Efficient Network Reachability Analysis Using a Succinct Control Plane Representation

Seyed K. Fayaz, Tushar Sharma, Ari Fogel, Ratul Mahajan, Todd Millsteinz, Vyas Sekar and George Varghese

Presented By: Sandeep Kumar, Pawan Rajotiya

Network Reachability

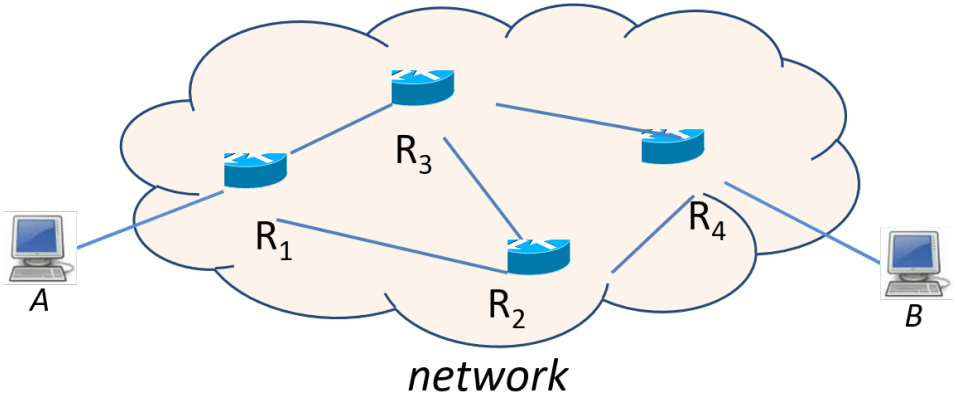


Figure 1: General Network

- Given a network, can node A *always* reach node B ?

Network Reachability

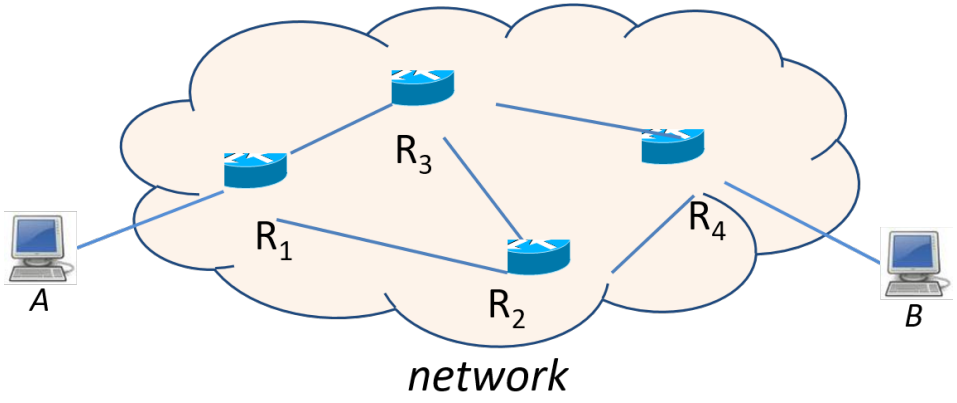


Figure 1: General Network

- Given a network, can node A *always* reach node B ?
  - Always is the key term here.

Network Reachability

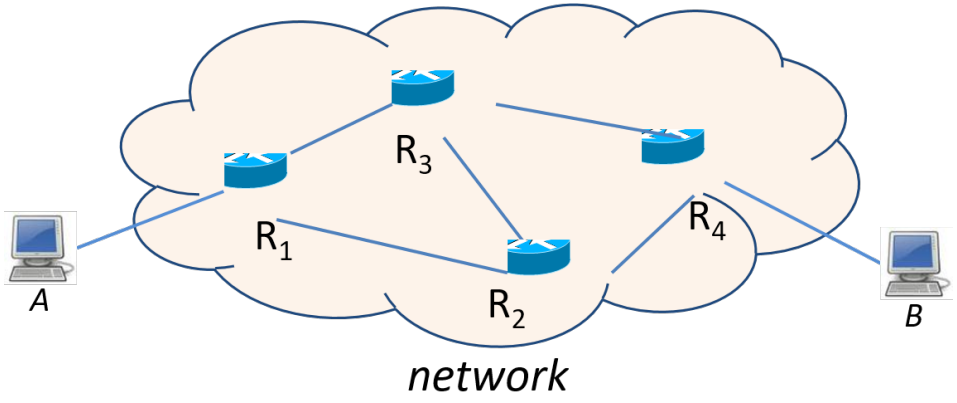


Figure 1: General Network

- Given a network, can node$A$ *always* reach node$B$?
  - Always is the key term here.
  - Network's routing information keeps changing.

