packet forwarding behaviours of a data plane, which is an edge-labelled directed graph with a label on each directed edge to represent the set of packets that can pass through this edge.

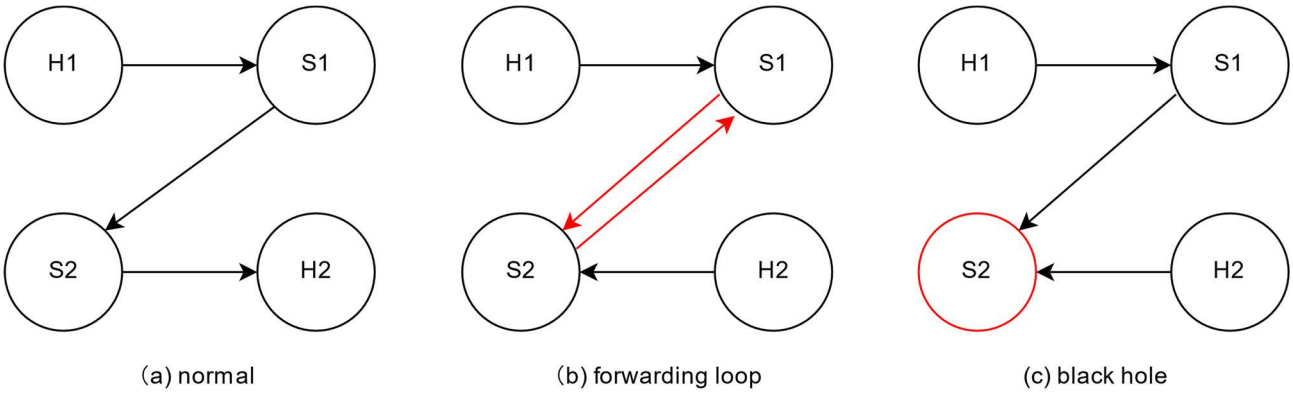

Figure 4-1: The Violation example of Network Invariants

Figure 4-1 illustrates two classic types of Network Invariant violations. The presented network has a total four devices, where S1 and S2 denote programmable switches in the data plane, and H1 and H2 denote two hosts in the network.

NOTE 1: The label is omitted since the set of packets that can pass on all edges in the diagram is the same.

- Forwarding Loop violation. A device forwards a packet back to itself or to a previous node in the packet's path. Figure 4-1(b) shows an example of a Forwarding Loop, where a Forwarding Loop S1-S2-S1 occurs when a packet forwarded from S1 to S2 is forwarded by S2 to S1 again.
- Black Hole violation. A black hole is a place in the network where incoming or outgoing traffic is unexpectedly discarded, so that the data did not reach its intended recipient. Figure 4-1(c) shows an example of a Black Hole when packets from S1 and H2 are forwarded to S2, but S2 does not forward them to other devices or intentionally drops them.

NOTE 2: All hosts can be considered as the end point of packet forwarding, so they are not considered as Black Hole.

User-defined policies can include network invariants to further constrain packet forwarding behaviour. Each User-defined Network Policy contains at least the following components:

NOTE 3: Users can expand the composition of user-defined policies according to their own needs.

- Source: The identification code of the source device that the packets affected by this policy are sent from.

NOTE 4: Identification code is one of MAC address, IP address, port or any kind of combination of above.

- Destination: The identification code of the destination device that the packets affected by this policy are sent to.
- Match: The set of packets affected by this policy.

NOTE 5: Match should consist of some parameters such as time, DSCP, IP address.

- Waypoints: Waypoints of packets affected by this policy during forwarding from Source to Destination.
- Count: Specifies the number of paths through Waypoints to the Destination for packets from the Source matched by the Match.

