Software-defined Network Assimilation: Bridging the Last Mile Towards Centralized Network Configuration Management with NAssim

Huangxun Chen¹, Yukai Miao², Li Chen³, Haifeng Sun⁴, Hong Xu⁵, Libin Liu⁶, Gong Zhang¹, Wei Wang⁷

¹Huawei Theory Lab, ²University of New South Wales, ³Zhongguancun National Lab
⁴Beijing University of Posts and Telecommunications, ⁵The Chinese University of Hong Kong,
⁶Shandong Computer Science Center (National Supercomputer Center in Jinan)
⁷Hong Kong University of Science and Technology (Guangzhou & Hong Kong)
The real network consists of multi-vendor heterogeneous devices

The network in the textbook (Homogeneous Device Model)

The network in reality (Heterogeneous Device Model)

Competition between multiple vendors

Coexistence of legacy & latest devices
Managing a real network is a pain...

Software-defined Network Assimilation (SNA): the process of introducing heterogeneous network devices (e.g., legacy devices and devices from a new vendor) into a centrally controlled, existing SDN network.

- **System View**: Create BGP
  
  CLI command:
  ```
  system-view
  bgp <as-number>
  ```

- **BFD Session -Link Bundle**: Read manuals, create mappings
  
  CLI command:
  ```
  system-view
  bfd <session-name> bind peer-ip <peer-ip>
  detect-multiplier <multiplier>
  ```

- **Service**: VPRN
  
  CLI command:
  ```
  service
  vprn <vprn-id> customer <cid> create automomous-system <as-number>
  ```

- **BFD Multiplier**: BFD RX Family
  
  CLI command:
  ```
  system-view
  lag <lag-id> bfd
  family { ipv4 | ipv6 }
  multiplier [ <multiplier> ]
  ```
Current SNA approaches require significant human effort

Vendor

Vendor-specific Device Model (VDM)

Manual

NetOps Expert

SDN Controller

Unified Device Model (UDM)

Proposed system: NAssim

VDM Construction Phase

VDM-UDM Mapping Phase

Is it feasible to liberate the NetOps from most tedious tasks by learning directly from devices' manuals?
Challenges for SDN Network Assimilation (SNA)

- Manual Format Heterogeneity
- Errors & Ambiguity in Manuals
- Heterogeneity between Configuration Models
Challenge 1: Manual Format Heterogeneity

peer as-number (BGP view)

Function
The peer as-number command creates a peer and configures an AS number for a specified peer. This peers as-number command defines the AS number of a specified peer. By default, no BGP peer is configured, and no AS number is specified for a peer.

Format
peer ipv6-adress as-number
undo peer ipv6-adress

Examples
# Set the AS number to 100 for IPv6 peer 10.1.1.1.
set peer 10.1.1.1 as-number 100

views
BGP view
Default Level
1. Configuration level

label-ipv4
Syntax
label ipv4 [word | class | none] label ipv4 [word | class | none]

Parameters
prefix — The maximum number of per-label/unique IPv4 prefixes that are allowed to be advertised to add-paths peers. (The actual number of advertised routes may be less, if the value is zero, the router does not inherit capability with respect to label IPv4 AFs.)
Value — 1 to 4094

Examples
# The router negotiates to receive multiple labeled-unicast routes per IPv4 peer.
none — The router does not negotiate to receive multiple labeled-unicast routes per IPv4 peer.

redistribute (BGP)

To inject routes from one routing domain into the Border Gateway Protocol (BGP), use the redistribute command. To remove the redistribute command from the configuration file and restore the system to its default condition in which the software does not redistribute routes, use the no form of this command.

redistribute [direct | eigrp instance-tag | ospf instance-tag | rip instance-tag | static | route-map map-name] no redistribute [direct | eigrp instance-tag | ospf instance-tag | rip instance-tag | static | route-map map-name]

Syntax Description
- direct - Direct routes that are directly connected on an interface.
- eigrp instance-tag - Specifies the name of an EIGRP instance. The instance-tag can be any case-sensitive, alphanumeric string up to 20 characters.
- ospf instance-tag - Specifies a destination AS number.
- rip instance-tag - Specifies routes from the RIP protocol. The instance-tag can be any case-sensitive, alphanumeric string up to 20 characters.
- static - Redistributes RIP IP static routes.
- route-map map-name — Optional. Specifies the identifier of a configured route map. Use a route map to filter which routes are redistributed into BGP.