Network Reachability

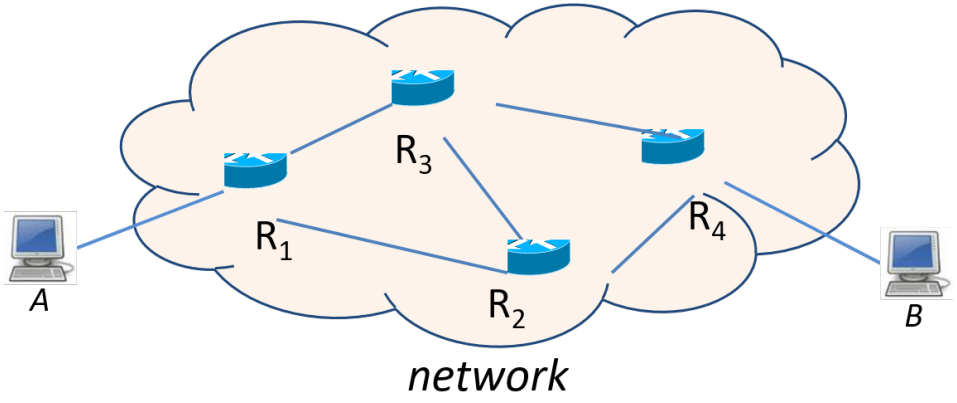


Figure 1: General Network

- Given a network, can node$A$ *always* reach node$B$?
  - Always is the key term here.
  - Network's routing information keeps changing.
  - Difficult, even Intractable to answer for a large cluster.

Network Aspects: Components

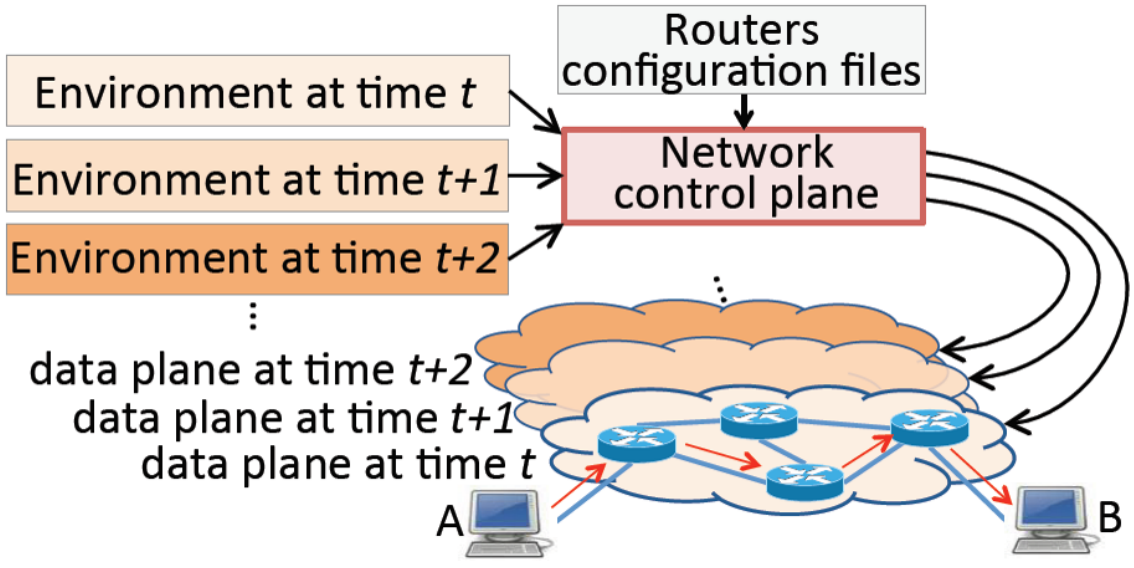


Figure 2: Aspects of a Network

- Control Plane
  - Configuration of Routers.
  - Does NOT changes with time.