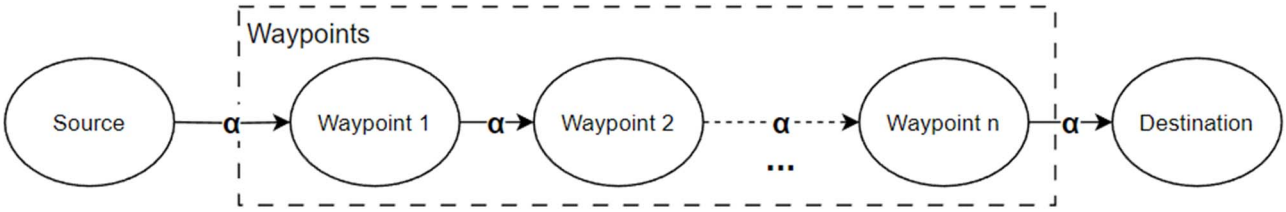

Figure 4-2: An example of User-defined Network Policy

Figure 4-2 shows an example of a User-defined Network Policy in the form of a forwarding diagram, where waypoints denote programmable switches in the data plane, while Source and Destination denote two hosts in the network. This policy specifies that for packets coming from the "source" and matching condition α, a path needs to exist in the network making them go through waypoint 1 - waypoint n to reach the "destination".

4.3.2 Input Frequency and Assisted System

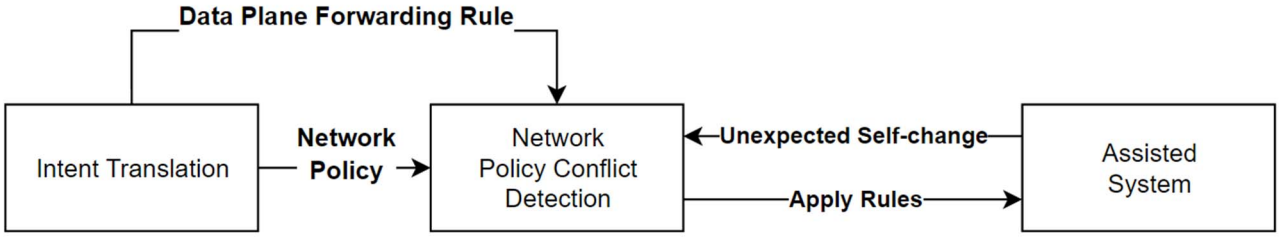

Figure 4-3: Detecting Network Policy Conflict

Figure 4-3 shows how the Network Policy Conflict Detection interacts with the Intent Translation Functional Block upstream and the data plane downstream (Assisted System). Whenever there are new forwarding rules to be sent down to the data plane, these rules need to be sent to the Network Policy Conflict Detection Functional Block before any new intent policies are processed. More specifically, the Network Policy Conflict Detection simulates rules in its internal Network Forwarding Model, and sends them down to the real data plane only when they do not violate any Network Policy.

To ensure the accuracy of the verification results, the Network Policy Conflict Detection Functional Block determines whether there is an unexpected Self-Change after the previous Rule is sent to the Assisted System before the next batch of rules can be verified. Therefore, after the real data plane has finished delivering these rules, the data plane returns any unexpected Self-Changes to the Network Policy Conflict Detection Functional Block (via the appropriate external Reference Point between the Assisted System and the Policy Management Functional Block) to ensure that its internal Network Model is consistent with the real data plane.

The Assisted System responds to commands from the ENI System that change the forwarding behaviour of the data plane using ENI Policies. ENI Policies can also change monitoring points in the network 5 [i.2]. For example, the Assisted System receives an API that contains one or more ENI Policies, and sends those ENI Policies to the appropriate element (e.g. a Manager or Controller). Similarly, the Assisted System needs to provide appropriate telemetry to the ENI System (via APIs) that contain any changes to the data plane.

NOTE: Although the forwarding behaviour of the data plane in general can only be changed by either ENI Policies or policies that the ENI System knows about, it can also change unpredictably due to Network Events such as node faults.

