SELFSTARTER

Finding Network Misconfigurations by Automatic Template Inference

Siva Kesava Reddy Kakarla[1]

Alan Tang[1] Ryan Beckett[2] Karthick Jayaraman[3] Todd Millstein[1,4] Yuval Tamir[1] George Varghese[1]

[1] University of California, Los Angeles, USA

[2] Microsoft Research [3] Microsoft Azure

[4] Intentionet

Network Misconfigurations Are Common

TRAVEL • AMERICAN AIRLINES

American Airlines Network Outage Delays Flights Nationwide

By David Z. Morris July 29, 2018

Internet Goes Down for Parts of the US Due to a Misconfiguration

By CircleID Reporter

Nov 06, 2017 8:58 PM PDT | Comments: 0 | Views: 5,914

Comment

Another massive outage takes down many of the internet's biggest sites and service

BGR Jacob Siegal, BGR News • July 2, 2019

THE ACCIDENTAL LEAK —

Google goes down after major BGP mishap routes traffic through China

How four rotten packets broke CenturyLink's network for 37 hours, knackering 911 calls, VoIP, broadband

FCC delivers postmortem after blunder cripples US fiber links

By Shaun Nichols in San Francisco 20 Aug 2019 at 20:12 53 SHARE ▼

Microsoft Azure recovering from major networking-related outage that took out Office 365, Xbox Live, and other services

BY TOM KRAZIT on May 2, 2019 at 2:48 pm

Google Cloud outage takes down Snapchat, YouTube, and Gmail in parts of the United States

Megan Hernbroth Jun. 2, 2019, 4:29 PM

"This outage was caused by an equipment failure catastrophically exacerbated by a network configuration error," America's communications regulator said in its summary of its inquiry, published yesterday.

Network Misconfigurations Are Common

TRAVEL • AMERICAN AIRLINES

American Airlines Network Outage Delays Flights Nationwide

By David Z. Morris July 29, 2018

Internet Goes Down for Par… Due to a Misconfiguration

By CircleID Reporter

Nov 06, 2017 8:58 PM PDT | Comments: 0 | Views: 5,914

How four rotten packets broke CenturyLink's network for 37 hours, knackering 911 calls, VoIP, broadband

FCC delivers postmortem after blunder cripples US fiber links

By Shaun Nichols in San Francisco 20 Aug 2019 at 20:12 53 SHARE ▼

…overing from major networking-…ook out Office 365, Xbox Live,

How can we detect misconfigurations?

Comment |

Another massive outage takes down many of the internet's biggest sites and service

BGR Jacob Siegal, BGR News • July 2, 2019

THE ACCIDENTAL LEAK —

Google goes down after major BGP mishap routes traffic through China

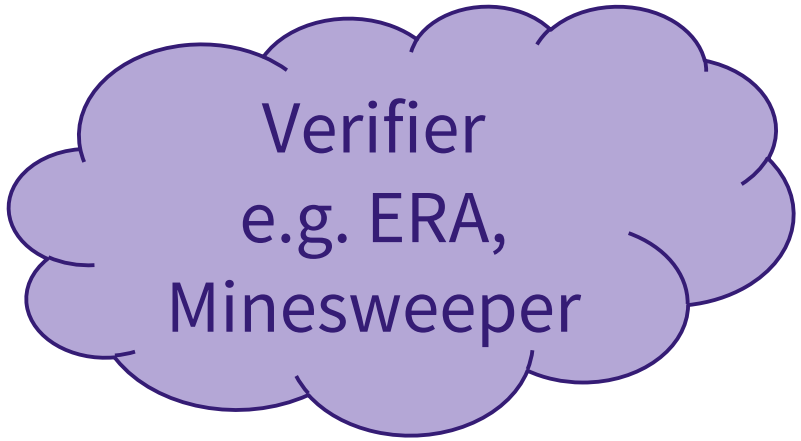
Google Cloud outage takes down Snapchat, YouTube, and Gmail in parts of the United States

Megan Hernbroth Jun. 2, 2019, 4:29 PM

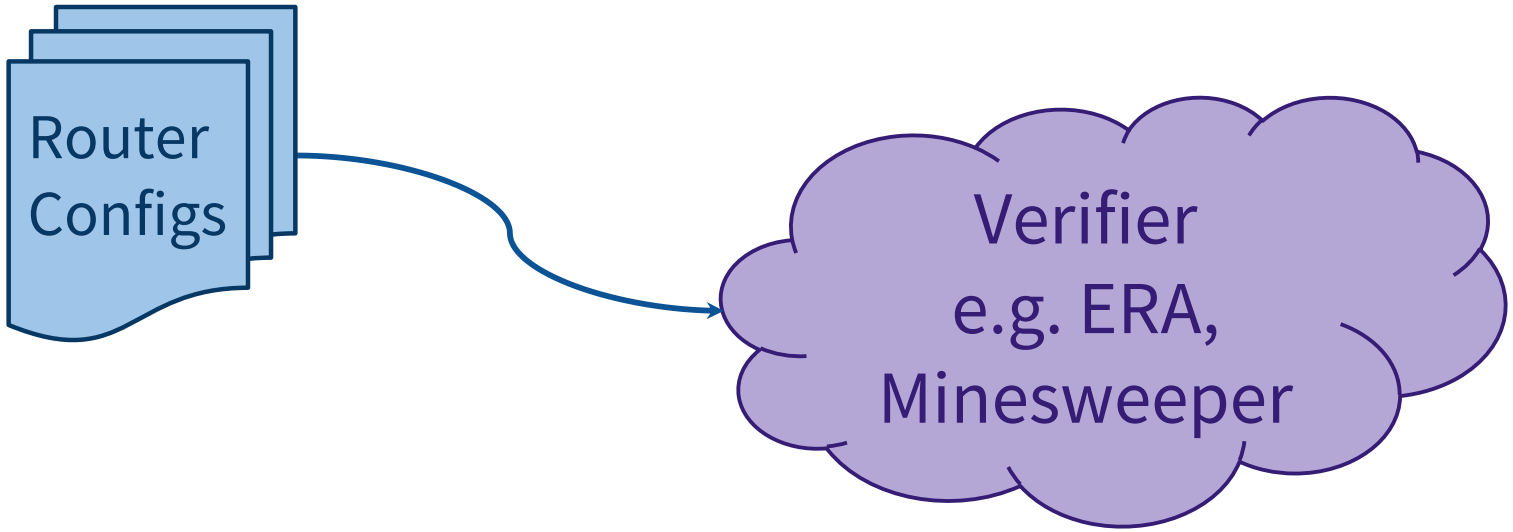
"This outage was caused by an equipment failure catastrophically exacerbated by a network configuration error," America's communications regulator said in its summary of its inquiry, published yesterday.