Network Aspects: Components

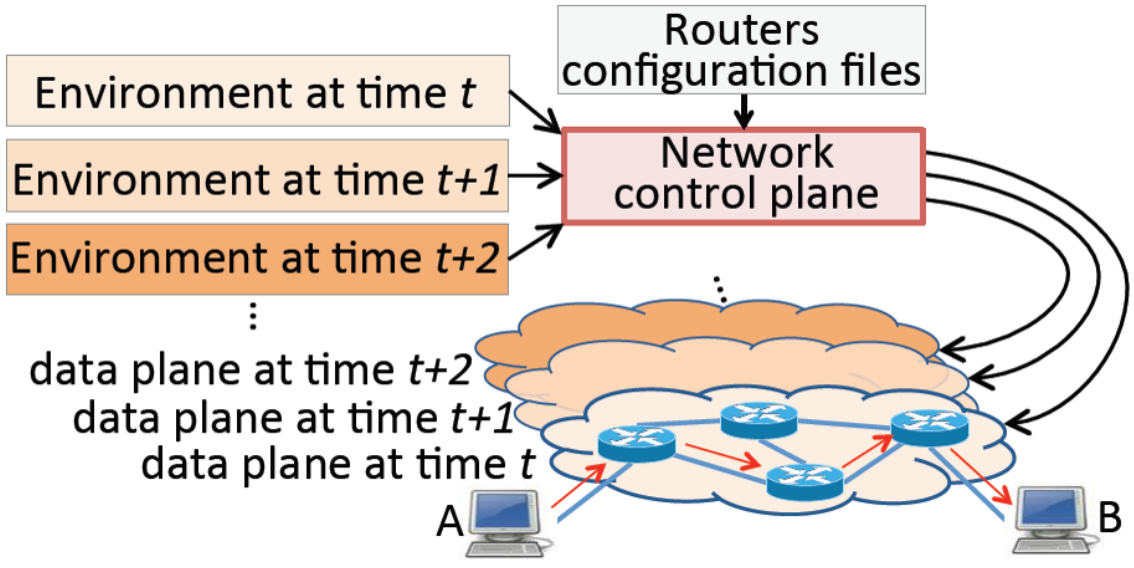


Figure 2: Aspects of a Network

- Data Plane
  - Routing information.
  - Keeps changing with time.

Network Aspects: Components

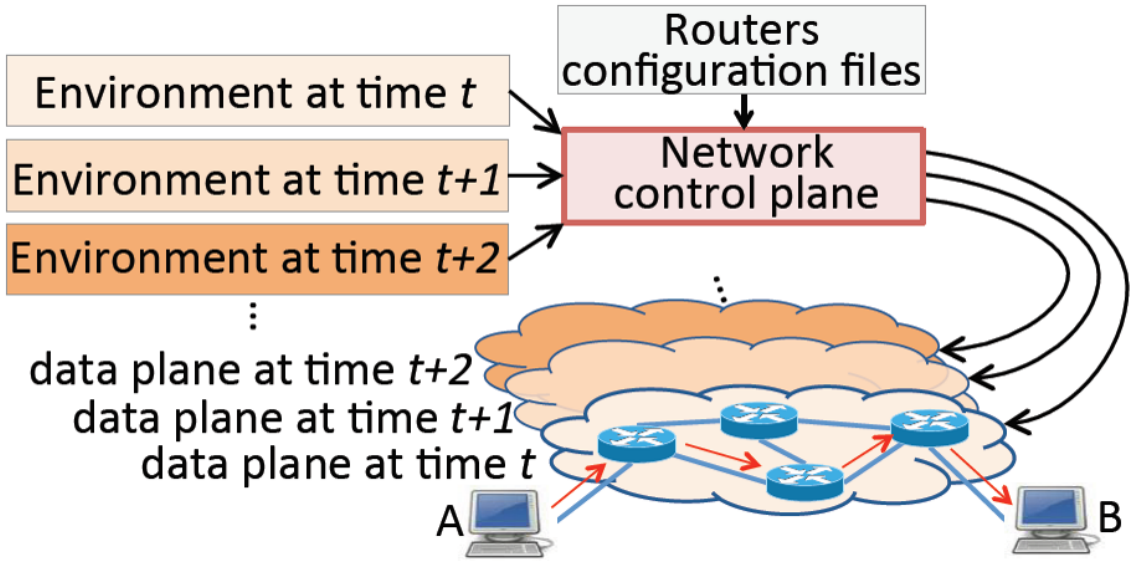


Figure 2: Aspects of a Network

- Environment
  - Everything outside the network considerations
  - Keeps changing with time.