Command Default
Disabled

Command Modes
Command History/Address family configuration mode
Router configuration mode
Router VRF configuration mode

Examples
# Specify the OSPF cost on the Interface as 33.
 OSPF cost 33

H3C
Challenge 2: Errors & Ambiguity in Manuals

An ambiguous CLI command template in a Cisco manual*:

```
neighbor { <ip-addr> | <ip-prefix/length> } [ remote-as { <as-num> [ <.as-num> ] | route-map <name> } }
```

Correction Option 1: removing the left bracket

```
neighbor { <ip-addr> | <ip-prefix/length> } remote-as { <as-num> [ <.as-num> ] | route-map <name> } }
```

Correction Option 2: adding a right bracket after remote-as symbol

```
neighbor { <ip-addr> | <ip-prefix/length> } [ remote-as ] { <as-num> [ <.as-num> ] | route-map <name> } }
```

Correction Option 3: adding a right bracket at the end of the CLI

```
neighbor { <ip-addr> | <ip-prefix/length> } [ remote-as { <as-num> [ <.as-num> ] | route-map <name> } ]
```

We need a rigorous validation scheme to catch inevitable mistakes and typos in human-written manuals.

The sheer number of CLI commands and parameters makes handcrafting mapping tedious and error-prone.
Our Contributions: NAssim

• With investigating commonalities and diversity of popular vendors’ manuals, we design a vendor-independent device model corpus format and enable the Test-Driven Development of parsers to enhance the reliability of device manual parsing.

• We design a rigorous and human-comprehensible validation framework for the parsed results. We perform validation on three levels: command-level, inter-command-level, and snippet-level. NAssim’s Validator also identifies 184 syntactic errors and 59 ambiguities in four mainstream vendors’ manuals.

• We employ state-of-the-art contextual learning NLP model, BERT, and augment it to create a domain-adapted version for our usecase, NetBERT. NetBERT achieves 89% and 70% top 10 recall for mapping device models of Huawei and Nokia to a given UDM respectively. The output of NetBERT is a mapping between parameters of different models, which is comprehensible and editable by NetOps experts.

• We release a validated and expert-curated dataset of parsed manual corpus for future research. ([https://github.com/AmyWorkspace/nassim](https://github.com/AmyWorkspace/nassim))
SDN Network Assimilation (NAssim) in a Nutshell

Vendor

Manual Format Heterogeneity

Errors & Ambiguity in Manuals

Configuration Files

Parser

Validator

Mapper

Heterogeneity between Configuration Models

Unified Device Model

An Assistant Framework for Bridging the Last Mile Towards Centralized Network Configuration Management
Solutions for SNA Challenges

NAssim Parser

Manual Format Heterogeneity

Errors & Ambiguity in Manuals

Heterogeneity between Configuration Models

SDN Controller
Despite diverse styles, all manuals serve the same purpose: show how to configure the devices via CLI.

Table 1: Diversity of Device User Manuals: The 'CLIs' field denotes the formal syntax of CLI commands, which are command templates with place-holder parameters and special characters to specify selection or optional branches. The 'ParaDef' field contains the implication and value range of place-holder parameters. The 'FuncDef' field describes the functionality of the complete CLI. The 'ParentViews' field indicates the parent/working views of CLIs, i.e., one CLI may have multiple viable working views. The 'Examples' field shows examples of common snippets, e.g., entering a parent view and issuing an instantiated CLI.
Solution: NA<sup>n</sup>sim Parser Framework

### Vendor-independent parsed format captures the commonality of manuals from different vendors, also balancing extendibility and human-readability.

### Preliminary VDM

```json
{
    "CLI": [
        "peer <ipv4-address> group <group-name>",
        "undo peer <ipv4-address> group <group-name>",
    ],
    "FuncDef": "The peer group command adds a peer to a peer. The undo peer group command deletes a peer from a peer group and all configurations of the peer. By default, no peer group is created",
    "ParentView": "[BGP view]",
    "ParaDef": [
        "Parameters": "ipv4-address",
        "Info": "Specifies the IPv4 address of a peer. It is in dotted decimal notation.",
    ],
    "Examples": [
        "<HUAWEI> system-view",
        "<HUAWEI> bgp 100",
        "<HUAWEI-bgp> group test internal",
        "<HUAWEI-bgp> peer 10.1.1.1 group test"
    ],
}
```

A sample of parsed VDM corpus.
Test-driven human-in-the-loop methodology improves the quality of the parsed corpora in an interpretable and reliable way.
Solutions for SNA Challenges

