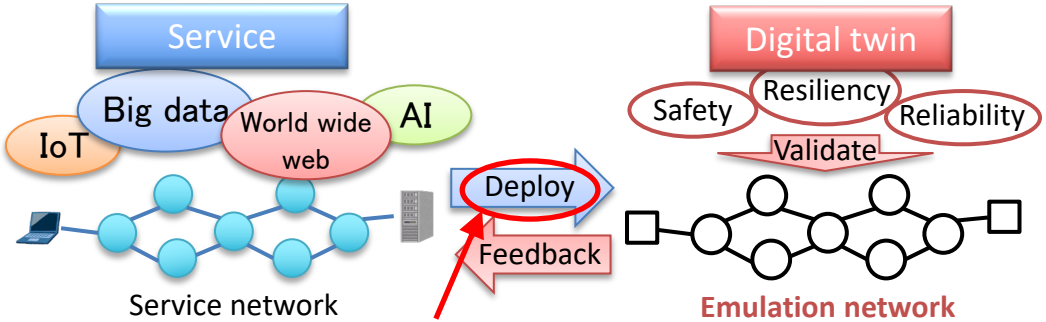
dot2net: A labeled graph approach for template-based configuration of emulation networks

Satoru Kobayashi[1], Ryusei Shiiba[2], Ryosuke Miura[3],
Shinsuke Miwa[3], Toshiyuki Miyachi[3], Kensuke Fukuda[24]
1: Okayama Univ, 2: Sokendai, 3: NICT, 4: NII

November 1, 2023

Network management with digital twins

- Digital twins
  - Digital copy in virtual environment for testing and simulation
- Network emulation (digital twins of networks)
  - Verify safety and fault tolerance of networks without affecting the network services



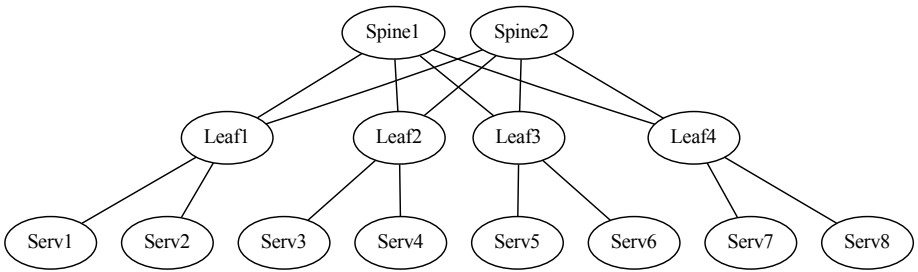
Configuring emulation networks

- Problem: Time-consuming to construct emulation networks
  - We cannot use production network configuration directly
    - Difference of devices or softwares -> Difference of configuration formats
    - Difference of parameter assignment such as IP addresses
  - It is time-consuming to configure large-scale networks
    - Describe a bunch of config lines similar to other devices or interfaces
    - Using “copy-and-paste” strategy sometimes cause errors due to a lack of consideration of parameter changes

Example of configuration failures for emulation networks

- TiNET [1]
  - Platform for Docker-based emulation networks
  - Provide various example network configurations

Example topology (3-tier CLOS)



Official configuration example of TiNET

```
39  - name: Leaf3
40    image: slankdev/frr
41    interfaces:
42      - { name: up1, type: direct, args: Spine1#dn3 }
43      - { name: up2, type: direct, args: Spine2#dn3 }
44      - { name: dn1, type: direct, args: Serv3#net0 }
45      - { name: dn2, type: direct, args: Serv4#net0 }
46  - name: Leaf4
47    image: slankdev/frr
48    interfaces:
49      - { name: up1, type: direct, args: Spine1#dn4 }
50      - { name: up2, type: direct, args: Spine2#dn4 }
51      - { name: dn1, type: direct, args: Serv3#net0 }
52      - { name: dn2, type: direct, args: Serv4#net0 }
```

Duplicated interfaces

```
59  - name: Serv3
60    image: slankdev/frr
61    interfaces: [ { name: net0, type: direct, args: Leaf2#dn1 } ]
62  - name: Serv4
63    image: slankdev/frr
64    interfaces: [ { name: net0, type: direct, args: Leaf2#dn2 } ]
```

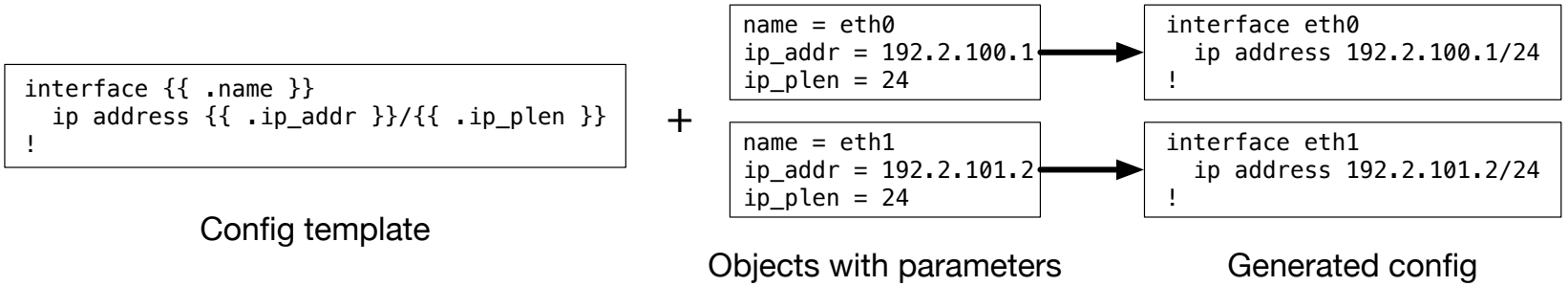
Inconsistent specification

https://github.com/tinynetwork/tinet/pull/86

[1] TiNET, https://github.com/tinynetwork/tinet/

Batch configuration of network devices

- Config templates
  - Embed parameters to templates to configure multiple objects