Data Plane

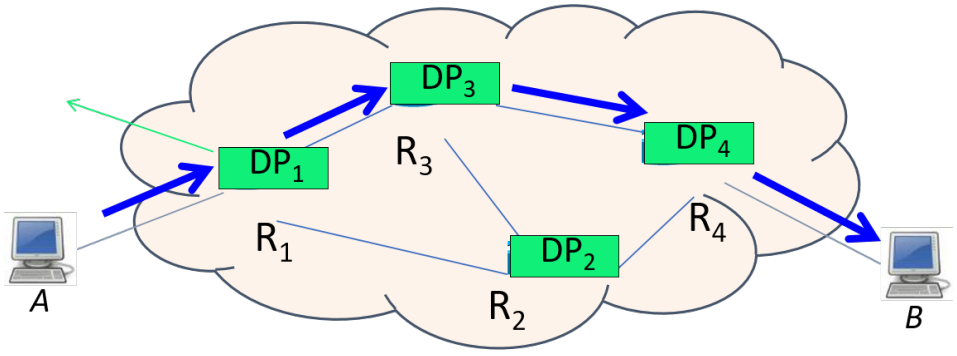


Figure 3: Network Showing Data planes state

- Data Plane consists of the routing information of the network.
- Forwarding/Routing Table in the router.
- Updated as per the announcements received.

Data plane analysis

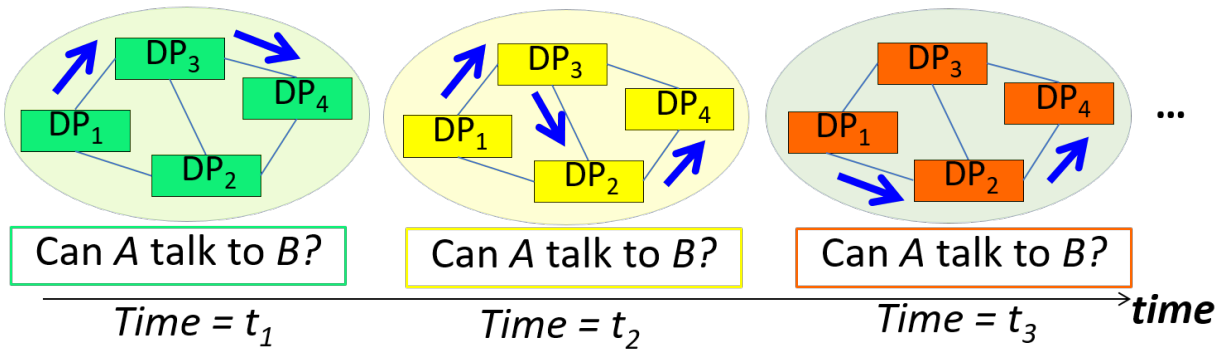
Focus of recent works is on the Data plane analysis.

Data plane analysis

Focus of recent works is on the Data plane analysis.

Problems:

- Data Plane keeps changing.