NAssim Parser

NAssim Validator

Manual Format Heterogeneity

Errors & Ambiguity in Manuals

Heterogeneity between Configuration Models
Solution: NAssim Validator

1. **Formal Syntax Validation**
   - Preliminary VDM
   - JSON

2. **Hierarchy Derivation & Validation**
   - Preliminary VDM (CLI syntax validated)
   - JSON
   - semantics
   - CLI
   - JSON
   - Refined VDM

3. **Device Configuration Validation**
   - Refined VDM (Empirically-validated)

Device Configurations
Solution: NAssim Validator – Formal Syntax Validation

1. Formal Syntax Validation

Command descriptions use these conventions:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface font</strong></td>
<td>Commands and keywords are in boldface.</td>
</tr>
<tr>
<td><em>italic font</em></td>
<td>Arguments for which you supply values are in italics.</td>
</tr>
<tr>
<td>[]</td>
<td>Elements in square brackets are optional.</td>
</tr>
<tr>
<td>{x</td>
<td>y</td>
</tr>
<tr>
<td>[x</td>
<td>y</td>
</tr>
<tr>
<td><strong>string</strong></td>
<td>A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.</td>
</tr>
</tbody>
</table>

Command convention of Cisco manuals

```python
import pyparsing as p

# syntax parser for Cisco CLI
word = p.Word(p.printables, exclude_chars='{}|\n').setParseAction(leaf_gen)

ele = p.Forward()

items = ele + p.ZeroOrMore(' | ' + ele)

option = p.Group('[ ' + items + ' ]').setParseAction(select_gen)

ele << p.ZeroOrMore(option ^ select ^ word).setParseAction(ele_gen)

syntax_parser = ele
```

Express command conventions-syntax into their equivalent Backus Normal Form (BNF) for formal validation.
Leverage example snippets to derive the relationship between individual CLI commands, i.e., CLI model hierarchy.
Solution: NAssim Validator – Hierarchy Derivation & Validation

CLI Graph Model (CGM)

CGM Constuction Algorithm

CLI Template

filter-policy { <acl-number>
  | ip-prefix <ip-prefix-name>
  | acl-name <acl-name> } { import | export }

CLI Instance

filter-policy acl-name acl1 export

CLI Instance-Template Matching Algorithm

CLI Template and CLI Instance association through CLI graph model.
Quantify the certainty of the derivation to facilitate expert intervention.
**Solution: NAssim Validator – Device Configuration Validation**

Leverage empirical device configurations to further validate the extracted VDM.
Solutions for SNA Challenges

- NAssim Parser
  - Manual Format Heterogeneity
- NAssim Validator
  - Errors & Ambiguity in Manuals
- NAssim Mapper
  - Heterogeneity between Configuration Models
Solution: NAssim Mapper

VDM

semantics

JSON

CLI

Context

Context Embedding

Mapper

UDM

1. Context Extraction

2. Context Encoding

3. Similarity Matcher

Context

Context Embedding

SCORE
For VDM (or UDM) \( M \), with \( n_M \) parameters and one of its parameter \( p_i^M (i \in 0, 1, \ldots, n_M - 1) \), its context is

\[
C(p_i^M) = [s_i^0, s_i^1, \ldots, s_i^{k_M-1}]
\]

where each \( s_i \) is a text sequence and \( k_M \) is the number of extracted sequences.

```json
{
  "CLIs": [
    "peer <ipv4-address> group <group-name>", [CMD]
    "undo peer <ipv4-address> group <group-name>"],
  "FuncDef": "The peer group command adds a peer to a peer. The undo peer group command deletes a peer from a peer group and all configurations of the peer. By default, no peer group is created",
  "ParentView": ["BGP view"],
  "ParaDef": [para
desp][CMD desp][parent view][para desp]
  ["Parameters": "ipv4-address",
  "Info": "Specifies the IPv4 address of a peer. It is in dotted decimal notation.",
  ["Parameters": "group-name",
  "Info": "Specifies the name of a peer group. The name is a string of 1 to 47 case-sensitive characters, with spaces not supported."}
]
  ["Examples": [para
desp][CMD desp][parent view][para desp]
  ["<HUAWEI> system-view",
    ["~HUAWEI] bgp 100",
    ["*HUAWEI-bgp] group test internal",
    ["*HUAWEI-bgp] peer 10.1.1.1 group test"]
  ]
}
```
Encode each of the text separately and then produce an embedding matrix.