- Difficulty in templating network configurations
  1. Limitation of per-device templates
  2. Difficulty for complicated network topologies

Difficulty in templating network configurations

1. Limitation of per-device templates

Per-device template with control syntaxes

```
ip forwarding
!
{% for interface in data.interfaces %}
interface {{ interface.id }}
  ip address {{ interface.ip }}/{{ interface.plen }}
!
{% endfor %}
{% if data.ospf.enabled %}
router ospf
  ospf router-id {{ data.loopback }}
{% for network in data.ospf.networks %}
  network {{ network }} area 0
{% endfor %}
{% endif %}
!
```

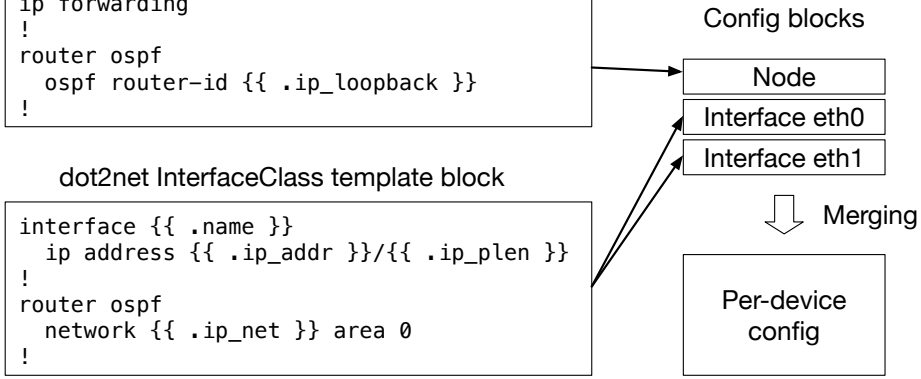
(a) Traditional templating approach

dot2net NodeClass template block

```
ip forwarding
!
router ospf
  ospf router-id {{ .ip_loopback }}
!
```

dot2net InterfaceClass template block

```
interface {{ .name }}
  ip address {{ .ip_addr }}/{{ .ip_plen }}
!
router ospf
  network {{ .ip_net }} area 0
!
```



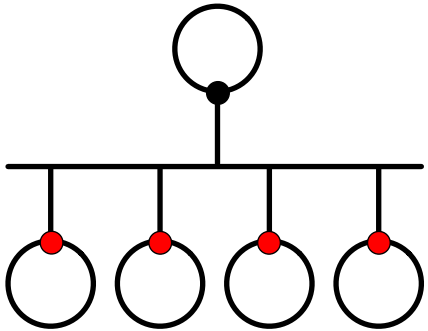
(b) Proposed approach (dot2net)

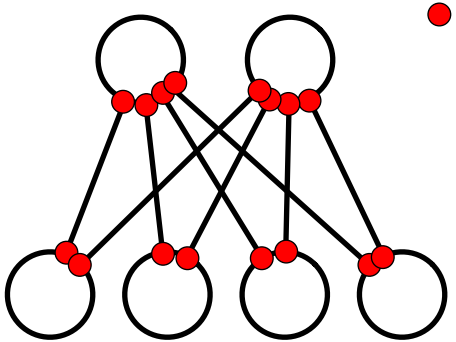
Existing tools are per-device
Requires control syntaxes (for and if)

Expect <u>appropriate granularity templates</u> without control syntaxes

Difficulty in templating network configurations

2. Difficulty for complicated network topologies





Simple network topology:
Consider only one of both link ends in parameter assignment

Complicated network topology:
- Necessary to consider combinations of link ends in parameter assignment
- Much more parameter values