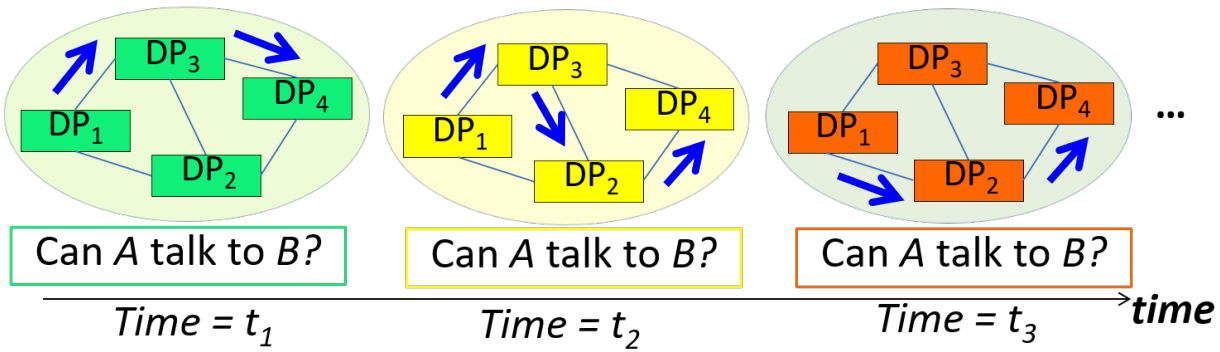


Data plane analysis

Focus of recent works is on the Data plane analysis.

Problems:

- Data Plane keeps changing.



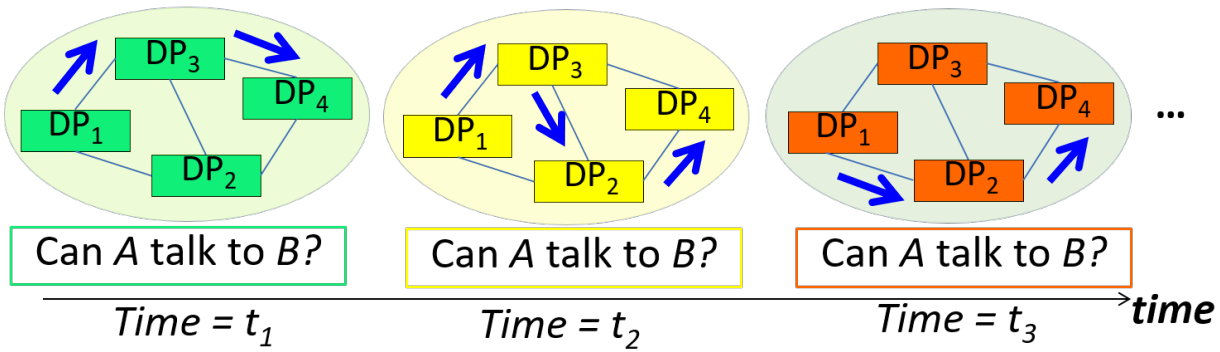
- Need to verify each of them to satisfy the *always* constraint.

Data plane analysis

Focus of recent works is on the Data plane analysis.

Problems:

- Data Plane keeps changing.



- Need to verify each of them to satisfy the *always* constraint.
- Difficult, even intractable for large networks.

Maintenance Triggered Bugs

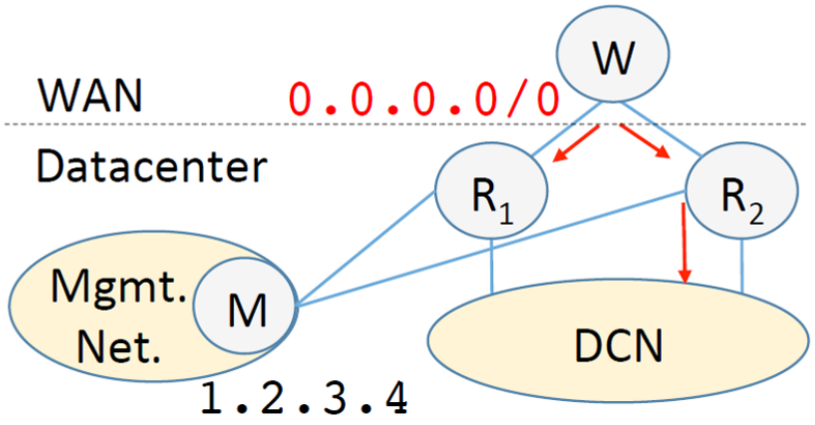


Figure 4: Problem due to Static Routing

Setup

DCN is connected to WAN via two routers, R_1 and R_2. There is also a management router M present connected to both the routers.