Network Verification

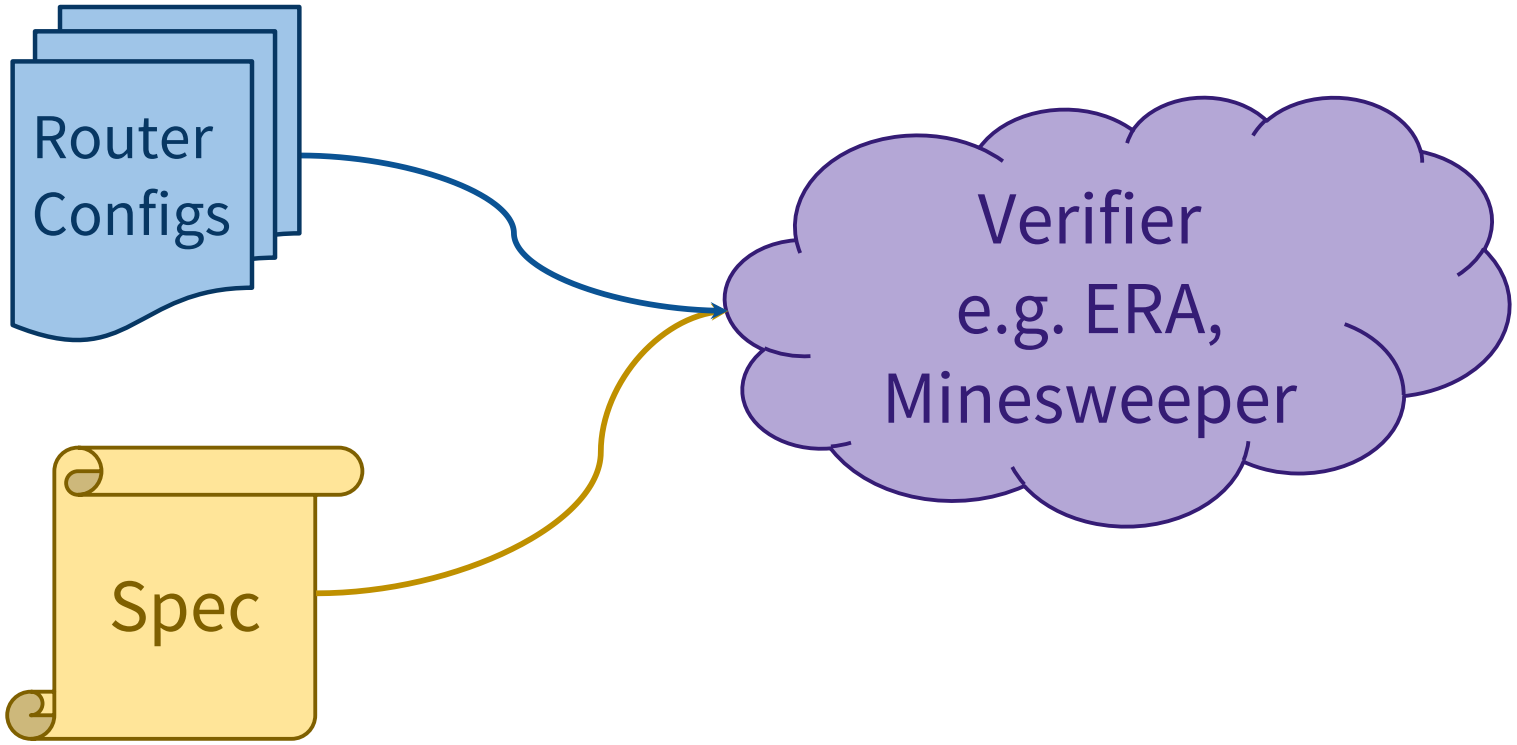
Network Verification



Network Verification

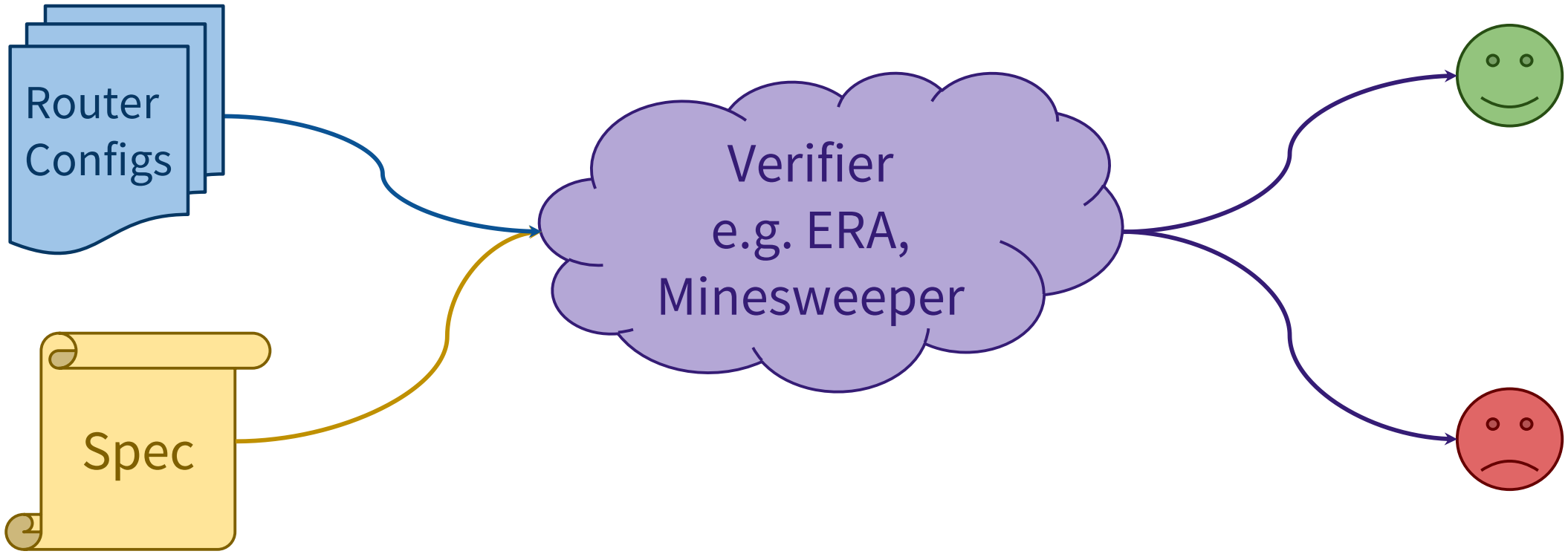


Network Verification

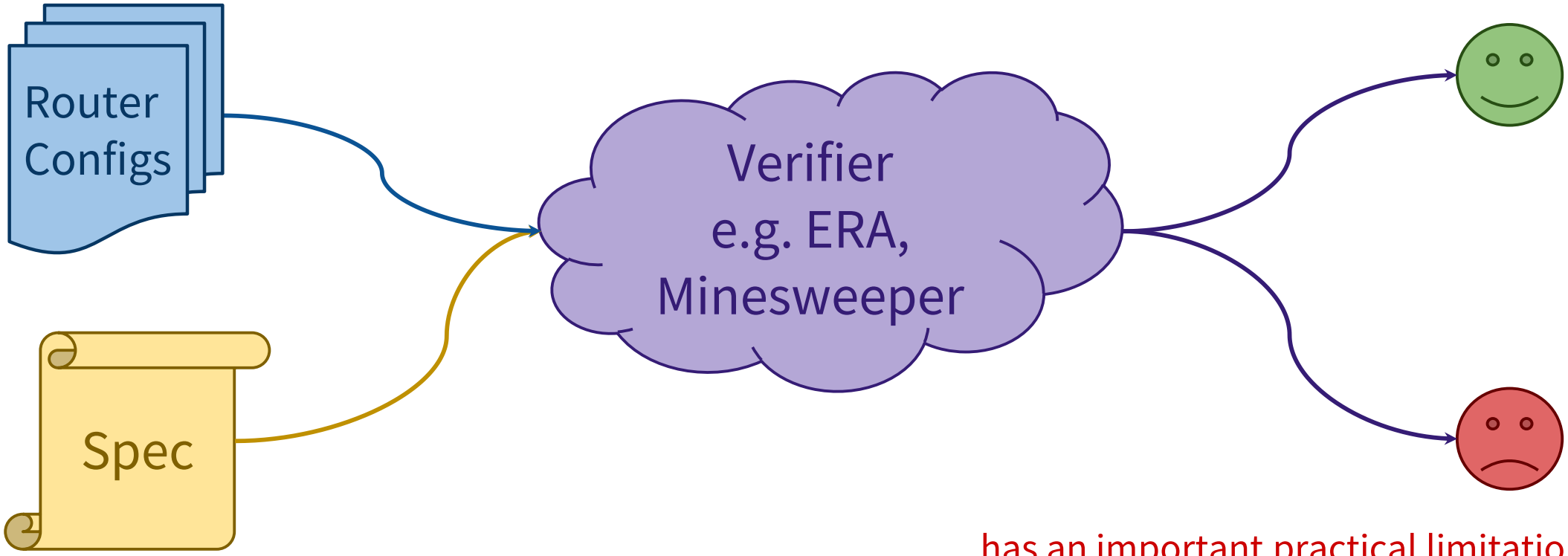


Network Verification

Network Verification

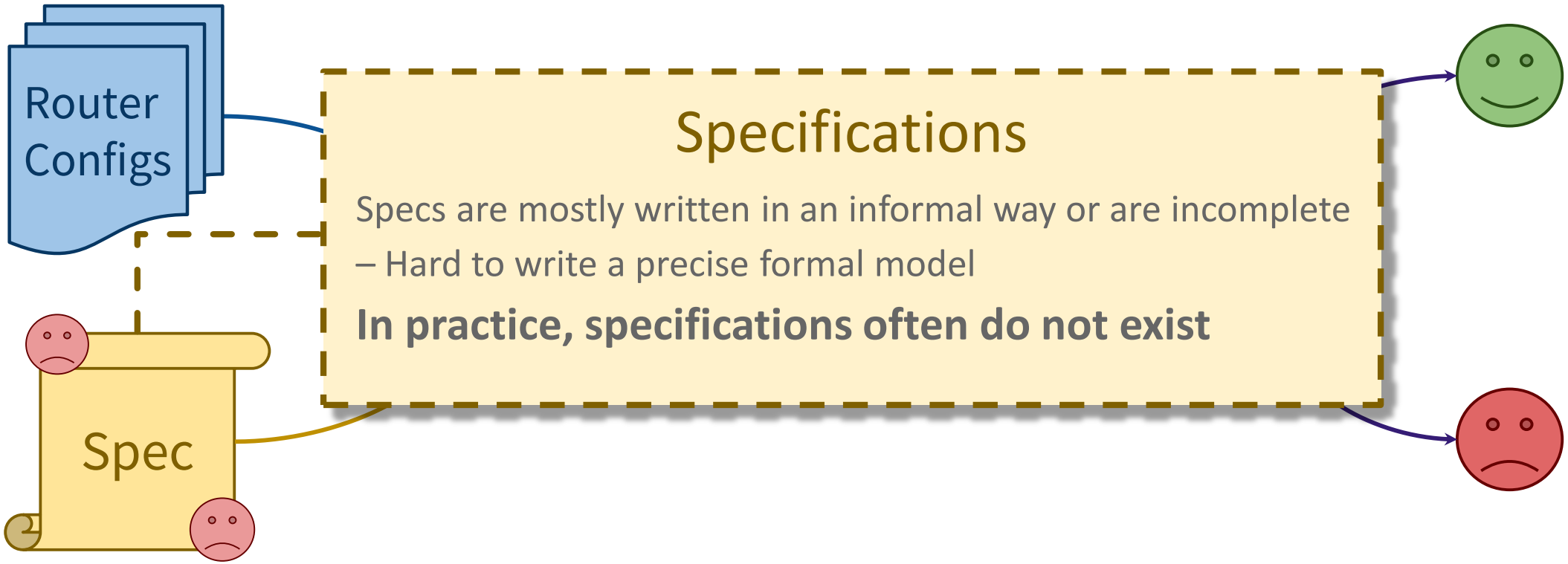


Network Verification



… has an important practical limitation.

Limitation: Lack of Specifications



PROBLEM:
How to find misconfigurations
without an explicit specification?

PROBLEM:

How to find misconfigurations without an explicit specification?

GENERAL APPROACH:

Bugs as deviant behavior!

"If thousands of people all do the same action, we know the majority is probably right, and any contradictory action is probably wrong without knowing the correct behavior."†

†Bugs as Deviant Behavior: A General Approach to Inferring Errors in Systems Code - SOSP 2001

PROBLEM:

How to find misconfigurations without an explicit specification?

GENERAL APPROACH:

Bugs as deviant behavior!

"If thousands of people all do the same action, we know the majority is probably right, and any contradictory action is probably wrong without knowing the correct behavior."†

OUR INSIGHT:

Exploit network device roles

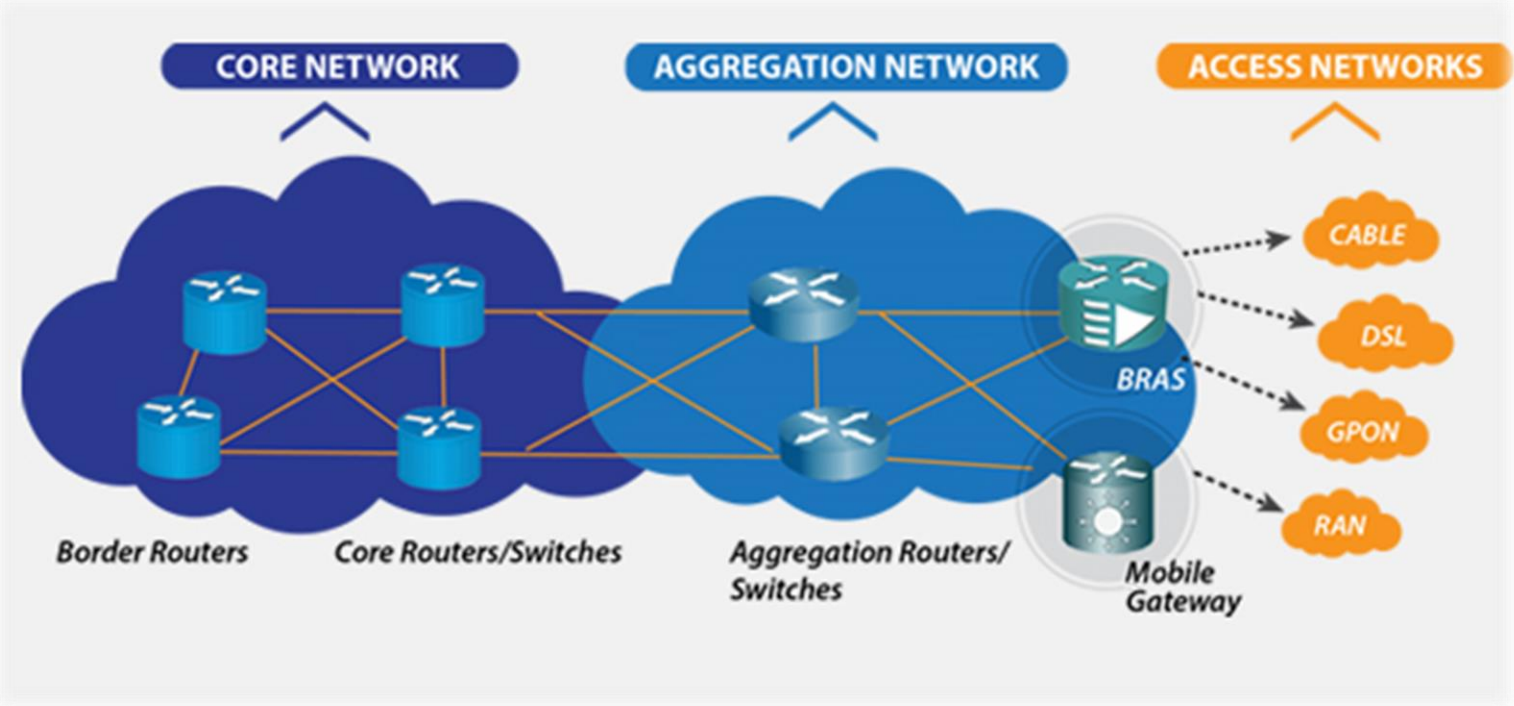
†Bugs as Deviant Behavior: A General Approach to Inferring Errors in Systems Code - SOSP 2001

Network Device Roles

- Routers in a network are generally assigned certain “roles” based on their use

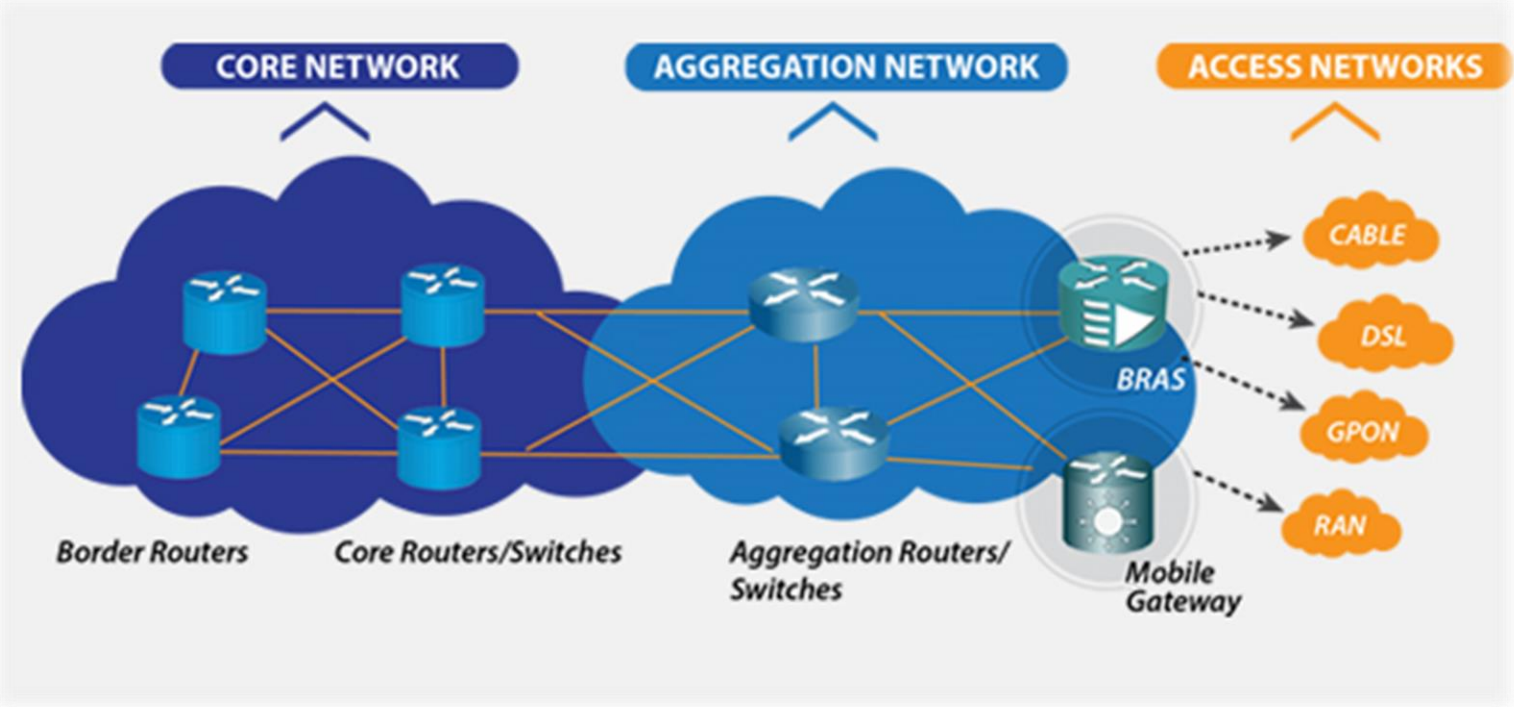
Network Device Roles

- Routers in a network are generally assigned certain "roles" based on their use



Network Device Roles

- Routers in a network are generally assigned certain "roles" based on their use
- Routers in the same role of a network are typically configured "similarly"



PROBLEM:

How to model config “similarity” and define “deviance” ?

PROBLEM:

How to model config “similarity” and define “deviance” ?

NAÏVE APPROACH:

Exact configuration equivalence

Too strong an assumption! Router configurations in a role have many intentional differences (e.g., local IPs)

PROBLEM:

How to model config "similarity" and define "deviance" ?