Requirements for emulation network configuration platform

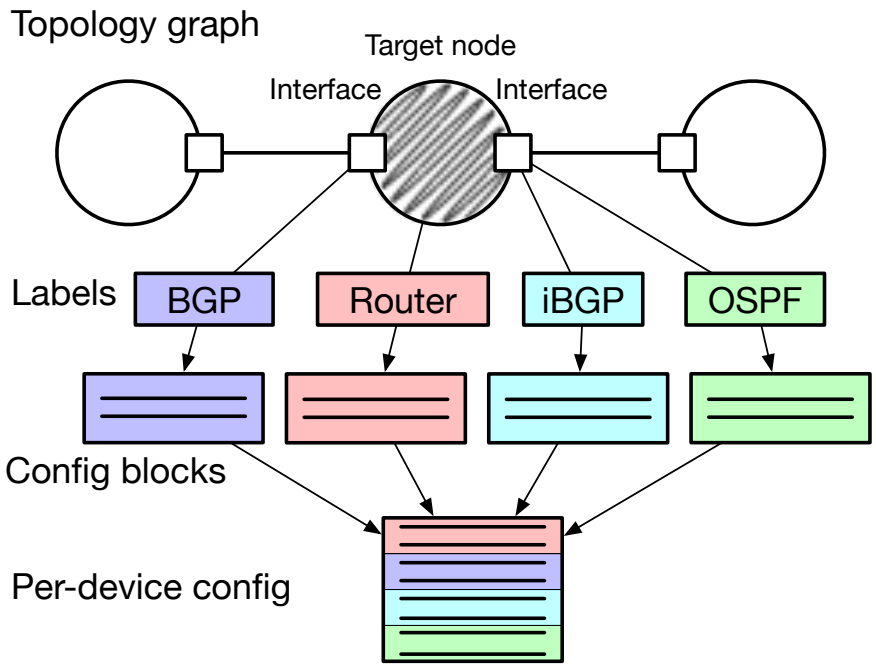
- Description simplicity
  - Efficient description without duplications
- Scalability for larger networks
  - Automated parameter assignment
  - Support large-scale networks (with hundreds of devices)
- Expressiveness for complicated networks
  - Accept advanced technologies for emulation

Research goal

Emulation network configuration platform dot2net that meets description simplicity, scalability, and expressiveness

Key idea: Separate network topology and configuration

- Network topology description as a labeled graph
  - Describe roles of devices or interfaces with labels
  - Generate config blocks with templates corresponding to labels
  - Obtain traditional per-device config by merging config blocks
- ➢ Easy to change topology by modifying the labeled graph



Five design principles of dot2net

1. Separate network topology and configuration (Key idea)
2. Declarative config description (for simplicity)
   - Easy to validate with external tools
3. No control syntax macros in templates (for simplicity)
   - Config template can be described in more simple format
4. Minimum manual parameter specification (for scalability)
   - Automatically assign parameters -> Decrease human failures
5. Accept advanced network technologies (for expressiveness)
   - Available for testing new network technologies

Two challenges to meet design principles

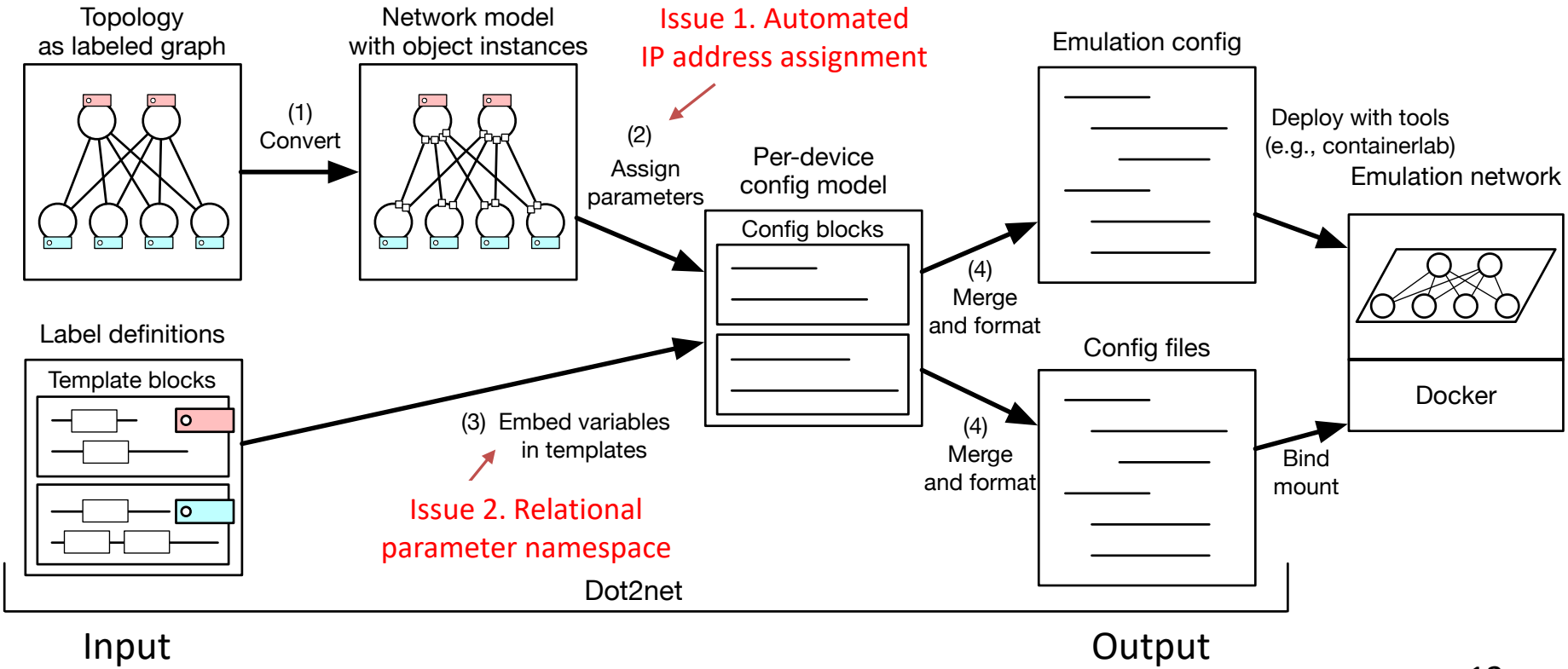
Challenge 1: Automated IP address assignment