4.4 Functional Workflow

Figure 4-4 depicts the functional workflow of the Network Policy Conflict Detection, which consists of three layers:

Parser: The Parser is responsible for the initial parsing of the outside input. This typically consists of Network Policy Parser, Network Config Parser and Network Forwarding Rule Parser (though other parsers can also be included).

- **Network Policy Parser:** The Network Policy Parser takes the upstream output of the User-defined Network Policy as input and constructs the User-defined Network Policy as the five-tuple described in clause 4.3.1.
- **Network Config Parser:** The Network Config Parser reads network topology and configuration information from the data plane for use by the Model Builder.

NOTE 1: Network topology and configuration information is obtained by CLI and RESTConf.

- **Network Forwarding Rule Parser:** The Network Forwarding Rule Parser takes the arrived forwarding rules or the returned Self-Changes as input and converts them into a specific format to the Model Updater.

NOTE 2: The recommended input and output formats for each of these Parsers are described in detail in clause 5.2.

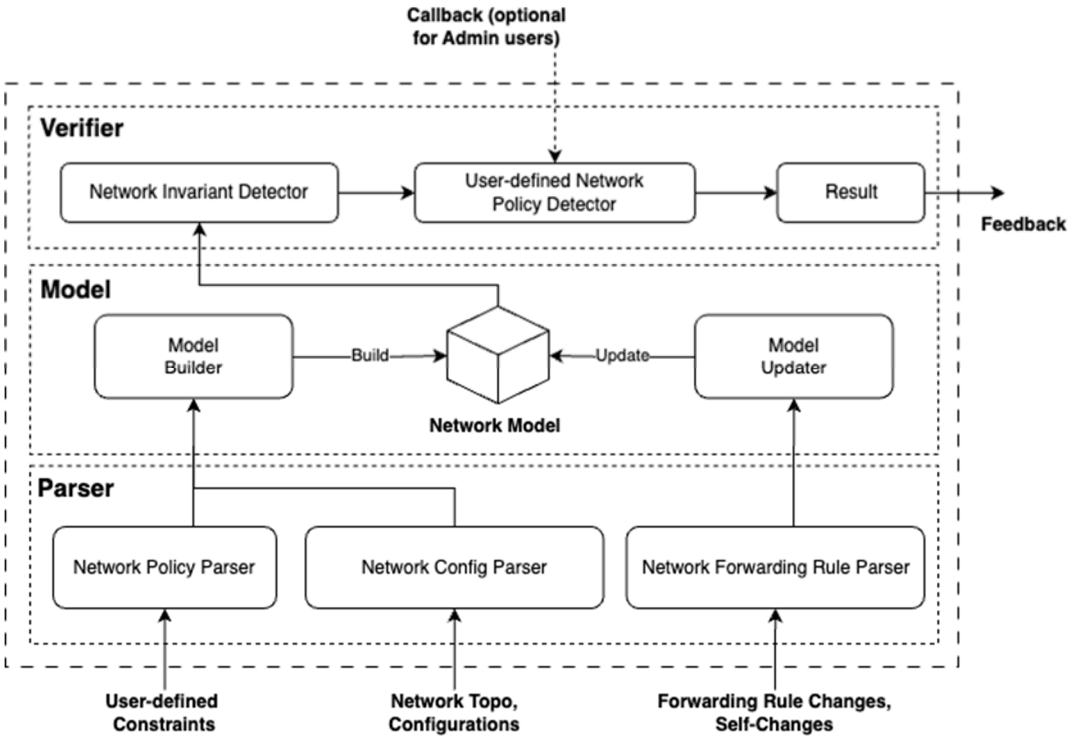

Figure 4-4: The Functional Architecture of Network Policy Conflict Detection

Model: The Model is responsible for constructing a forwarding model consistent with the data plane forwarding state in the form of a directed graph with a label on each directed edge representing the set of packets that can pass through that edge.