NAÏVE APPROACH:

Exact configuration equivalence

Too strong an assumption! Router configurations in a role have many intentional differences (e.g., local IPs)

OUR APPROACH:

Infer parameterized *templates* to distinguish intentional differences from likely bugs

Contributions

- First automatic template inference algorithm for *any* configuration segment, e.g. ACLs, policy maps and so on

Contributions

- First automatic template inference algorithm for *any* configuration segment, e.g. ACLs, policy maps and so on
- Concrete instantiations of our generic algorithm for three widely used segments – ACLs, prefix lists and route policies

Contributions

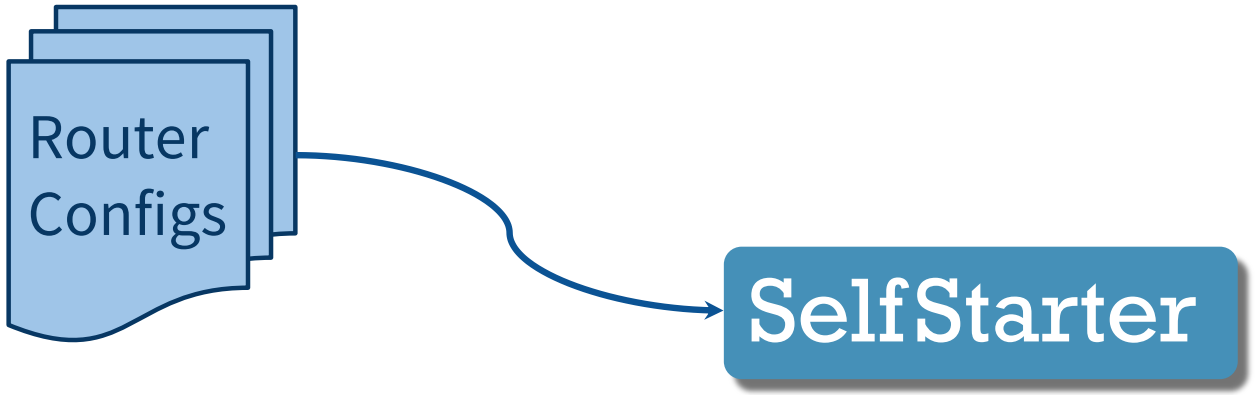
- First automatic template inference algorithm for *any* configuration segment, e.g. ACLs, policy maps and so on
- Concrete instantiations of our generic algorithm for three widely used segments – ACLs, prefix lists and route policies
- SelfStarter – Tool for finding potential bugs in router configurations with actionable feedback for operators using automatic template inference

Contributions

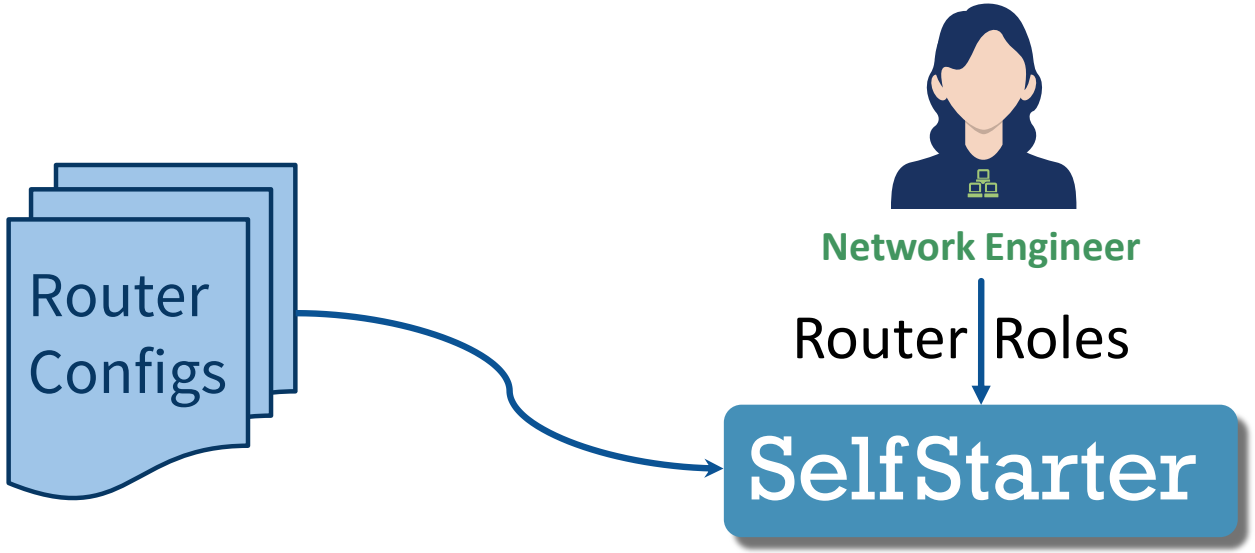
- First automatic template inference algorithm for *any* configuration segment, e.g. ACLs, policy maps and so on
- Concrete instantiations of our generic algorithm for three widely used segments – ACLs, prefix lists and route policies
- SelfStarter – Tool for finding potential bugs in router configurations with actionable feedback for operators using automatic template inference
- SelfStarter found 43 previously unknown bugs in total in Microsoft networks and a large campus network.