- – Unlike other parameters, we need to consider subnet consistency
- – How to handle advanced network technologies?
- ➢ Issue 1: Automated IP address assignment considering network layers

Challenge 2: Relational parameter reference from templates

- – How to refer parameters from templates without control syntax?
- – e.g., Cannot refer parameters in lists because no “for” syntax
- ➢ Issue 2: Relational parameter namespace

Overview of dot2net architecture



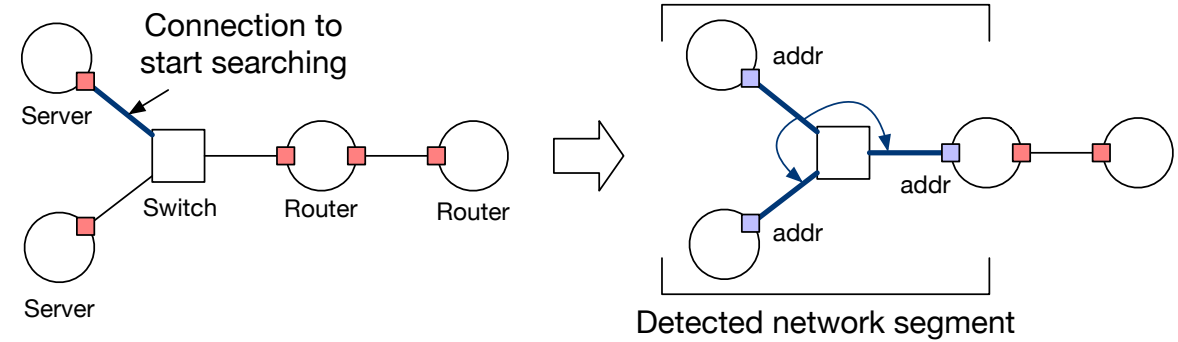
Issue 1. Automated IP address assignment considering network layers

There are two major challenges:

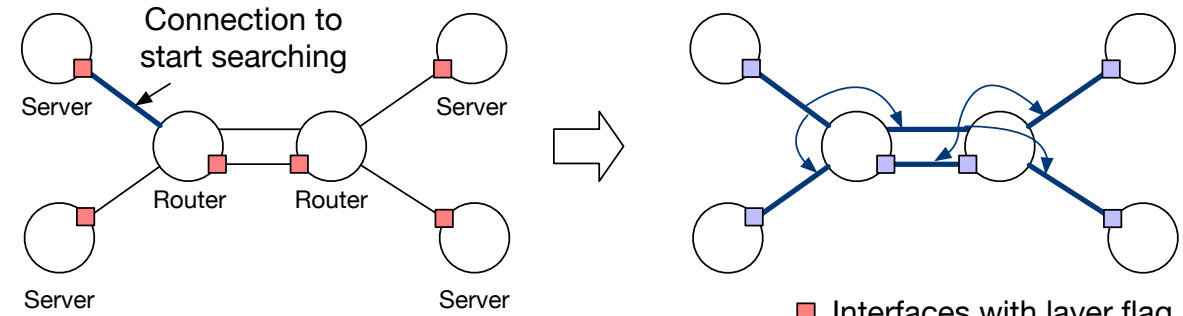
A) How to determine IP subnets (network segments)
- Adjacent devices must have IP address of same IP subnet
- Some devices (e.g., L2 switches) do not have IP addresses

B) How to support advanced network technologies
- Advanced network protocols basically have network layers
- Layer-separated IP address assignment enables many of them

1.A) IP subnet decision algorithm



(a) A network topology with L2 switch



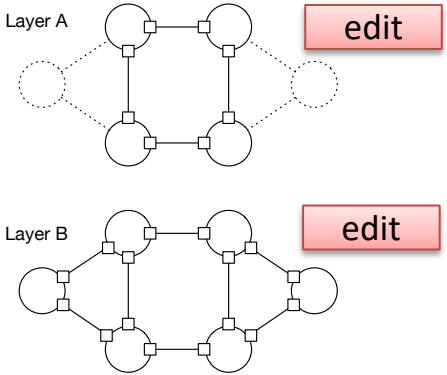
(b) A network topology with L2 tunneling

1.B) Layers in dot2net labeled graph

- Intuitive: Describe topologies by layers (Layer approach)
  - Need to modify all layers when adding a device to topology
  - Cause errors by emissions in modifications
- Proposed: Specify layers in label definitions (Label approach)
  - Users only need to modify one labeled graph for topology changes