- Model Builder: The Model Builder initializes the network model using the information obtained from the Network Policy Parser and the Network Config Parser.
- Model Updater: Whenever new Network Forwarding Rules or data plane Self-Change arrive; Model Updater applies them to the Network Model to simulate their affect in the real data plane.

NOTE 3: The Network Model updating progress is detailed in clause 5.3.

Verifier: The Verifier is responsible for detecting and locating network policy conflicts as well as returning the results.

- **Network Invariant Detector:** The Network Invariant Detector aims to detect the presence of any property of a networking object that does not change during an operation in the Network Model and locate them (if any).
- **User-Defined Network Policy Detector:** The User-defined Network Policy Detector aims to detect if any of the user-defined policies is violated in the Network Model and locate all violated user-defined policies. User-defined network policies are defined by an appropriated element (e.g. end user, application, OSS, BSS, or Orchestrator) as described in ETSI GS ENI 005 [i.2]. To define new user policies, user should create a customized Callback function and hook it with user-defined Policy Detector.
- **Callback (Function):** A Callback function is a function that is passed as an argument to another function. This allows a function to call another function. Discriminant to judge whether there exists a specific user pre-banned network forwarding behaviour in the network.

NOTE 4: Network Policy Conflict Detection is only responsible for detecting conflicts and feeding back the problems, not for fixing them.

NOTE 5: A user such as an admin should create a Callback function.

5 Network Policy Conflict Detection Reference Working Pipeline

5.1 Introduction

The following clauses define reference design principles and describe necessary functionalities in Network Policy Conflict Detection. The definition and description will be grouped by different steps in Network Policy Conflict Detection, respectively.

NOTE: In Release 4, there are four general steps in Network Policy Conflict Detection, they are pre-processing, action simulation, conflict detecting and locating, and result feedback.

As described in clause 4.4, working as a Network Policy Conflict Detection Functional Block relies on the Network Policy, Network Topology and Configuration and Forwarding Rules and Self-Changes received by the Parser layer. This clause specifies the format requirements for all of these inputs and what pre-processing will be done on them for the Model layer to use.

5.2 Input

5.2.1 User-defined Network Policy as Input

The User-defined Network Policy needs to meet the following requirements:

- Match is a union of a set of packets that are represented by their Header Fields.
- Source is a string that uniquely identifies a device, either by its name or another custom serial number, etc..

- Destination is the same as Source and is a string that uniquely identifies a device.
- (optional) Waypoint is an array of string, each of which uniquely identifies a device.
- (optional) Count is a number indicating how many paths from source to destination.

The Parser is responsible for translating the five elements of the User-Defined Network Policy that arrive from the Assisted System into a data structure that the Model Layer can understand.

5.2.2 Network Topo & Configuration as Input

The network topology information provided to the Parser layer should consist of:

- Node information of the network.
- Link information of the network.

For Network Invariant and forwarding-related User-defined Network Policy, no additional network configuration information is required. However, if there is a need to extend the scope of verification, additional network configuration information may be required as input. For example, if the need is to verify the network bandwidth load, the network link bandwidth-related configuration information is required.

5.2.3 Forwarding Rule Changes or Self-Changes as Input

The forwarding behaviour of a computer network is determined by the forwarding rule on each device collectively. However, other factors may affect this, such as access control lists, protocol precedence, and quality of service policies. Therefore, any change of forwarding rule should be informed to the Parser layer to help the Model layer to build a Network Model that is consistent with the actual network forwarding behaviour.

Each forwarding rule change indicates the insertion or deletion of a forwarding rule on a device, and therefore needs to contain the following information:

- Device: The device(s) affected by this rule change.
- Match: refer to the definition of Mathematics in clause 5.2.1.
- Action: The action to be applied, consisting of forward, drop, buffer, duplicate, or modify. In addition, it may be possible to add other actions, such as log, depending on the network and the device.
- Priority: The priority of the rule being inserted or deleted, with higher Priority indicating execute first and override any similar rules (e.g. if rule 1 says set foo to 10 with priority 1 and rule 2 says set foo to 3 with priority 10, only rule 2 will be executed).
- Type: Indicates whether the change is a delete rule or an insert rule.