End-to-End Design

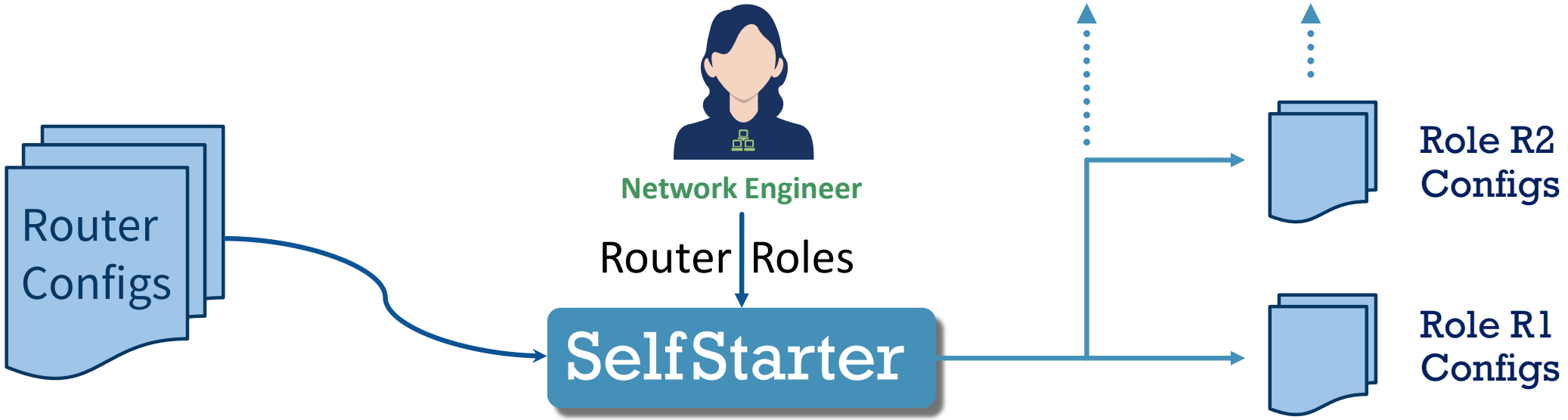
End-to-End Design



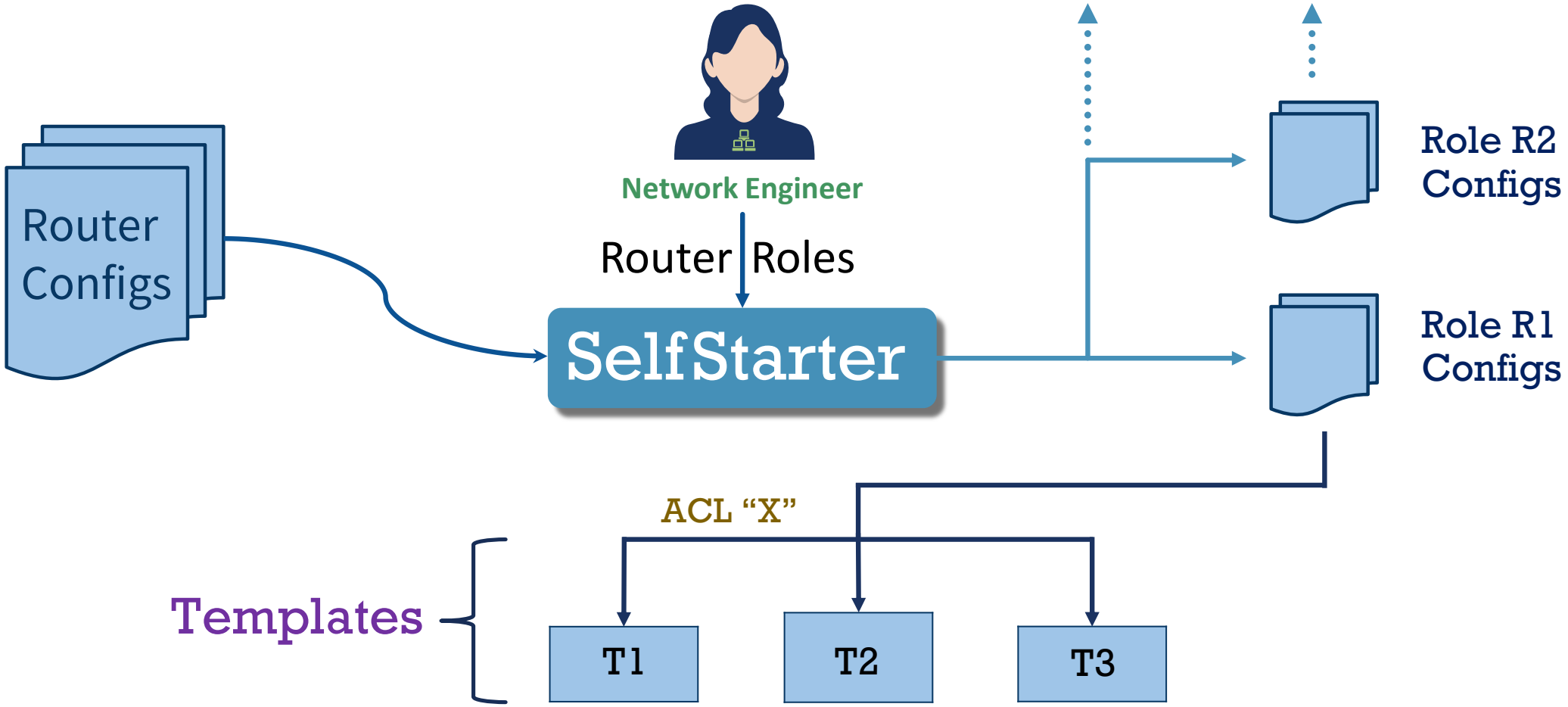
End-to-End Design



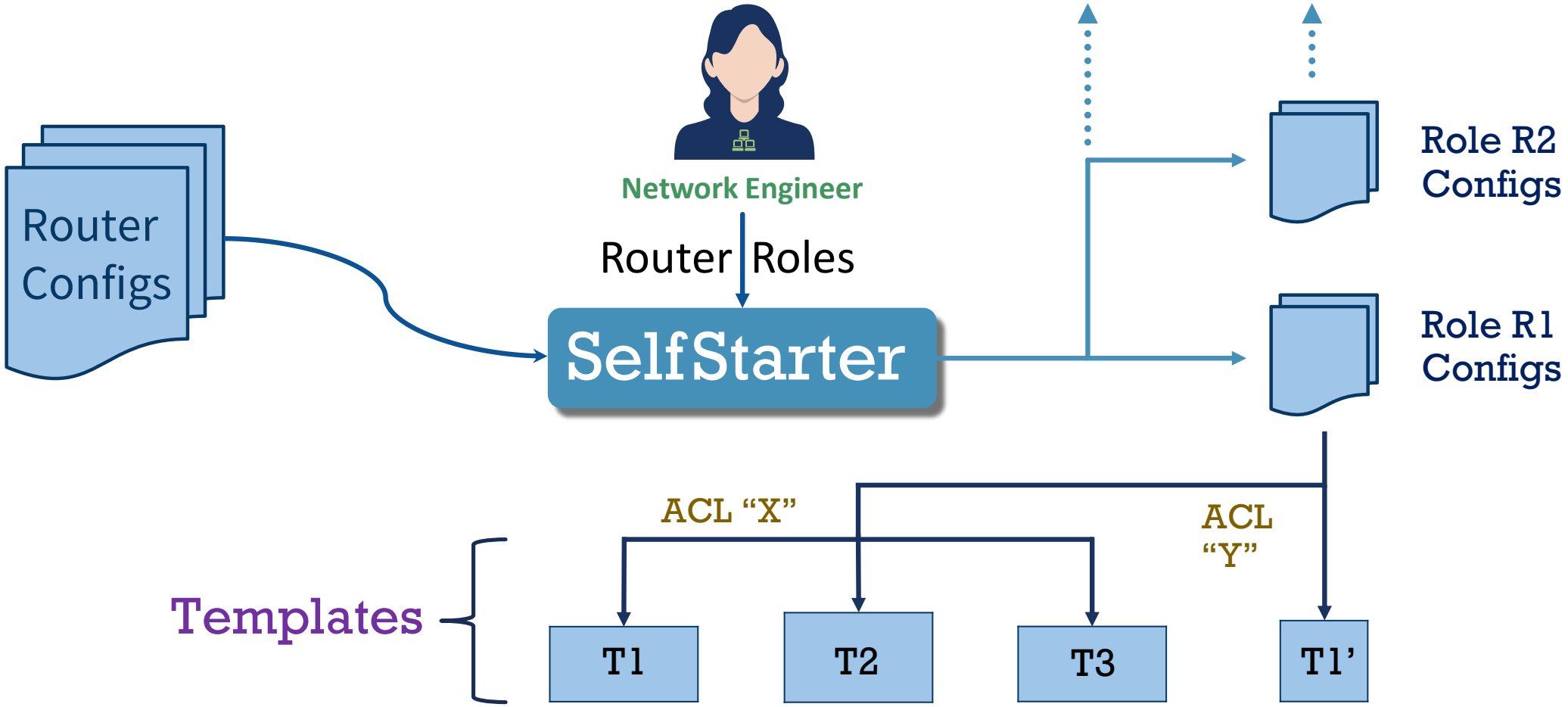
End-to-End Design



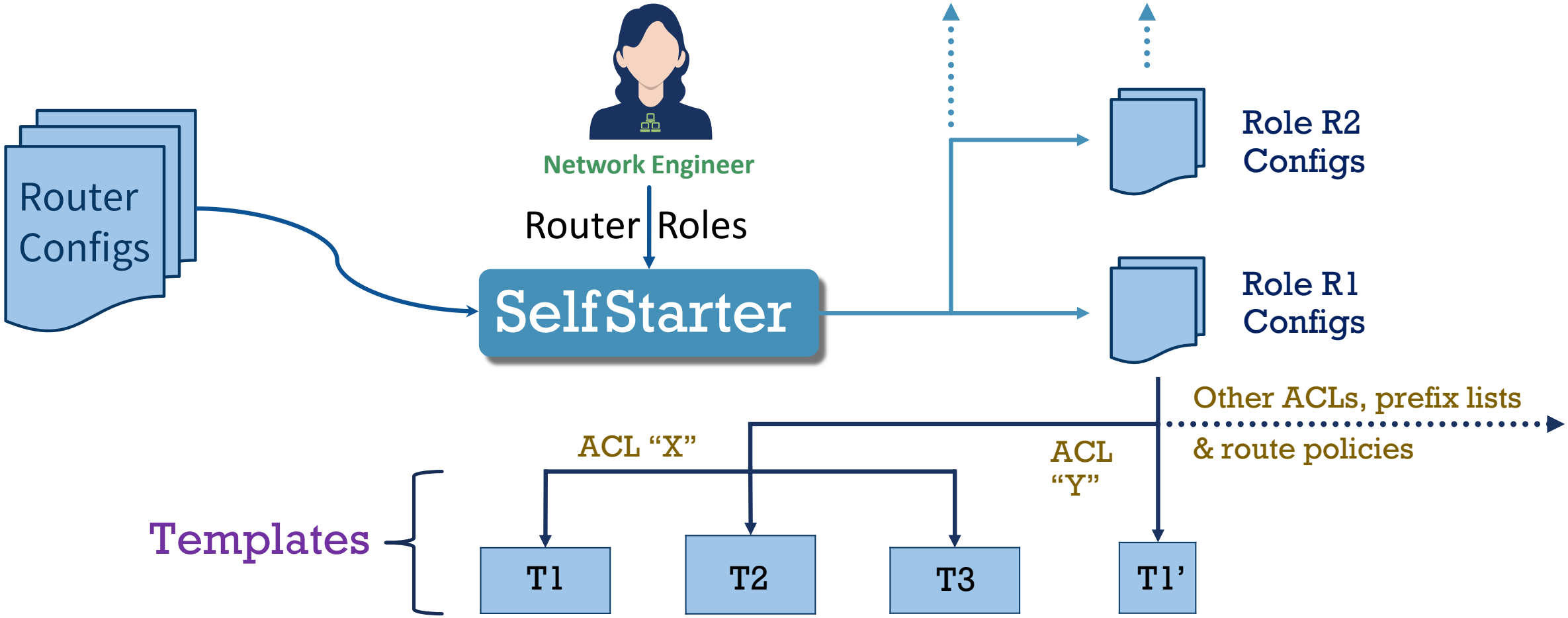
End-to-End Design



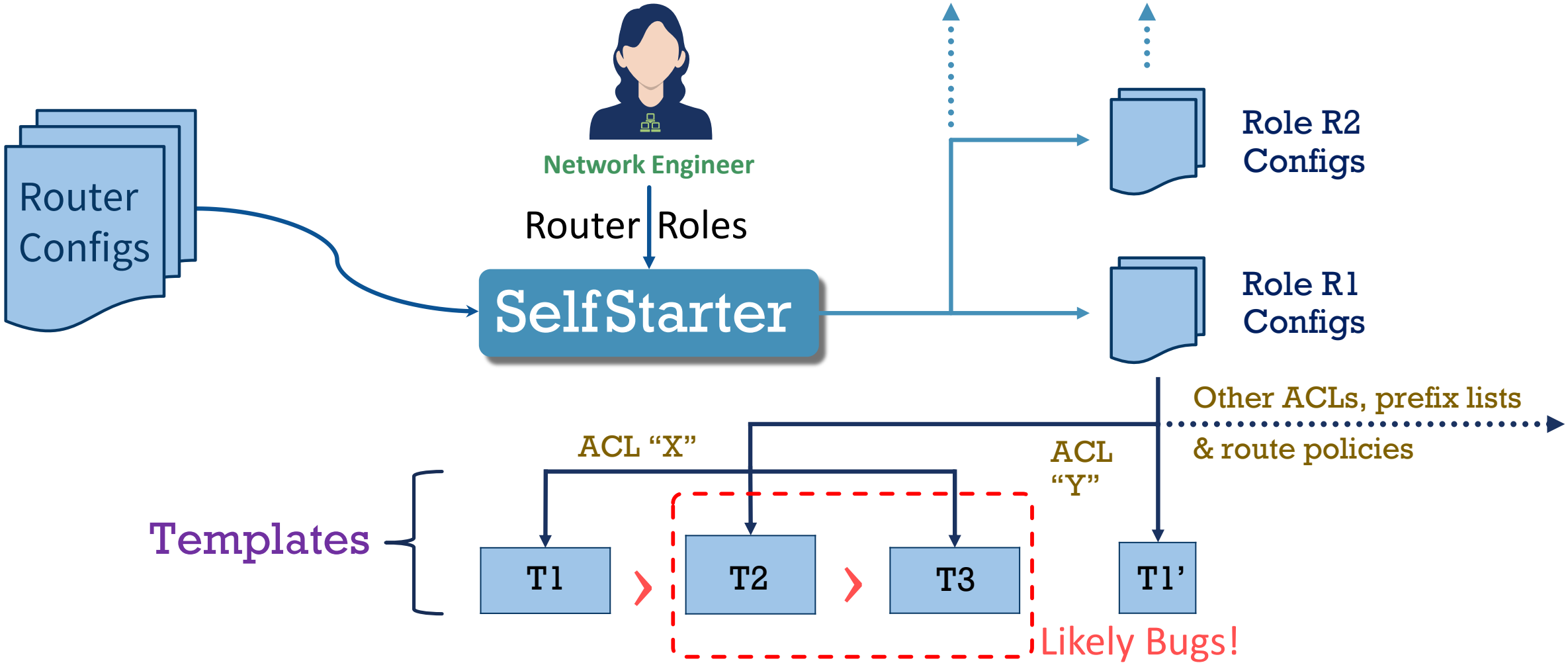
End-to-End Design



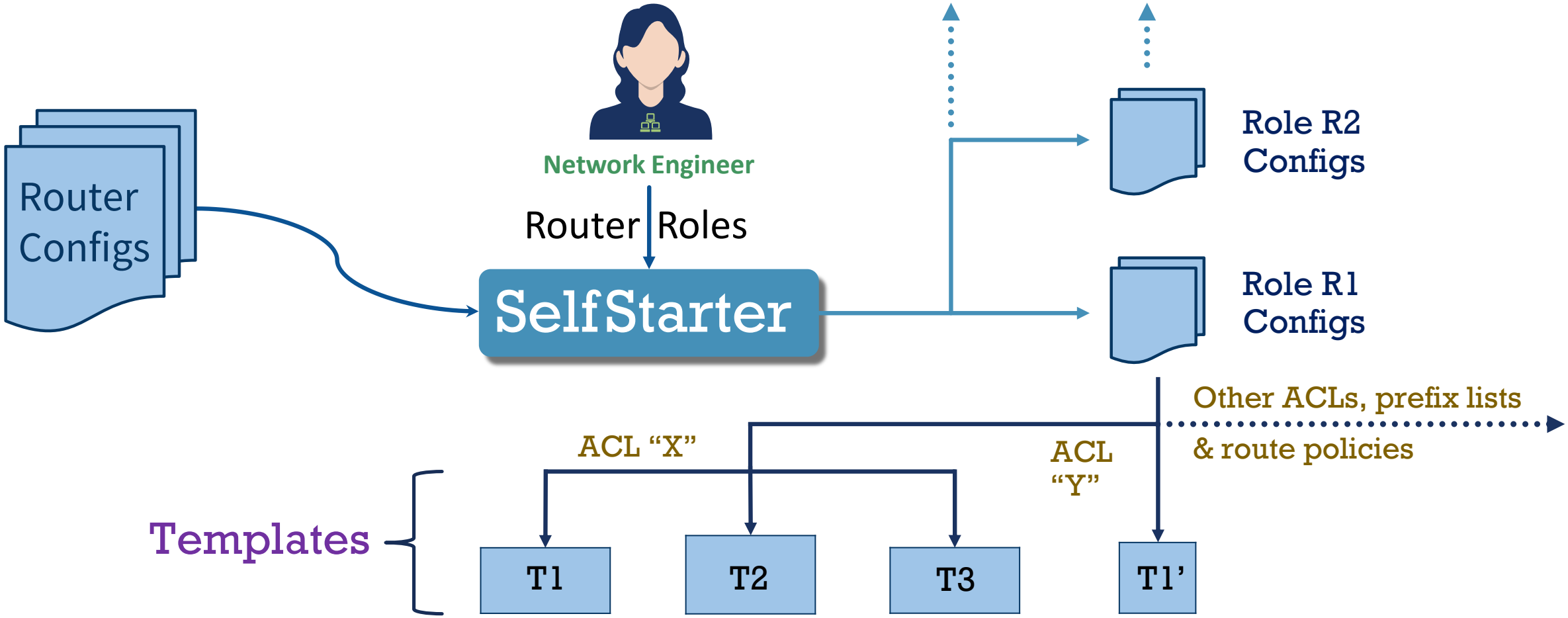
End-to-End Design



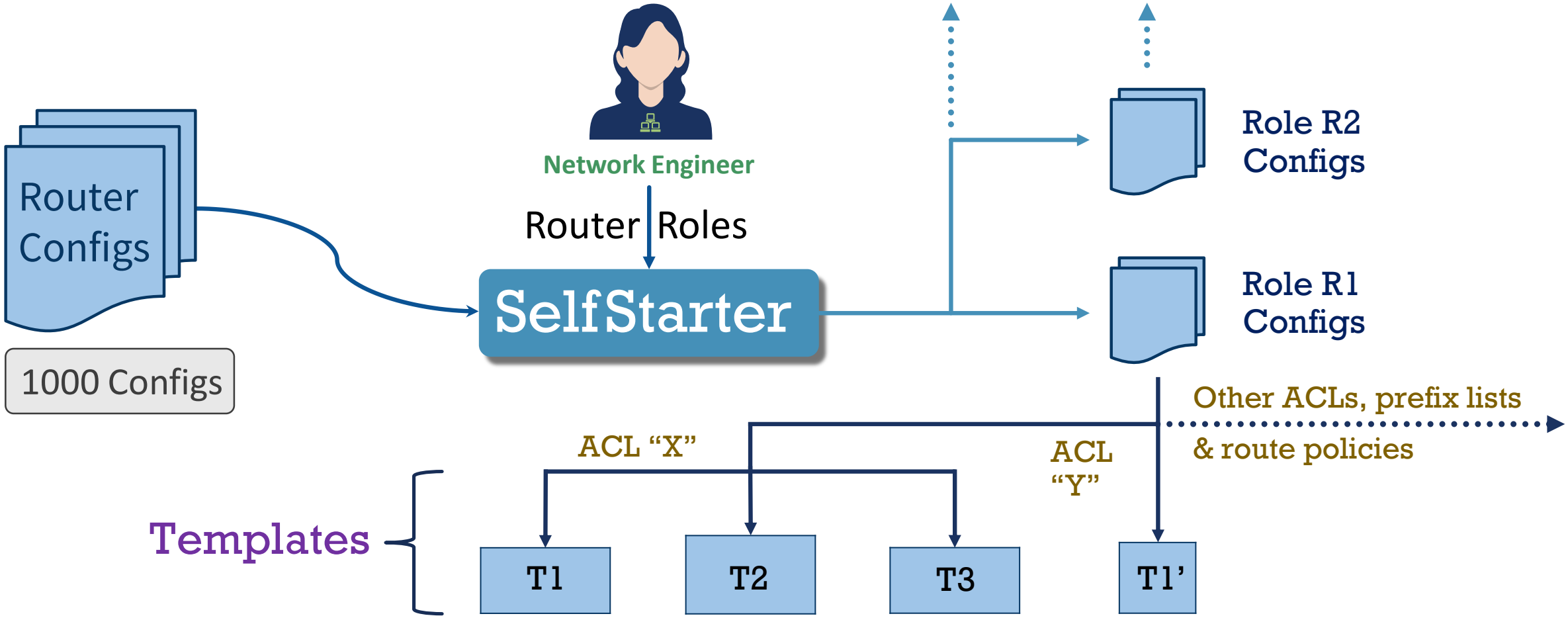
End-to-End Design



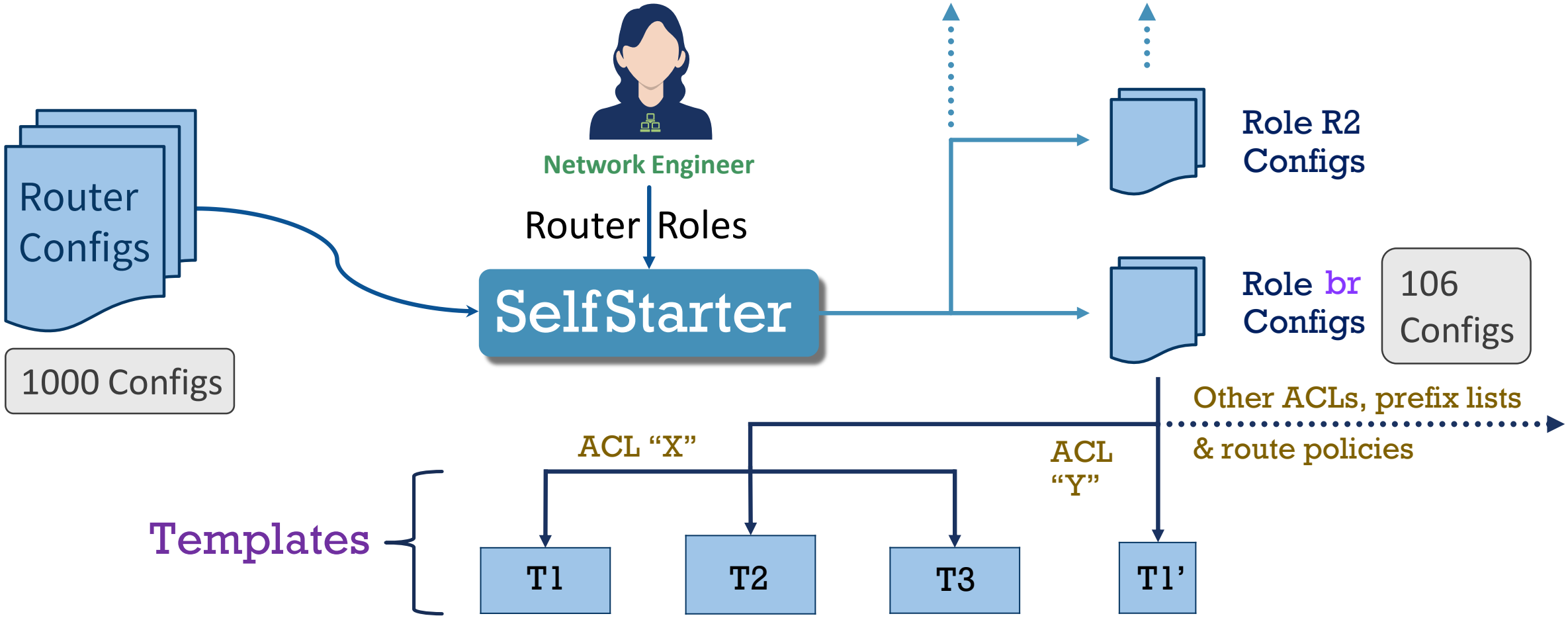
An Example: Campus Network



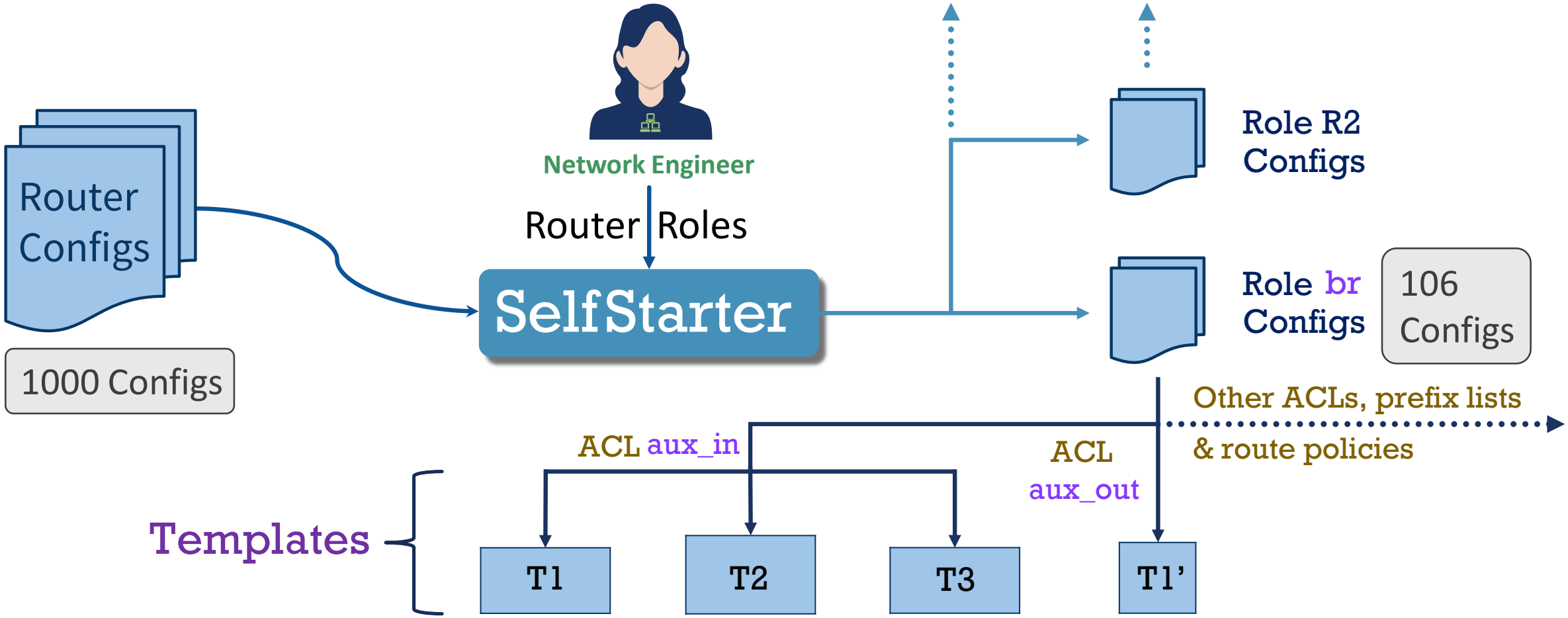
An Example: Campus Network



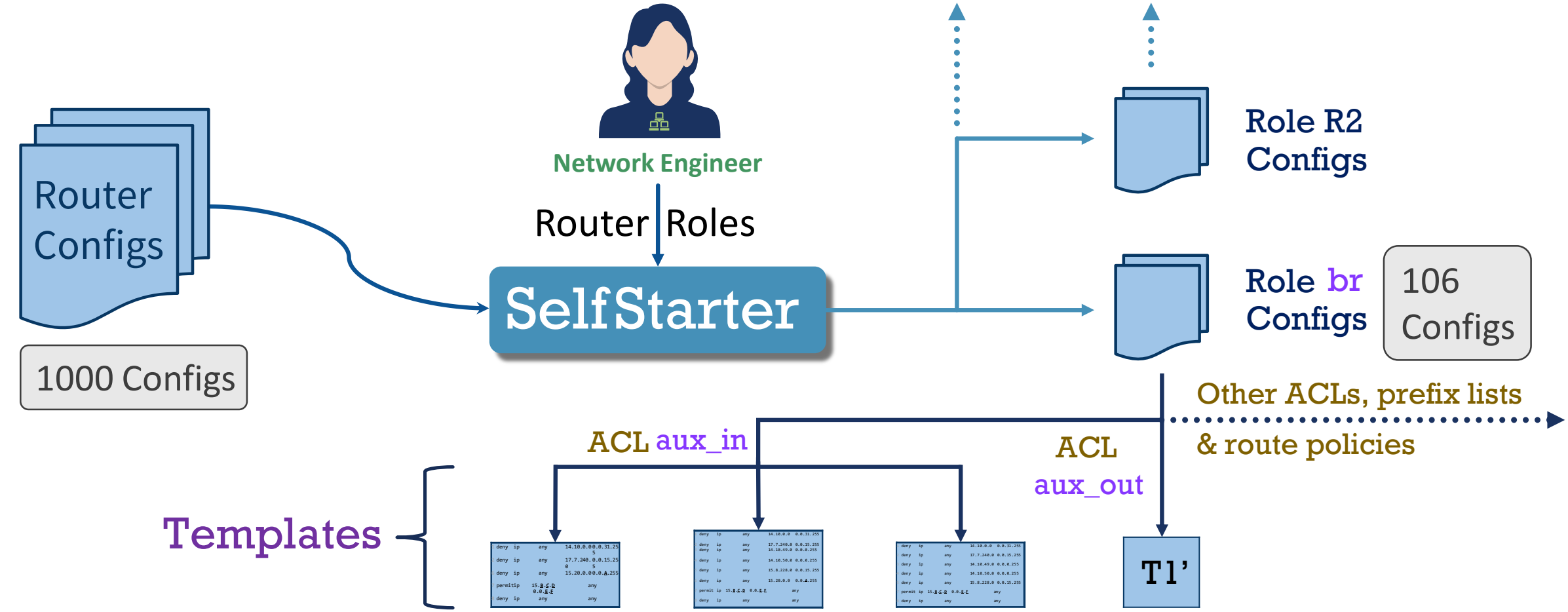
An Example: Campus Network



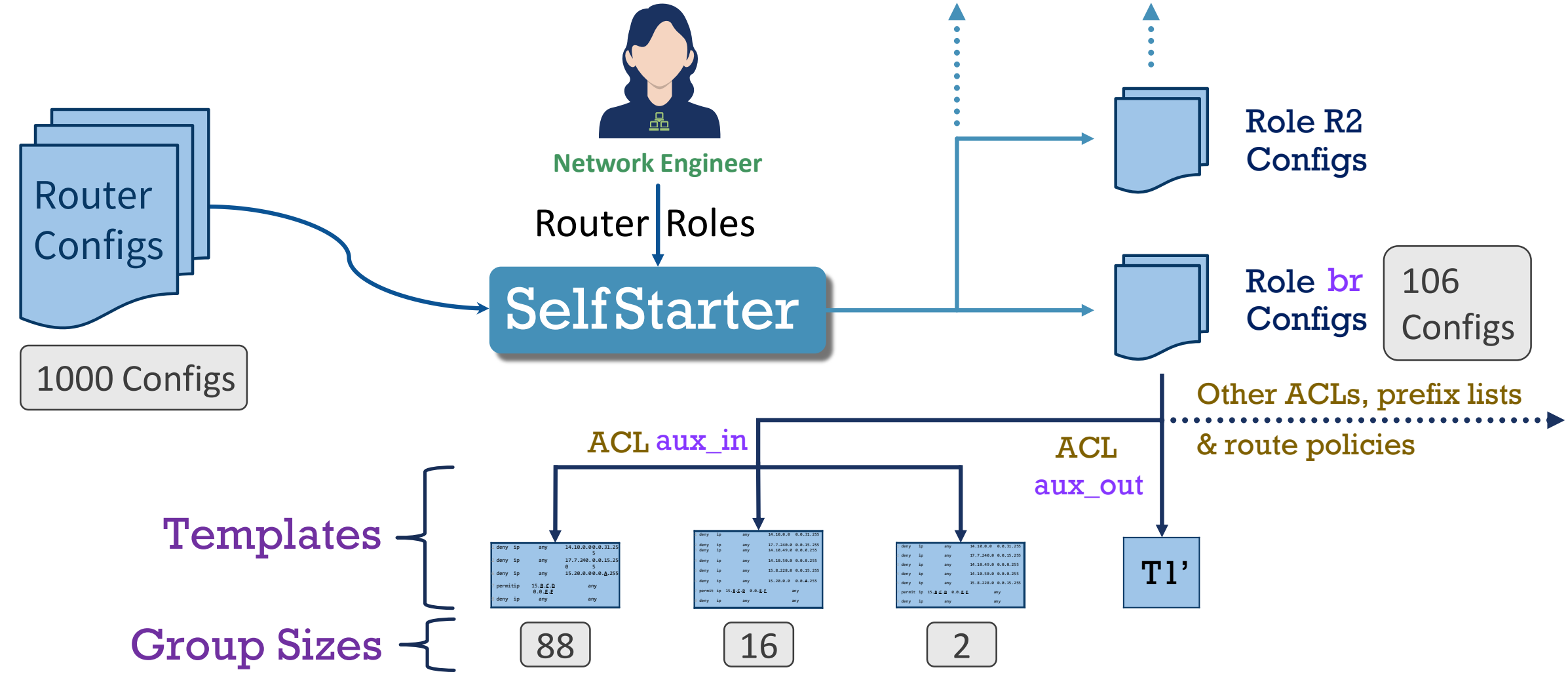
An Example: Campus Network



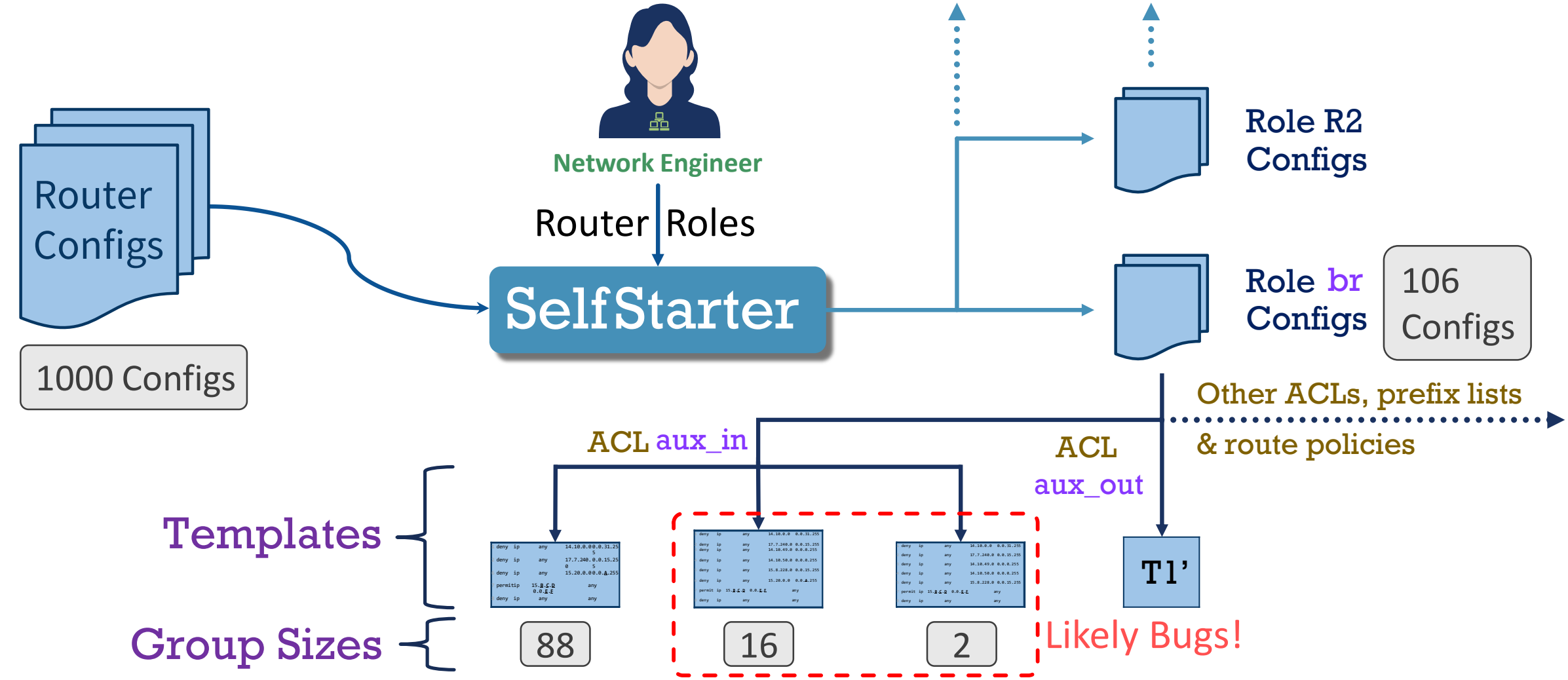
An Example: Campus Network



An Example: Campus Network



An Example: Campus Network



SelfStarter Example Output

- Input – 106 ACL configurations from a role in the campus network

SelfStarter Example Output

- Input – 106 ACL configurations from a role in the campus network

Metatemplate

```
1  deny    ip          any          14.10.0.0    0.0.31.255
2  deny    ip          any          17.7.240.0   0.0.15.255
3  deny    ip          any          14.10.49.0   0.0.0.255
4  deny    ip          any          14.10.50.0   0.0.0.255
5  deny    ip          any          15.8.228.0   0.0.15.255
6  deny    ip          any          15.20.0.0    0.0.A.255
7  permit  ip     15.B.C.D  0.0.E.F                 any
8  deny    ip          any                       any
```

Group 1 (88 ACLs)	Group 2 (16 ACLs)	Group 3 (2 ACLs)

SelfStarter Example Output

- Input – 106 ACL configurations from a role in the campus network

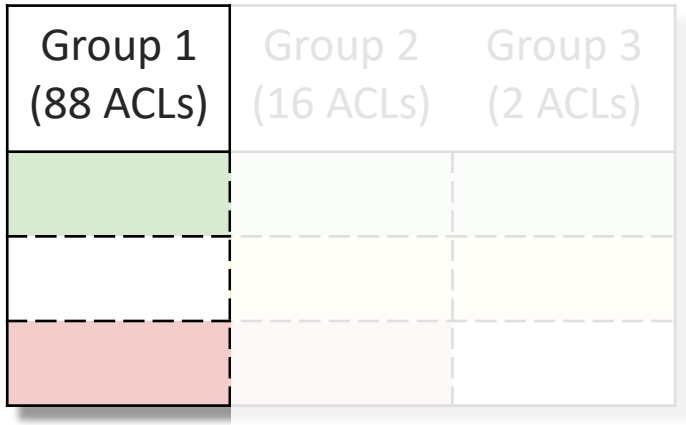
Metatemplate

```
1  deny    ip          any          14.10.0.0     0.0.31.255
2  deny    ip          any          17.7.240.0    0.0.15.255
3  deny    ip          any          14.10.49.0    0.0.0.255
4  deny    ip          any          14.10.50.0    0.0.0.255
5  deny    ip          any          15.8.228.0    0.0.15.255
6  deny    ip          any          15.20.0.0     0.0.A.255