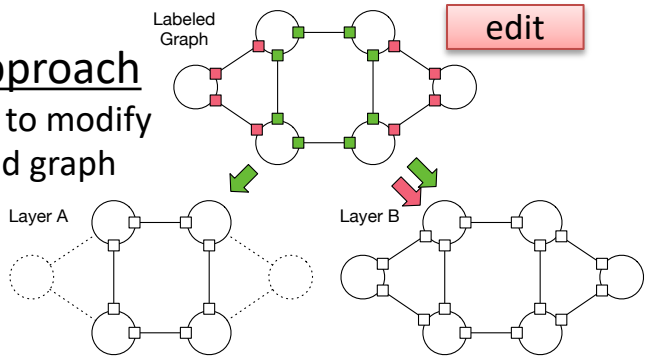
Layer approach

Need to modify all layers for topology changes



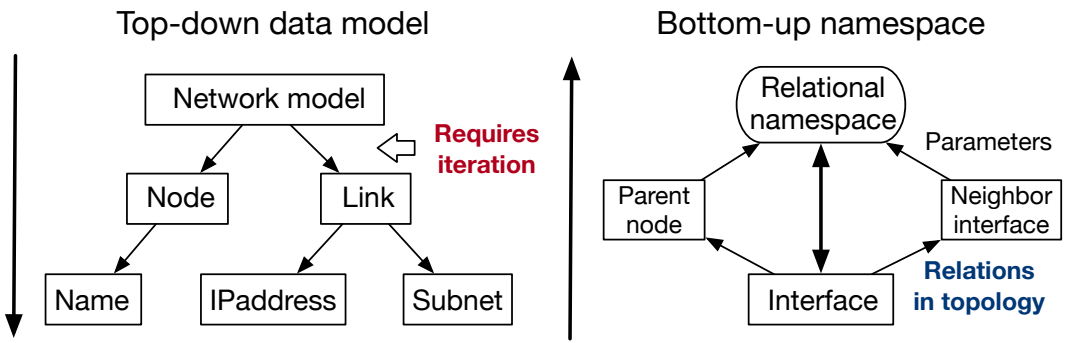
Label approach

Need only to modify one labeled graph



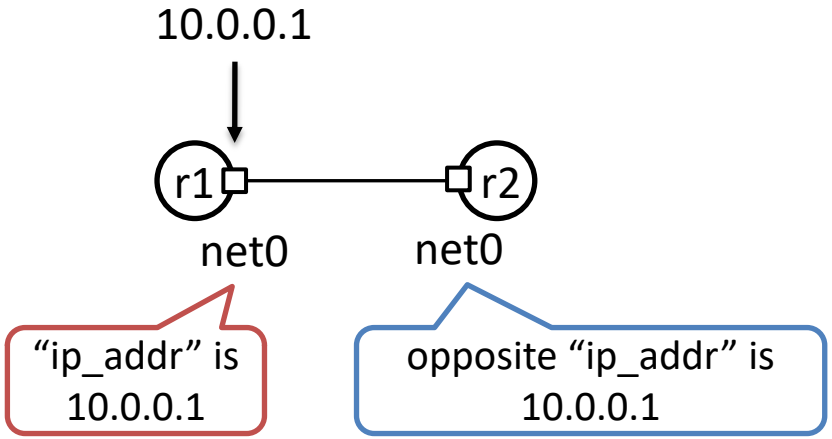
Issue 2. Relational parameter namespace

- Traditional parameter data model is top-down
  - Requires control syntax (for, if) to specify parameter placement
- Proposed parameter namespace is bottom-up
  - Parameters are specified with relations to the object for config generation (devices or interfaces)



```
r1 {{ .name }} = r1
r1.net0 {{ .ip_addr }} = 10.0.0.1
r1.net0 {{ .ip_net }} = 10.0.0.0/24
r1.net0 {{ .ip_plen }} = 24
r1.net0 {{ .name }} = net0
r1.net0 {{ .node_name }} = r1
r1.net0 {{ .opp_ip_addr }} = 10.0.0.2
r1.net0 {{ .opp_ip_net }} = 10.0.0.0/24
r1.net0 {{ .opp_ip_plen }} = 24
r1.net0 {{ .opp_name }} = net0
r1.net0 {{ .opp_node_name }} = r2
r2 {{ .name }} = r2
r2.net0 {{ .ip_addr }} = 10.0.0.2
r2.net0 {{ .ip_net }} = 10.0.0.0/24
r2.net0 {{ .ip_plen }} = 24
r2.net0 {{ .name }} = net0
r2.net0 {{ .node_name }} = r2
r2.net0 {{ .opp_ip_addr }} = 10.0.0.1
r2.net0 {{ .opp_ip_net }} = 10.0.0.0/24
r2.net0 {{ .opp_ip_plen }} = 24
r2.net0 {{ .opp_name }} = net0
r2.net0 {{ .opp_node_name }} = r1
...
```