The Parser Layer is responsible for translating the above forwarding rule change information into a data structure that the Model Layer can understand.

Although theoretically the forwarding behaviour of the network does not change when the forwarding rule is not modified, in reality many unexpected Self-Changes can affect the network forwarding behaviour, including but not limited to: link fault, device fault, etc. When unexpected Self-Changes occur, the Assisted System should inform the Parser Layer what changes the data plane has undergone, and the Parser Layer needs to translate these changes into information that the Model Layer can understand to ensure the consistency of the Network Model with the actual network. For example, when a link fault occurs, the Parser Layer needs to inform the Model Layer to disconnect the corresponding edges in the Network Model.

5.3 Forwarding Rule Changes Simulation

The Model Layer represents the forwarding behaviour of the data plane by constructing an edge-labelled directed graph to represent the forwarding behaviour. Each node in the graph represents a device and each edge represents a unidirectional link.

The core of maintaining the consistency of the forwarding graph with the actual forwarding behaviour of the data plane is to simulate the insertion or deletion of each forwarding rule. Note that multiple rules on the same device may have overlapping match fields, and packets will take the action of the rule with the highest priority. Given a User-defined Network Policy r, some packets in r.match (i.e. the match field of r) may not "hit" rule r due to the presence of some higher-priority rules. To represent the headers that actually "hit" a rule, the hit field for each rule r is defined as:

$$r.\,\mathrm{hit} \triangleq \neg(\vee_{r'.\mathrm{prio}>r.\mathrm{prio}}\ r'.\,\mathrm{match}) \wedge r.\,\mathrm{match}$$

NOTE 1: The above equation may be interpreted as "A routing rule r is defined as not having any other rules r' with a higher priority that also match, and r itself also matches".

After calculating the hit field of rule r, the Model Layer simulates Rule insertion or Rule deletion by the following steps:

- Rule insertion: incorporates r.hit from the label on the edge (Device, Action) directly affected by r and subtracts r.hit from the label in the other edge (Device, *).

NOTE 2: The * here represents all the devices adjacent to the Device.

- Rule deletion: subtracts r.hit from the labels on edges (Device, Action) directly affected by r, and expands the labels on edges directly affected by other rules whose priority is lower than r while expanding their hit fields.

5.4 Network Policy Conflict Detecting and Locating

5.4.1 Parser Layer Detecting and Locating

The Network Policy Conflict Detection Functional Block validates network policies in two places: the Parser Layer and the Verifier Layer.

When a new User-defined Network Policy arrives, the Parser Layer first needs to verify the syntactic correctness of the User-defined Network Policy. Assuming that this has been done, it then needs to verify and locate the following three types of conflicts:

- Multiple reachability policies form loops with each other.
- Reachability policy and isolation policy conflict with each other.
- The source of two reachability policies with overlapping Match is the same but the Destination is different.

NOTE: This situation conflicts only because Conflict Detection only supports unicast policies.

5.4.2 Verifier Layer Detecting and Locating

Based on the forwarding graph provided by the Model Layer, the Verifier Layer verifies whether the network forwarding behaviour violates any User-defined Network Policy (including any network invariants it may contain) by executing a Depth-First Search (DFS) traversal of the forwarding graph starting from its Source. For a Reachability policy, it is validated when all packets in the Match can reach the Destination through waypoints; for Isolation, it is validated when all packets in the Match cannot reach the Destination.

5.5 Result Feedback

5.5.1 Parser Layer Feedback

The Network Policy Conflict Detection Functional Block also returns result feedback to upstream users at the Parser and Verifier layer.

When a User-defined Network Policy conflicts with another policy (i.e. ENI Policy or another User-defined Network Policy), the Parser Functional Block needs to return feedback containing at least the following information:

- Which policies conflict with the newly arrived policies.