7  permit  ip      15.B.C.D  0.0.E.F                 any
8  deny    ip          any                        any
```

Group 1 (88 ACLs)	Group 2 (16 ACLs)	Group 3 (2 ACLs)

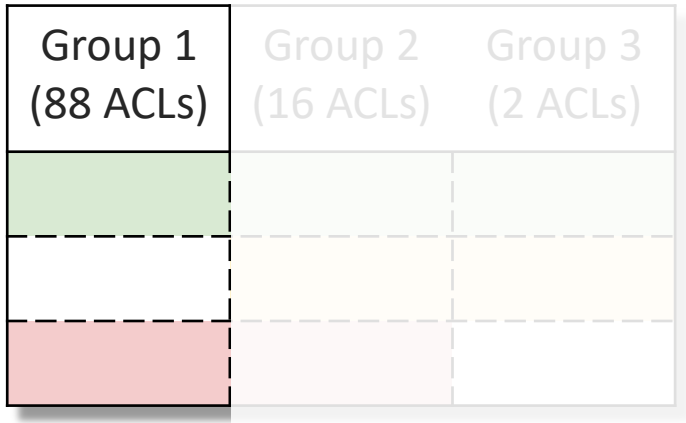
Group 1 Template

```
1   deny    ip          any          14.10.0.0     0.0.31.255
2   deny    ip          any          17.7.240.0    0.0.15.255
3   deny    ip          any          14.10.49.0    0.0.0.255
4   deny    ip          any          14.10.50.0    0.0.0.255
5   deny    ip          any          15.8.228.0    0.0.15.255
6   deny    ip          any          15.20.0.0     0.0.A.255
7   permit  ip      15.B.C.D  0.0.E.F                 any
8   deny    ip          any                        any
```



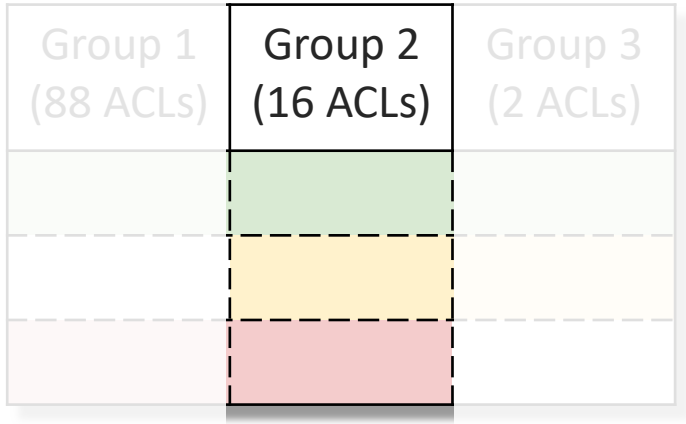
Group 1 Template

```
1   deny     ip              any            14.10.0.0     0.0.31.255
2   deny     ip              any            17.7.240.0    0.0.15.255
3   deny     ip              any            14.10.49.0    0.0.0.255
4   deny     ip              any            14.10.50.0    0.0.0.255
5   deny     ip              any            15.8.228.0    0.0.15.255
6   deny     ip              any            15.20.0.0     0.0.A.255
7   permit   ip       15.B.C.D  0.0.E.F                   any
8   deny     ip              any                          any
```



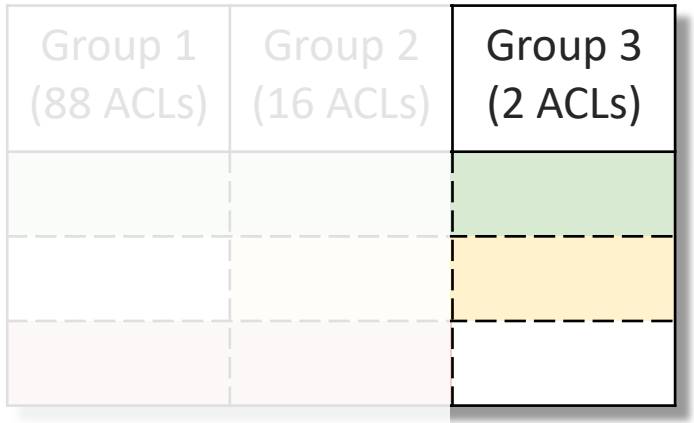
Group 2 Template

```
1   deny    ip              any             14.10.0.0     0.0.31.255
2   deny    ip              any             17.7.240.0    0.0.15.255
3   deny    ip              any             14.10.49.0    0.0.0.255
4   deny    ip              any             14.10.50.0    0.0.0.255
5   deny    ip              any             15.8.228.0    0.0.15.255
6   deny    ip              any             15.20.0.0     0.0.A.255
7   permit  ip      15.B.C.D  0.0.E.F                     any
8   deny    ip              any                         any
```



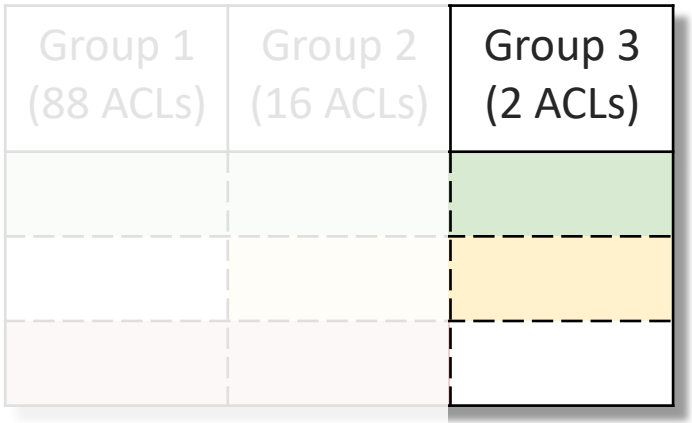
Group 3 Template

```
1   deny    ip          any          14.10.0.0     0.0.31.255