Example of namespace



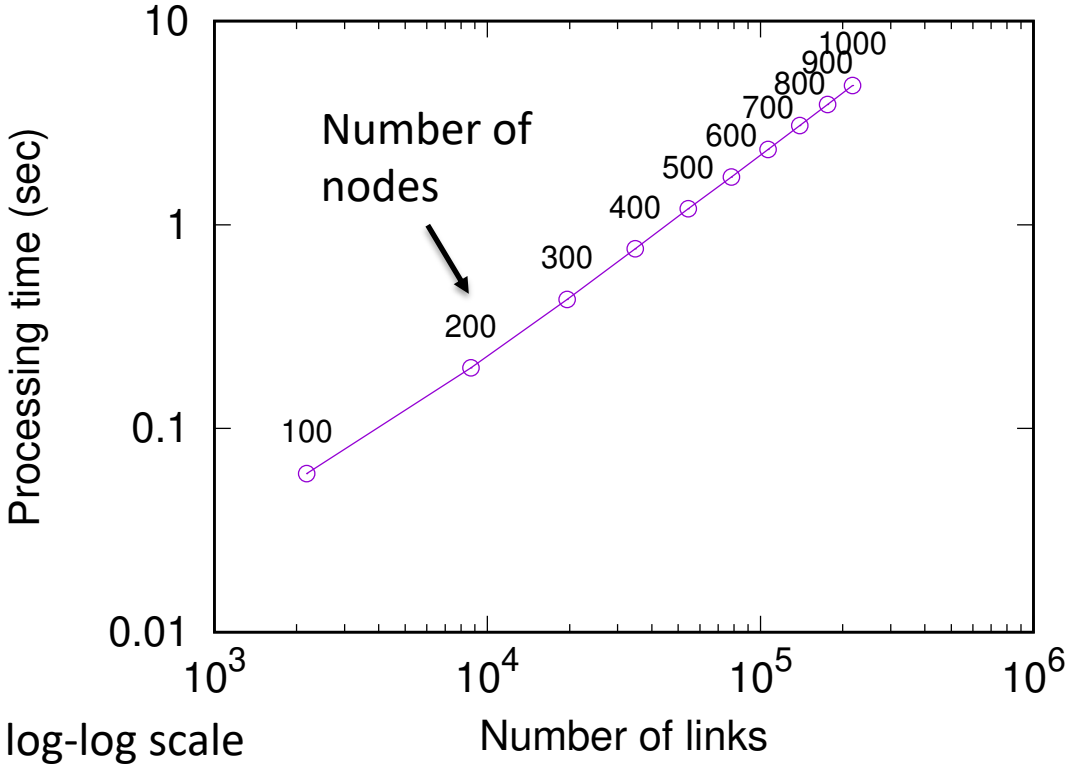
dot2net implementation

- Implemented in Go language, publicly available as OSS
  - https://github.com/cpflat/dot2net
- Input: A labeled graph (DOT) and label definitions (YAML)
- Output: Config files for Docker-based network emulation platforms such as Containerlab or TiNET
  - Including emulation network structure and router software settings
- Dot2net can automate deploying emulation networks by collaborating with above network emulation platforms

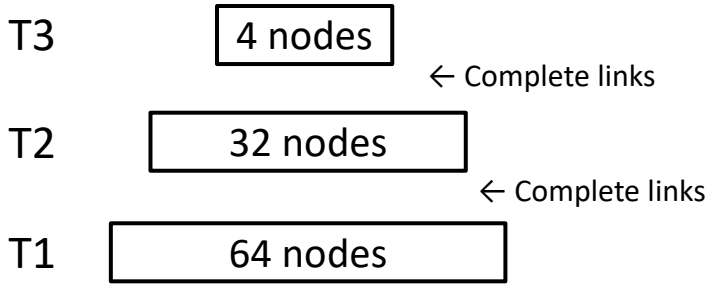
Evaluation of dot2net

- Performance evaluation (Scalability)
  - Evaluate processing time for generating configuration files of large-scale CLOS networks with hundreds of devices
- Description efficiency evaluation (Description simplicity)
  - Describe five testing network for FRRouting with dot2net
  - Evaluate configuration changes (bytes) on simple topology changes
- Expressiveness evaluation (Expressiveness)
  - Discuss how to describe network topologies with advanced technologies on the above testing networks

Performance evaluation (processing time)



3-tier CLOS networks (100 nodes)



Dot2net can generate config files of a network with 1,000 devices only in 3.5 seconds

Note: containerlab takes more than 5 minutes to deploy a network with 100 devices

Description efficiency evaluation (bytes)

Scenario	(Expansion)	dot2net Topology	(diff)	Config	(diff)	Total	(diff)	Containerlab	(diff)	TiNET	(diff)
rip_topo1		536		3,128		3,664		7,140		7,509	
rip_topo1	(+1)	591	(+55)	3,128	(±0)	3,719	(+55)	8,929	(+1,789)	9,323	(+1,814)
ospf_topo1		539		2,962		3,501		11,547		11,906	
ospf_topo1	(+1)	582	(+43)	2,962	(±0)	3,544	(+43)	13,708	(+2,161)	14,085	(+2,179)
ospf6_topo1		982		2,813		3,795		9,872		10,265	
ospf6_topo1	(+1)	1,026	(+44)	2,813	(±0)	3,839	(+44)	11,762	(+1,890)	12,180	(+1,915)
bgp_features		1,037		6,234		7,271		19,119		19,815	
bgp_features	(+1)	1,122	(+85)	6,234	(±0)	7,356	(+85)	21,999	(+2,880)	22,713	(+2,898)
bgp_evpn_vxlan_topo1		777		4,171		4,948		13,787		14,088	
bgp_evpn_vxlan_topo1	(+1)	862	(+85)	4,171	(±0)	5,033	(+85)	15,024	(+1,237)	15,345	(+1,257)
basic_clos		858		1,275		2,133		9,300		10,275	
basic_clos	(+1)	909	(+51)	1,275	(±0)	2,184	(+51)	9,932	(+632)	10,970	(+695)