- What type of conflict is generated.

5.5.2 Verifier Layer Feedback

When there is a Network Policy verification difference, the Verifier needs to return feedback containing at least the following information:

- Which devices constitute Forwarding Loops on which packets.
- Which devices generate Black Holes for which packets.
- Which policies have not passed validation.

6 Use Cases

6.1 Network Policy Detection in VxLAN

6.1.1 Use Case Context

In a large-scale enterprise with a global presence, the IT infrastructure is a critical component for day-to-day operations and long-term strategic initiatives. The enterprise encompasses a diverse array of departments, including research and development, finance, marketing, customer support, and IT services. Each department has unique network requirements in terms of access, security, and performance.

The IT department, tasked with the continuous improvement of the network infrastructure, has identified the need to enhance network segmentation and flexibility to better support the varying needs of these departments. The existing network architecture, while robust, is not agile enough to adapt to the rapidly changing demands of the modern digital environment. This led to the decision to deploy a Virtual eXtensible LAN (VxLAN) network.

VxLAN technology offers several advantages over traditional networking solutions. It allows for the creation of a large-scale overlay network with improved segmentation capabilities. This is particularly beneficial for the enterprise as it ensures secure and isolated communication paths for different departments and applications. VxLAN also facilitates more efficient utilization of network resources and simplifies network management, a critical factor for the IT team that manages the complex network infrastructure spanning multiple geographical locations.

The network spans across multiple data centres located in different regions of the world, as well as various cloud environments. This heterogeneous environment hosts a multitude of applications and services, ranging from internal business applications, customer-facing services, to data-intensive research and development workloads. The deployment of VxLAN networks is seen as a strategic move to unify these disparate elements under a cohesive, flexible, and secure network umbrella.

The overarching goal of the VxLAN deployment is not only to address the current networking challenges but also to lay a foundation for future growth and innovation. By implementing VxLAN, the enterprise aims to boost its network efficiency, enhance security, and ensure a scalable infrastructure that can adapt to future technological advancements and business needs.

As such, the IT department is now focused on validating the VxLAN network policies. This process is crucial to ensure that the network operates securely and efficiently, and that the data traffic across different segments of the network adheres to the established policies and compliances. The validation process will involve rigorous testing of connectivity, policy enforcement, performance, and security compliance to guarantee that the VxLAN network meets the high standards required by the enterprise.

6.1.2 Description of the use case

6.1.2.1 Overview

The integration strategy for ENI within VxLAN networks is outlined in the present document, focusing on automating policy verification. This process begins with an input phase, where network states are captured and translated into a set of standardized network policies that define network behaviour and configurations.

Subsequent to the initial translation, the methodology establishes a network model that accurately mirrors the actual network architecture. This model aids in the analysis and manipulation of network data, ensuring that all elements are considered for validation and optimization purposes.

6.1.2.2 Motivation

The integration of ENI into VxLAN networks represents a significant advancement in the field of network management and optimization. VxLAN, known for its ability to extend LAN-type networks over a large geographical area, often faces challenges related to complexity in scalability, security, and traffic management. The application of ENI's AI-driven approach promises to revolutionize how these challenges are addressed.

NOTE: The use of AI is out-of-scope for the present document.

Furthermore, the use of ENI's standardized interfaces and protocols within VxLAN networks fosters interoperability and simplifies the integration of diverse network components. This results in a more streamlined, efficient, and secure network infrastructure, capable of adapting to changing demands with minimal manual intervention.

By adopting ENI's experiential network intelligence, VxLAN networks can achieve a new level of automation, efficiency, and flexibility, aligning with the evolving needs of modern network environments. This integration not only enhances the operational capabilities of VxLAN networks but also opens up new possibilities for advanced network services and applications.