2   deny    ip          any          17.7.240.0    0.0.15.255
3   deny    ip          any          14.10.49.0    0.0.0.255
4   deny    ip          any          14.10.50.0    0.0.0.255
5   deny    ip          any          15.8.228.0    0.0.15.255
6   deny    ip          any          15.20.0.0     0.0.A.255
7   permit  ip      15.B.C.D  0.0.E.F                 any
8   deny    ip          any                        any
```



Group 3 Template

```
1   deny    ip          any          14.10.0.0     0.0.31.255
2   deny    ip          any          17.7.240.0    0.0.15.255
3   deny    ip          any          14.10.49.0    0.0.0.255
4   deny    ip          any          14.10.50.0    0.0.0.255
5   deny    ip          any          15.8.228.0    0.0.15.255
6   deny    ip          any          15.20.0.0     0.0.A.255
7   permit  ip      15.B.C.D  0.0.E.F                 any
8   deny    ip          any                        any
```

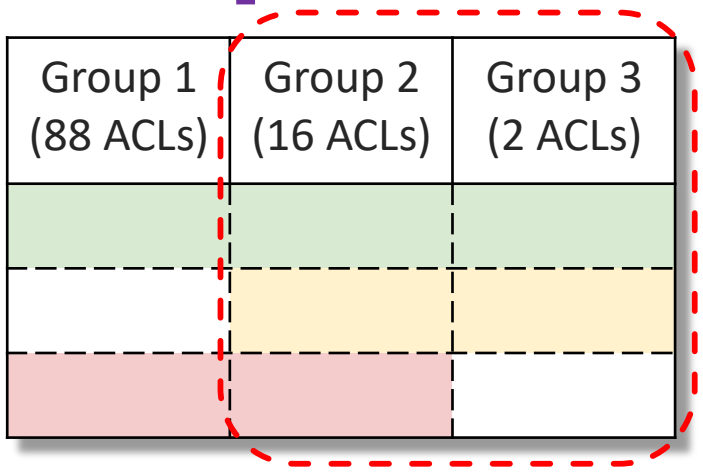


Deviance Identification

- |Groups 2 and 3| ≪ |Group 1| → Groups 2 and 3 ACLs are potentially misconfigured

1	deny	ip	any	14.10.0.0	0.0.31.255
2	deny	ip	any	17.7.240.0	0.0.15.255
3	deny	ip	any	14.10.49.0	0.0.0.255
4	deny	ip	any	14.10.50.0	0.0.0.255
5	deny	ip	any	15.8.228.0	0.0.15.255
6	deny	ip	any	15.20.0.0	0.0.A.255
7	permit	ip	15.B.C.D	0.0.E.F	any
8	deny	ip	any		any

Group Outliers

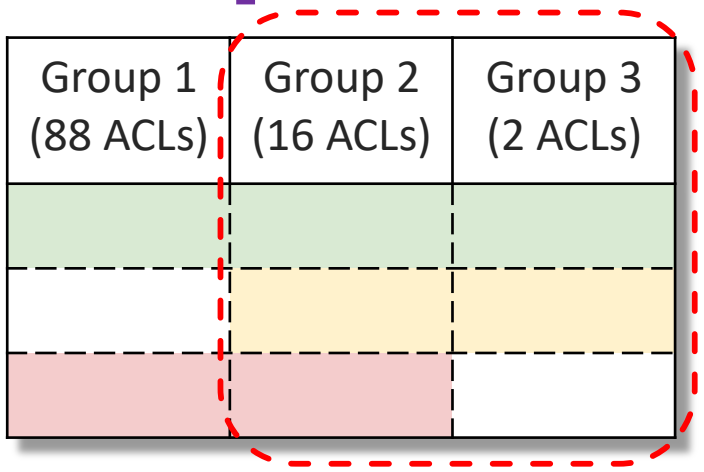


Deviance Identification

- |Groups 2 and 3| ≪ |Group 1| → Groups 2 and 3 ACLs are potentially misconfigured
- Metatemplate → Actionable feedback to the network engineers

```
1   deny     ip              any          14.10.0.0     0.0.31.255
2   deny     ip              any          17.7.240.0    0.0.15.255
3   deny     ip              any          14.10.49.0    0.0.0.255
4   deny     ip              any          14.10.50.0    0.0.0.255
5   deny     ip              any          15.8.228.0    0.0.15.255
6   deny     ip              any          15.20.0.0     0.0.A.255
7   permit   ip       15.B.C.D  0.0.E.F                 any
8   deny     ip              any                        any
```