Assume the output dimension of the encoder $e(\cdot)$ is $m$, the context embedding of $p_i^M$ is:

$$E_i^M = e(C(p_i^M)) = e([s_i^0, s_i^1, ..., s_i^{k_M-1}]) \in R^{k_M \times m}$$
Solution: NAssim Mapper – Context Extraction

SBERT*

-1 ... 1

Cosine-Sim(u, v)

u

v

pooling

pooling

BERT

BERT

Sentence A

Sentence B

Fine Tuning Scheme

NetBERT

A few UDM-VDM mapping data annotated by experts

For \( E^V_i \) and \( E^U_j \) with the number of contexts \( k_V \) and \( k_U \), the similarity score for context embedding is:

\[
\text{Sim}(E^V_i, E^U_j) = w \cdot (E^V_i \otimes E^U_j)
\]

where \( w = \langle w_0, w_1, ..., w_{k_V \times k_U - 1} \rangle \) \((s.t. \sum_{t=0}^{k_V \times k_U - 1} w_t = 1)\)

\(\otimes\) : compute row-wise cosine similarities of two matrices

I. The effectiveness and reliability of VDM construction phase

II. The performance of VDM-UDM mapping phase
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Statistics</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#CLI Commands</td>
<td>12874</td>
<td>278</td>
<td>14046</td>
<td>759</td>
</tr>
<tr>
<td>#Views</td>
<td>607</td>
<td>27</td>
<td>3832</td>
<td>28</td>
</tr>
<tr>
<td>#CLI-View Pairs</td>
<td>36274</td>
<td>366</td>
<td>22734</td>
<td>851</td>
</tr>
<tr>
<td>Adaptation Cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parsing() LOC</td>
<td>45</td>
<td>52</td>
<td>57</td>
<td>41</td>
</tr>
<tr>
<td>get_cli_parser() LOC</td>
<td>8</td>
<td>6</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Syntax Validation</td>
<td>#Invalid CLI Commands</td>
<td>13</td>
<td>19</td>
<td>137</td>
</tr>
<tr>
<td>Model Hierarchy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Derivation &amp; Validation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Examples Snippets</td>
<td>15466</td>
<td>523</td>
<td>/</td>
<td>1147</td>
</tr>
<tr>
<td>Construction Time (second)</td>
<td>785.58</td>
<td>14.29</td>
<td>94.56*</td>
<td>34.3</td>
</tr>
<tr>
<td>#Ambiguous Views</td>
<td>47</td>
<td>8</td>
<td>/</td>
<td>4</td>
</tr>
<tr>
<td>Device Configuration</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validation</td>
<td>#Config Files</td>
<td>/</td>
<td>416</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>Matching Ratio</td>
<td>/</td>
<td>100%</td>
<td>/</td>
</tr>
</tbody>
</table>

Table 3: Evaluation of the VDM Construction Phase. *Nokia manuals do not provide examples, but they explicitly specify model hierarchy in the manuals. Thus, we extract the hierarchy using `Parser_<nokia>` by implementing extra functions.
# Evaluation: VDM-UDM Mapping Phase

<table>
<thead>
<tr>
<th>Mapping Setting</th>
<th>Models</th>
<th>k in recall@top k (%)</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Huawei-UDM</td>
<td>IR</td>
<td>41</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>SimCSE</td>
<td>40</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>SBERT</td>
<td>53</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>IR+SimCSE</td>
<td>43</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>IR+SBERT</td>
<td>56</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>NetBERT</td>
<td>57</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>IR+NetBERT</td>
<td>58</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5401</td>
<td></td>
</tr>
<tr>
<td>Nokia-UDM</td>
<td>IR</td>
<td>24</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>SimCSE</td>
<td>20</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>SBERT</td>
<td>34</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>IR+SimCSE</td>
<td>24</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>IR+SBERT</td>
<td>34</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>NetBERT</td>
<td>34</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>IR+NetBERT</td>
<td>35</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.3498</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.2679</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.3908</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.3241</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.417</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.4322</td>
<td></td>
</tr>
</tbody>
</table>

Recall@top-k denotes the percentage of test cases where the correct matching parameters are in top k recommendation by Mapper.
Summary

• The current soft-defined network assimilation process for managing multi-vendor network requires significant human efforts and error-prone.

• Our solution NAssim features a unified parser framework, a rigorous validator and a mapper using the domain-adapted BERT model to produce human-comprehensible recommended mapping between the validated configuration model and the one in the SDN controller.

• NAssim act as an assistant framework to provides tools and recommendations to NetOps engineers to accelerate the current SNA process, bridging the last mile towards centralized network configuration management.

• We release a validated and expert-curated dataset of parsed manual corpus for future research. (https://github.com/AmyWorkspace/nassim)
Thank you!
Q&A