6.1.2.3 Actors and Roles

- Inputter: An application or user that will generate an intent policy and input it into ENI System. The intent policy sufficiently describes the requirements of a specific VxLAN.
- ENI System: Translates intent policies into network policies, then detect if there will be any network conflicts.

6.1.2.4 Initial context configuration

All VxLAN networks assisted by ENI are running normally.

6.1.2.5 Triggering condition

The inputter inputs an intent policy and the ENI system successfully translates the intent policy into a network policy.

6.1.2.6 Operational Flow of Actions

1) Snapshot Translation: This step involves translating snapshot information into forwarding, filtering, and rewriting rules. Forwarding rules include destination IP and forwarding interfaces, where the forwarding interface for VxLAN tunnel entry is a VxLAN Tunnel EndPoint (VTEP), which can be either software (e.g. implemented by the hypervisor of a virtual machine) or hardware (e.g. a router or switch).
2) Rule Model Establishment: This includes match domain, hit domain, and action of the rules. It involves converting binary expressions of source and destination IPs into logical expressions and storing them using Binary Decision Diagrams (BDDs). This step ensures that overlapping rules are appropriately managed by removing overlaps in lower-priority rules.
3) Network Model Establishment: Based on global network topology, a device topology graph is constructed. This model breaks down network elements into functional elements and functions, such as forwarding elements and filtering elements, connected via logical and actual interfaces.

4) Model Update: This involves storing the predicate 'True' on the default logical interface of all established functional elements, representing the set of all data packets. The model is updated as new forwarding or filtering rules are inserted, matching rules with the predicates stored on the corresponding functional elements' interfaces.

5) Conflict Detection: This process verifies policies by converting packet information (source and destination IPs) into logical expressions. These expressions are matched with the predicates maintained in the model to extract a forwarding graph. Reachability and invariants like loops are checked using methods like Depth-First Search (DFS) on the forwarding graph.

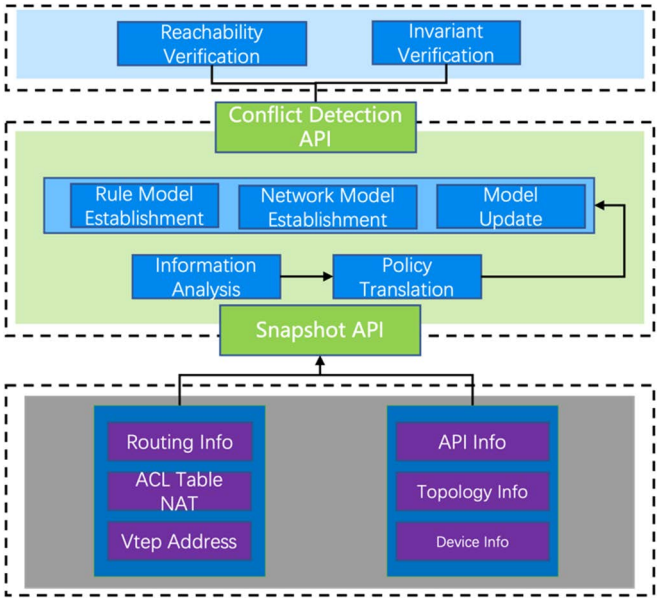

Figure 6-1: Workflow of Network Policy Conflict Detection in VxLAN

6.1.2.7 Post-conditions

If there is no conflict detected:

- The network policy is dispatched to the designated VxLAN network and specific devices.

NOTE: The use of AI is out-of-scope for the present document.

If there are any network policy conflicts:

- ENI system should continue to run further diagnosis to eliminate the conflicts.

7 Areas of Future Study (informative)

7.1 Open Issues for the present document

Void.

7.2 Issues for Future Study

From clause 4.4 (Functional Workflow):

- The workflow introduced in the present document is limited to ENI intent policy ONLY.

From clause 6 (Use Cases):

- Additional use cases may be added to a future version of the present document.

History

Document history		
V4.1.1	July 2024	Publication