Group Outliers



Deviance Identification

- $|\text{Groups 2 and 3}| \ll |\text{Group 1}| \rightarrow$ Groups 2 and 3 ACLs are potentially misconfigured
- Metatemplate → Actionable feedback to the network engineers
- $|\text{Group 1}|_{A=127} \ll |\text{Group 1}|_{A=255} \rightarrow$ 10 ACLs are potentially permitting more traffic than intended

```
1   deny     ip              any          14.10.0.0     0.0.31.255
2   deny     ip              any          17.7.240.0    0.0.15.255
3   deny     ip              any          14.10.49.0    0.0.0.255
4   deny     ip              any          14.10.50.0    0.0.0.255
5   deny     ip              any          15.8.228.0    0.0.15.255
6   deny     ip              any          15.20.0.0     0.0.A.255
7   permit   ip       15.B.C.D  0.0.E.F                any
8   deny     ip              any                       any
```

Group Outliers

Group 1 (88 ACLs)	Group 2 (16 ACLs)	Group 3 (2 ACLs)

Parameter Outliers

A	# of ACLs in Group 1
255	78
127	10

Deviance Identification

- $|\text{Groups 2 and 3}| \ll |\text{Group 1}| \rightarrow$ Groups 2 and 3 ACLs are ~~potentially~~ misconfigured
- Metatemplate → Actionable feedback to the network engineers
- $|\text{Group 1}|_{A=127} \ll |\text{Group 1}|_{A=255} \rightarrow$ 10 ACLs are ~~potentially~~ permitting more traffic than intended

```
1   deny     ip              any          14.10.0.0     0.0.31.255
2   deny     ip              any          17.7.240.0    0.0.15.255
3   deny     ip              any          14.10.49.0    0.0.0.255
4   deny     ip              any          14.10.50.0    0.0.0.255
5   deny     ip              any          15.8.228.0    0.0.15.255
6   deny     ip              any          15.20.0.0     0.0.A.255
7   permit   ip       15.B.C.D  0.0.E.F                any
8   deny     ip              any                       any
```

Group Outliers

Group 1 (88 ACLs)	Group 2 (16 ACLs)	Group 3 (2 ACLs)

Parameter Outliers

A	# of ACLs in Group 1
255	78
127	10

Template Inference: Key Ideas

```
ip access-list extended ACL1
1  deny    udp   host         0.0.0.0      any
2  permit  tcp   17.12.11.0   0.0.0.255    any
3  deny    icmp  17.12.11.0   0.0.0.255    any
4  permit  ip    16.21.0.0    0.0.63.255   any
5  permit  ip    17.12.11.0   0.0.0.255    any
```



```
ip access-list extended ACL2
1  deny    udp   host         0.0.0.0      any
2  permit  tcp   17.12.13.0   0.0.0.255    any
3  deny    icmp  17.12.13.0   0.0.0.255    any
4  permit  ip    17.12.13.0   0.0.0.255    any
5  permit  ip    16.23.0.0    0.0.63.255   any
```

Template Inference: Key Ideas

```
ip access-list extended ACL1
1  deny    udp   host         0.0.0.0      any
2  permit  tcp   17.12.11.0   0.0.0.255    any
3  deny    icmp  17.12.11.0   0.0.0.255    any
4  permit  ip    16.21.0.0    0.0.63.255   any
5  permit  ip    17.12.11.0   0.0.0.255    any
```

```
ip access-list extended ACL2
1  deny    udp   host         0.0.0.0      any
2  permit  tcp   17.12.13.0   0.0.0.255    any
3  deny    icmp  17.12.13.0   0.0.0.255    any
4  permit  ip    17.12.13.0   0.0.0.255    any
5  permit  ip    16.23.0.0    0.0.63.255   any
```

Challenge: Non-identical lines

Template Inference: Key Ideas

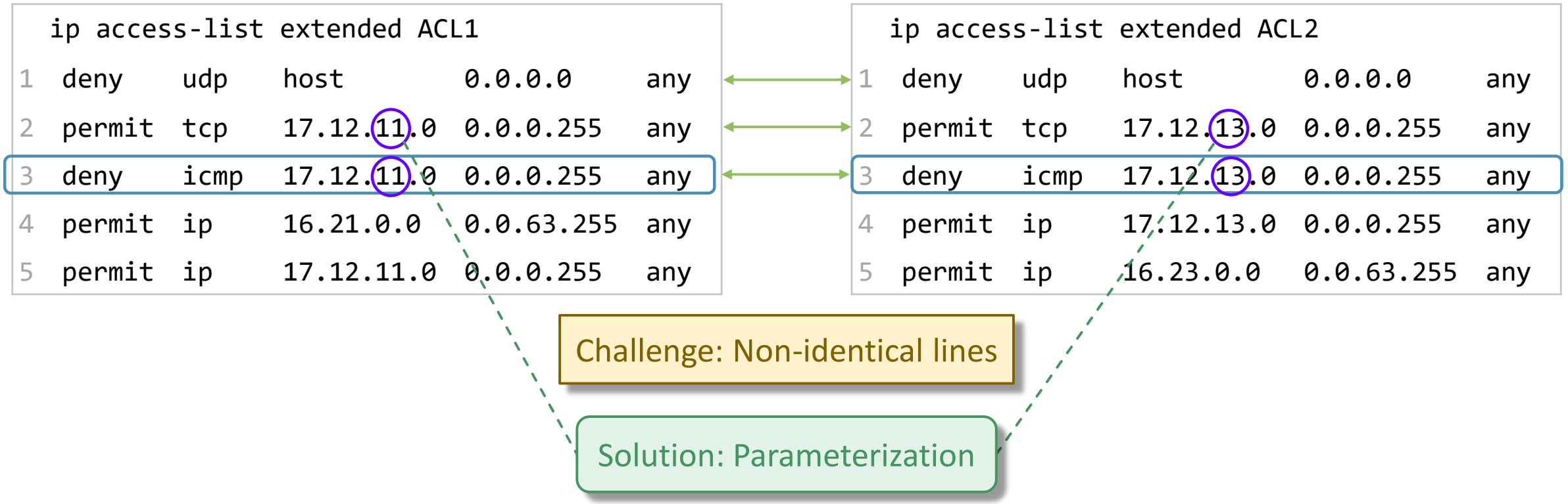
```
   ip access-list extended ACL1
1  deny    udp   host         0.0.0.0      any
2  permit  tcp   17.12.11.0   0.0.0.255    any
3  deny    icmp  17.12.11.0   0.0.0.255    any
4  permit  ip    16.21.0.0    0.0.63.255   any
5  permit  ip    17.12.11.0   0.0.0.255    any
```

```
   ip access-list extended ACL2
1  deny    udp   host         0.0.0.0      any
2  permit  tcp   17.12.13.0   0.0.0.255    any
3  deny    icmp  17.12.13.0   0.0.0.255    any
4  permit  ip    17.12.13.0   0.0.0.255    any
5  permit  ip    16.23.0.0    0.0.63.255   any
```

Challenge: Non-identical lines

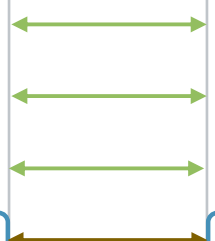
Solution: Parameterization

Template Inference: Key Ideas



Template Inference: Key Ideas

```
  ip access-list extended ACL1
1 deny    udp   host        0.0.0.0      any
2 permit  tcp   17.12.11.0  0.0.0.255    any
3 deny    icmp  17.12.11.0  0.0.0.255    any
4 permit  ip    16.21.0.0   0.0.63.255   any
5 permit  ip    17.12.11.0  0.0.0.255    any
```



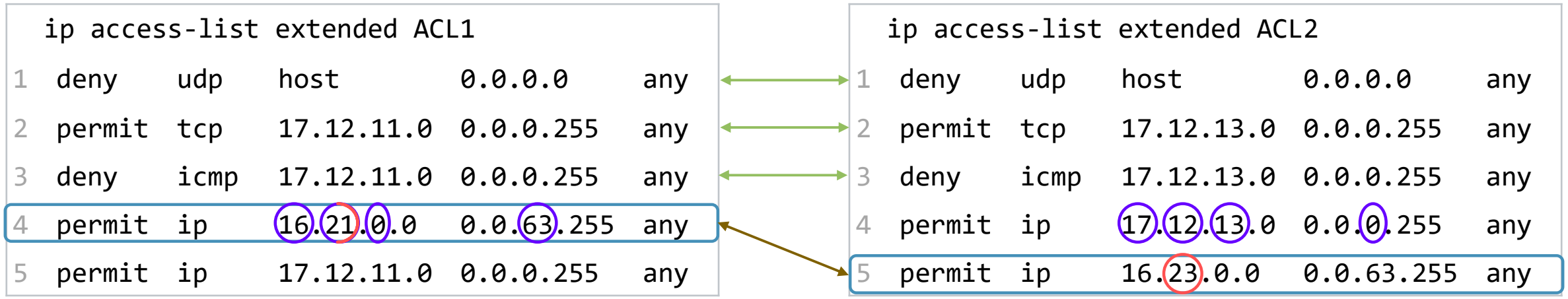
```
  ip access-list extended ACL2
1 deny    udp   host        0.0.0.0      any
2 permit  tcp   17.12.13.0  0.0.0.255    any
3 deny    icmp  17.12.13.0  0.0.0.255    any
4 permit  ip    17.12.13.0  0.0.0.255    any
5 permit  ip    16.23.0.0   0.0.63.255   any
```

Template Inference: Key Ideas

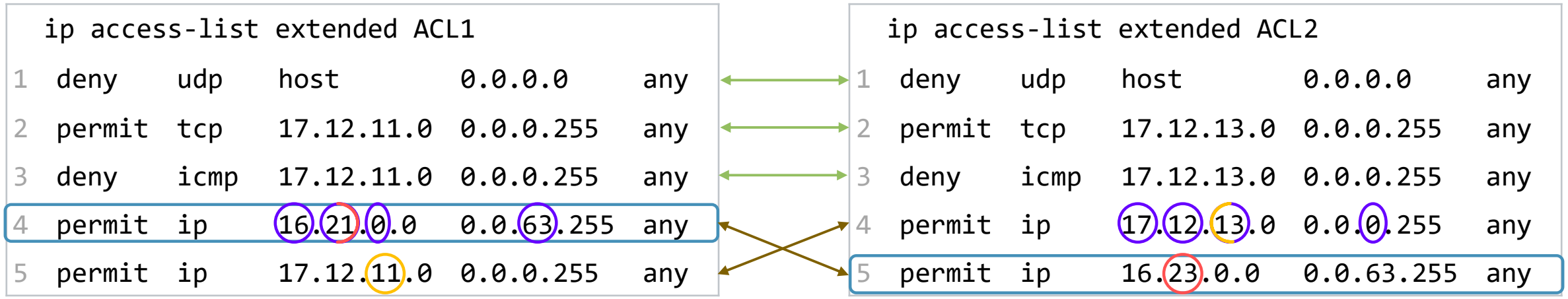
```
  ip access-list extended ACL1
1 deny    udp   host         0.0.0.0      any
2 permit  tcp   17.12.11.0   0.0.0.255    any
3 deny    icmp  17.12.11.0   0.0.0.255    any
4 permit  ip    16.21.0.0    0.0.63.255   any
5 permit  ip    17.12.11.0   0.0.0.255    any
```

```
  ip access-list extended ACL2
1 deny    udp   host         0.0.0.0      any
2 permit  tcp   17.12.13.0   0.0.0.255    any
3 deny    icmp  17.12.13.0   0.0.0.255    any
4 permit  ip    17.12.13.0   0.0.0.255    any
5 permit  ip    16.23.0.0    0.0.63.255   any
```