(+1) means
one node and one link
are added to the topology

(diff) is the difference of
configuration files
on the expansion

Additional config
description is reduced to
less than 10%

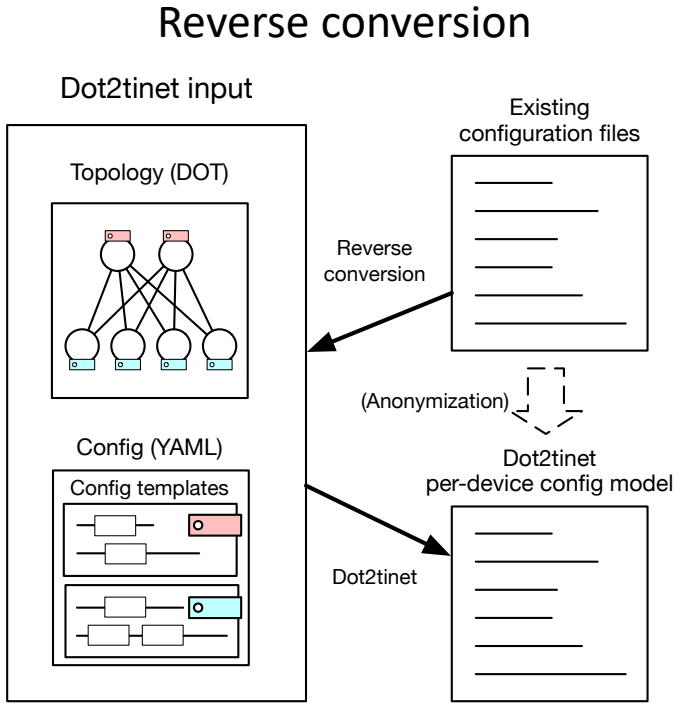
Discussion

- Dot2net does not depend on specific network protocol
  - Many existing methods [3,4] only support limited protocols (e.g., BGP)
  - Dot2net just generate config description -> no protocol limitation
- It is still difficult to handle some technologies with dot2net
  - ACLs (depend on service policy rather than network topology)
  - Source routing policies (DOT cannot describe paths)
  - Value lists (not reasonable for one-to-one parameter namespace)

[3] S. Knight et al. “An automated system for emulated network experimenta tion,” in Proceedings of CoNEXT ’13, pp. 235–246, 2013
[4] R. Beckett et al. “Don’t mind the gap: Bridging network-wide objectives and device-level config urations,” in Proceedings of SIGCOMM ’16, pp. 328–341, 2016

Future work

- Incremental config deployment
  - Dot2net currently generate all config files for each time of topology changes
  - Due to emulation platforms such as Containerlab and TiNET
- Reverse conversion from existing config
  - Support topology changes of existing network configurations
  - Support config anonymization (for data publication)



Conclusion

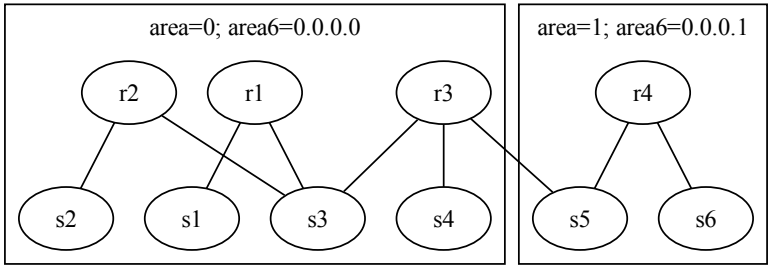
- Proposed design and implementation of dot2net, a template-based configuration platform
  - For simple, scalable, and expressive configuration description
  - Separate network topology as labeled graph and label definitions as config template blocks
  - Available in https://github.com/cpflat/dot2net
- Demonstrate the effectiveness in performance and description efficiency with test topologies for FRRouting
  - Dot2net generates network configuration files with a thousand of devices in 3.5 seconds
  - Dot2net reduces the increase in config file size to less than 10%

Object classes in dot2net

Name	Parent	Instance	Labels	Template
NodeClass	-	Node	✓	✓
InterfaceClass	-	Interface	✓	✓
ConnectionClass	-	Interface	✓	✓
GroupClass	-	Group	✓	
NeighborClass	InterfaceClass	Neighbor		✓
MemberClass	any[1]	Member		✓

Example of automated IP address assignment

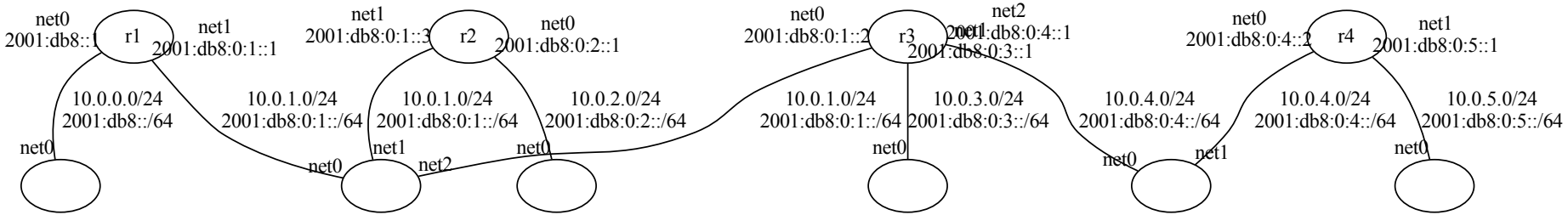
Input network topology (ospf_topo1)