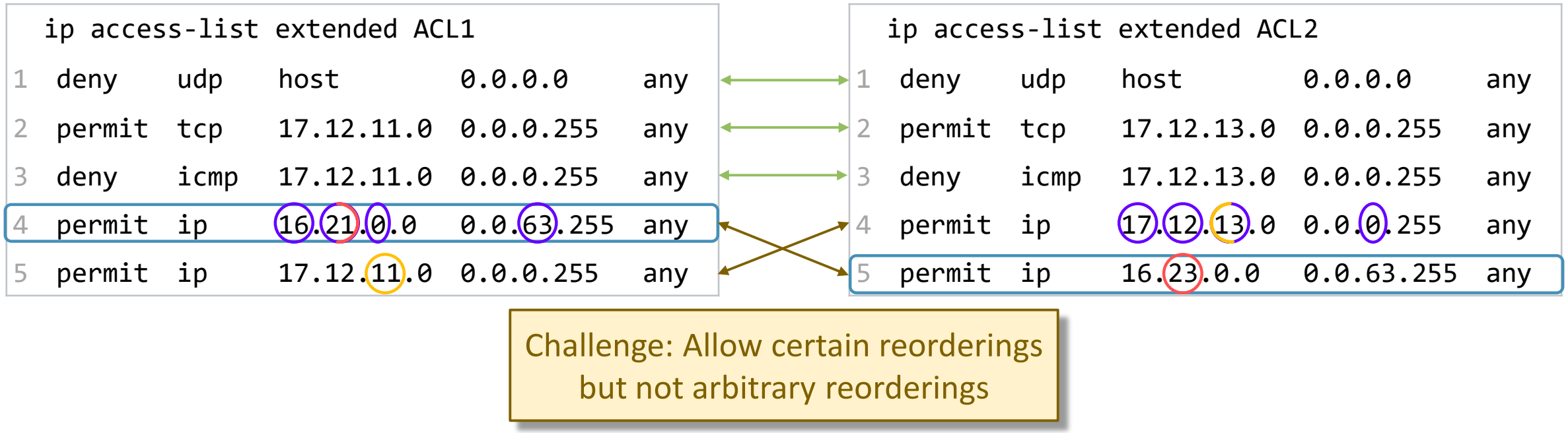
Template Inference: Key Ideas



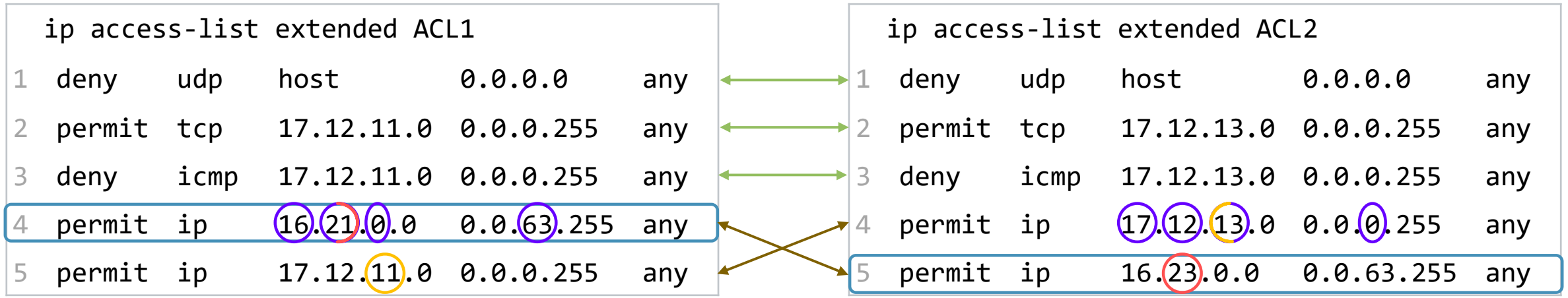
Template Inference: Key Ideas



Template Inference: Key Ideas



Template Inference: Key Ideas

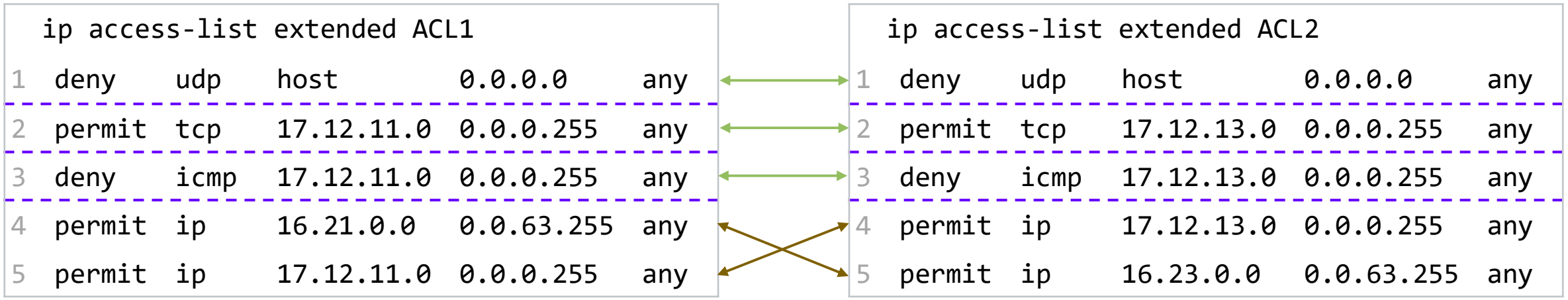


Challenge: Allow certain reorderings but not arbitrary reorderings

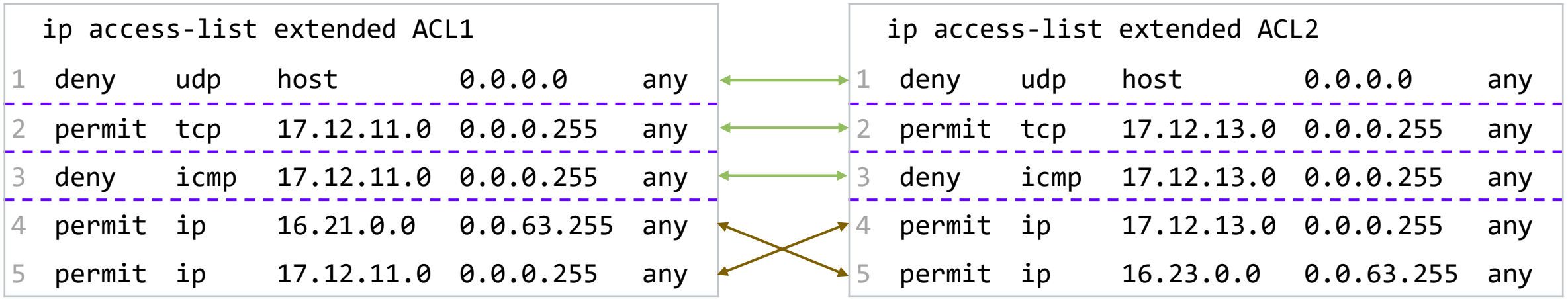
Solution: Two-level abstraction using blocks

A block is a contiguous sequence of lines that can be arbitrarily reordered but the order of blocks is important.

Template Inference: Key Ideas



Template Inference: Key Ideas



Block Alignment → *Sequence alignment* to prevent cross-block reordering

(see our paper for details)

Template Inference: Key Ideas

	ip access-list extended ACL1				
1	deny	udp	host	0.0.0.0	any
2	permit	tcp	17.12.11.0	0.0.0.255	any
3	deny	icmp	17.12.11.0	0.0.0.255	any
4	permit	ip	16.21.0.0	0.0.63.255	any
5	permit	ip	17.12.11.0	0.0.0.255	any

	ip access-list extended ACL2				
1	deny	udp	host	0.0.0.0	any
2	permit	tcp	17.12.13.0	0.0.0.255	any
3	deny	icmp	17.12.13.0	0.0.0.255	any
4	permit	ip	17.12.13.0	0.0.0.255	any
5	permit	ip	16.23.0.0	0.0.63.255	any

Block Alignment → *Sequence alignment* to prevent cross-block reordering

Line Reorderings → *Minimum-weight bipartite matching* to allow within-block line reorderings

(see our paper for details)

Template of ACL1 and ACL2

```
ip access-list extended ACL1
1  deny    udp   host        0.0.0.0      any
2  permit  tcp   17.12.11.0  0.0.0.255    any
3  deny    icmp  17.12.11.0  0.0.0.255    any
4  permit  ip    16.21.0.0   0.0.63.255   any
5  permit  ip    17.12.11.0  0.0.0.255    any
```

```
ip access-list extended ACL2
1  deny    udp   host        0.0.0.0      any
2  permit  tcp   17.12.13.0  0.0.0.255    any
3  deny    icmp  17.12.13.0  0.0.0.255    any
4  permit  ip    17.12.13.0  0.0.0.255    any
5  permit  ip    16.23.0.0   0.0.63.255   any
```

```
ip access-list extended ACL*
1  deny    udp   host        0.0.0.0      any
2  permit  tcp   17.12.A.0   0.0.0.255    any
3  deny    icmp  17.12.B.0   0.0.0.255    any
4  permit  ip    17.12.C.0   0.0.0.255    any
5  permit  ip    16.D.0.0    0.0.63.255   any
```

Results – Summary

- Triple: (segment type, role, segment name)

Results – Summary

- Triple: (segment type, role, segment name)

Network	Segment Type	Consistent Triples (only one group)	Inconsistent Triples (at least 2 groups)		
			Identified	Investigated	True Positives (% of investigated)
Campus	ACLs	0	6	3	3 (100%)
Microsoft WAN	Prefix lists	10042	166	138	7 (5%)
	Route policies	10969	56	33	33 (100%)
Microsoft Data center	ACLs	9700	938	400*	400 (100%)*
	Prefix lists	2954	0	-	-
	Route policies	11653	230	230*	230 (100%)*

* These were known inconsistencies

Results – Summary

- Triple: (segment type, role, segment name)

Network	Segment Type	Consistent Triples (only one group)		Inconsistent Triples (at least 2 groups)		
				Identified	Investigated	True Positives (% of investigated)
Campus	ACLs		**0**	6	3	3 (100%)
Microsoft WAN	Prefix lists	20 min	**10042**	166	138	7 (5%)
	Route policies		**10969**	56	33	33 (100%)
Microsoft Data center	ACLs		**9700**	938	400*	400 (100%)*
	Prefix lists	90 min	**2954**	0	-	-
	Route policies		**11653**	230	230*	230 (100%)*

* These were known inconsistencies

Results – Summary

- Triple: (segment type, role, segment name)

Network	Segment Type	Consistent Triples (only one group)	Inconsistent Triples (at least 2 groups)		
			Identified	Investigated	True Positives (% of investigated)
Campus	ACLs	0	6	3	**3 (100%)**
Microsoft WAN	Prefix lists	10042	166	138	**7 (5%)**
	Route policies	10969	56	33	**33 (100%)**
Microsoft Data center	ACLs	9700	938	400*	**400 (100%)***
	Prefix lists	2954	0	-	**-**
	Route policies	11653	230	230*	**230 (100%)***

* These were known inconsistencies

Results – Summary

- Triple: (segment type, role, segment name)

Network	Segment Type	Consistent Triples (only one group)	Inconsistent Triples (at least 2 groups)		
			Identified	Investigated	True Positives (% of investigated)
Campus	ACLs	0	6	3	**3 (100%)**
Microsoft WAN	Prefix lists	10042	166	138#	**7 (5%)#**
	Route policies	10969	56	33	**33 (100%)**
Microsoft Data center	ACLs	9700	938	400*	**400 (100%)***
	Prefix lists	2954	0	-	**-**
	Route policies	11653	230	230*	**230 (100%)***

#These prefix lists dynamically expand to multiple lines

* These were known inconsistencies

Results – Sources of Misconfigurations

- Campus
  - Decentralized management - Central and departmental
  - Reference templates for operators to follow

Results – Sources of Misconfigurations

- Campus
  - Decentralized management - Central and departmental
  - Reference templates for operators to follow

- × Inconsistent manual policy updates
- × Reference templates might be stale

Results – Sources of Misconfigurations

- Campus
  - Decentralized management - Central and departmental
  - Reference templates for operators to follow

- Microsoft WAN
  - Semi-automated via scripts
  - Implicit templates within scripts

- × Inconsistent manual policy updates
- × Reference templates might be stale

Results – Sources of Misconfigurations

- Campus
  - Decentralized management - Central and departmental
  - Reference templates for operators to follow

- × Inconsistent manual policy updates
- × Reference templates might be stale

- Microsoft WAN
  - Semi-automated via scripts
  - Implicit templates within scripts

- × Bug in the scripts
- × Erroneous manual updates

Results – Sources of Misconfigurations

- Campus
  - Decentralized management - Central and departmental
  - Reference templates for operators to follow

× Inconsistent manual policy updates
× Reference templates might be stale

- Microsoft WAN
  - Semi-automated via scripts
  - Implicit templates within scripts

× Bug in the scripts
× Erroneous manual updates

- Microsoft Data center
  - Automated update service
  - Explicit templates for each role

Results – Sources of Misconfigurations

- Campus
  - Decentralized management - Central and departmental
  - Reference templates for operators to follow

× Inconsistent manual policy updates
× Reference templates might be stale

- Microsoft WAN
  - Semi-automated via scripts
  - Implicit templates within scripts

× Bug in the scripts
× Erroneous manual updates

- Microsoft Data center
  - Automated update service
  - Explicit templates for each role

× Delayed updates during automation

Prior Work

- **Bayesian Detection of Router Configuration Anomalies:**
  - Structured Bayes [2005 Khalid et al]
    - Misconfigurations in router data are identified as statistical anomalies within a Bayesian framework.

Prior Work

- **Bayesian Detection of Router Configuration Anomalies:**
  - Structured Bayes [2005 Khalid et al]
    - Misconfigurations in router data are identified as statistical anomalies within a Bayesian framework.

- **Detecting Network-Wide and Router-Specific Misconfigurations Through Data Mining:**
  - Minerals [2009 Le et al]
    - Applied association rules mining to the configuration files

Prior Work

- **Bayesian Detection of Router Configuration Anomalies:**
  - Structured Bayes [2005 Khalid et al]
    - Misconfigurations in router data are identified as statistical anomalies within a Bayesian framework.

- **Detecting Network-Wide and Router-Specific Misconfigurations Through Data Mining:**
  - Minerals [2009 Le et al]
    - Applied association rules mining to the configuration files

Key Limitations

- ✕ Exact equivalence-based similarity
- ✕ Do not apply to complex configuration segments like ACLs and Prefix lists
- ✕ Allow arbitrary reordering of lines

Conclusion

- First general template inference algorithm for configuration segments
- SelfStarter – Our tool for finding potential network misconfigurations without a specification, using automatic template inference
- Provides actionable feedback to the operators
- Found 43 unknown bugs in Microsoft networks and campus network
- Source Code : https://github.com/SivaKesava1/SelfStarter
- Reach me at: sivakesava@cs.ucla.edu