⇦ Four routers

⇦ Six switches

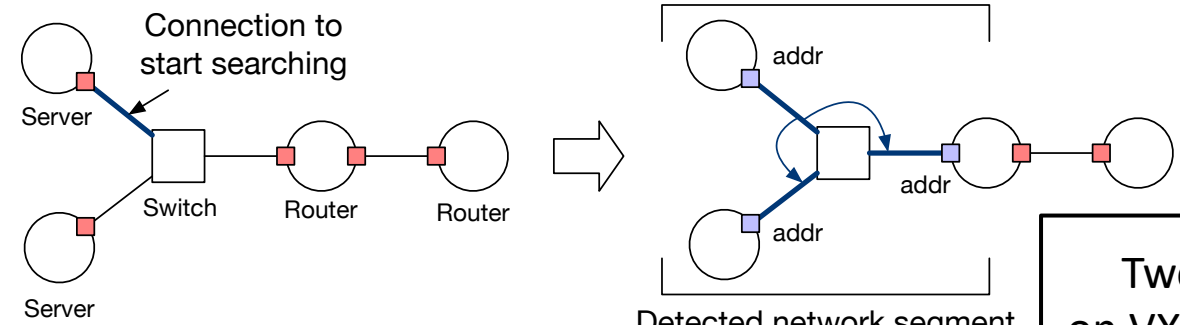
IP address assignment (IPv4/IPv6 dualstack)



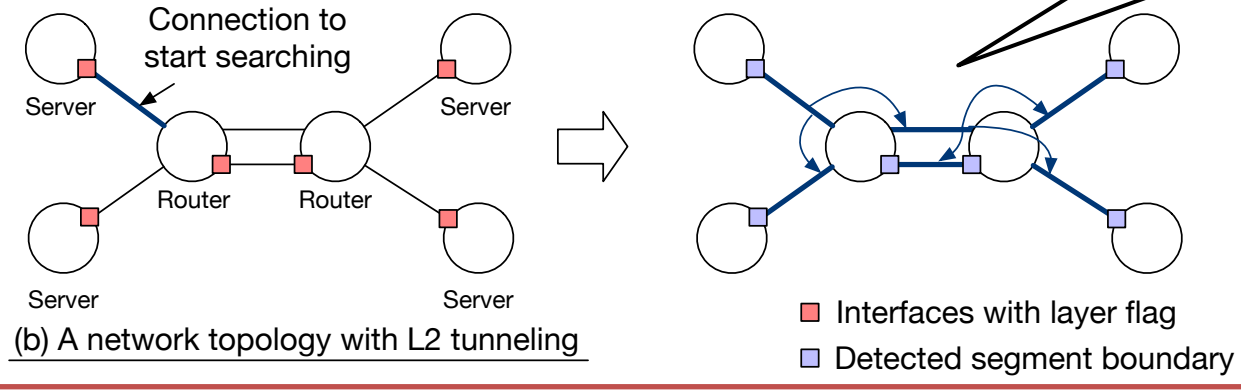
Expressiveness – IP addressing issue

- IP dual stack
  - “ospf_topo1” scenario is IPv4/IPv6 dualstack
  - Works well, shown as an example
- VXLAN
  - “bgp_evpn_vxlan_topo1” scenario depends on VXLAN
  - Separate IP layer and IP-over-VXLAN layer

Automated IP addressing for VXLAN network



Two logical links
on VXLAN connection



Expressiveness – Namespace issue

- BGP Neighbor
  - "bgp_features" scenario includes both BGP and non-BGP routers/switches
    - Require parameters of L3 neighbor devices which can be more than one
  - Define Neighbor subclass that belong to Interface classes
    - The subclass will generate as many config blocks as the L3-neighboring objects
- Stub network
  - "rip_topo1" and "ospf6_topo1" scenarios has mixed routing policies of dynamic routing and static routing (stub)
  - Put virtual nodes on the topology
    - No configuration but available parameters of the virtual nodes in namespace

Comparison with existing methods

Method	Approach	Description simplicity	Scalability	Expressiveness
PRESTO [7]	Extended template blocks	**Complicated due to extended control syntax**	Scalable	VPN, VoIP, etc.
AutoNetkit [5], [20]	Scripted model + templates	**Complicated due to procedural style and control syntax**	Limited support of parameter assignment	**Limited to BGP and OSPF**
Propane [21]	Product graph + domain specific language	Sophisticated but not intuitive	Scalable	**Limited to BGP**
HolistIX [22]	Topology diagram	Simple and intuitive	**Requires manual parameters**	**Only for IX network**
dot2net	Labeled graph + template blocks	Simple and intuitive	Easily scalable as § V-B, § V-C	Expressive as § V-D

- Dot2net satisfies all of description simplicity, scalability, expressiveness as explained
- Existing approaches have limitation mainly on expressiveness (supporting limited protocols or environment)