Maintenance Triggered Bugs

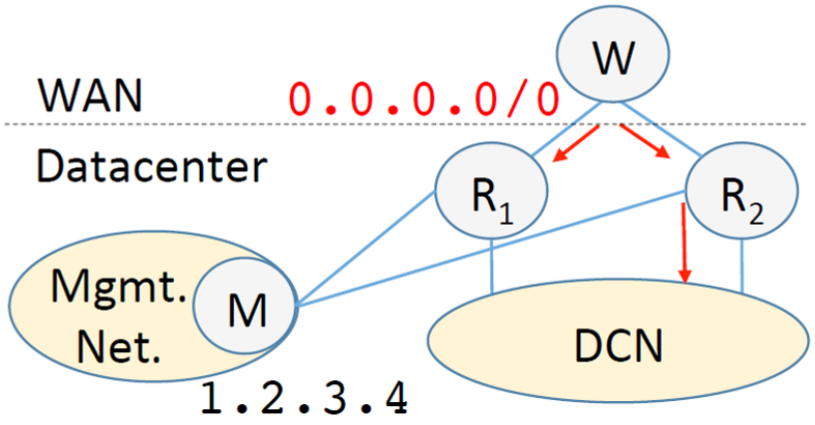


Figure 4: Problem due to Static Routing

Problem

As the Router R_2 is taken offline, DCN losses connectivity.

Maintenance Triggered Bugs

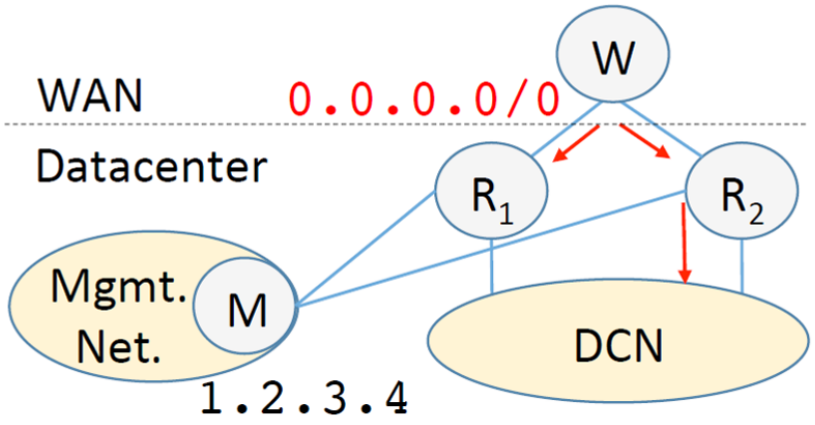


Figure 4: Problem due to Static Routing

Reason

Because of the static route, 0.0.0.0/0, R_1 forwards all the traffic to M. M sends it to DCN using R_2. This makes R_2 essential.

Maintenance Triggered Bugs

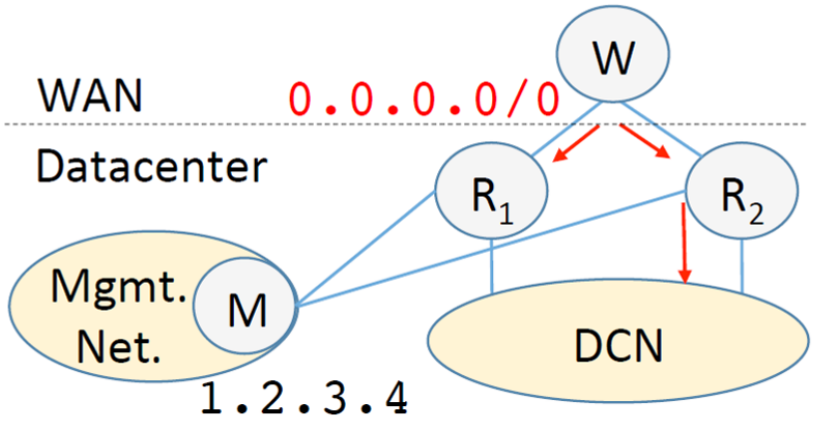


Figure 4: Problem due to Static Routing

Reason

Because of the static route, 0.0.0.0/0, R_1 forwards all the traffic to M. M sends it to DCN using R_2. This makes R_2 essential.

Static Route

Static routes defined by an *admin*. It has the highest priority and is not announced to nearby routers.

Failure-Triggered Bugs

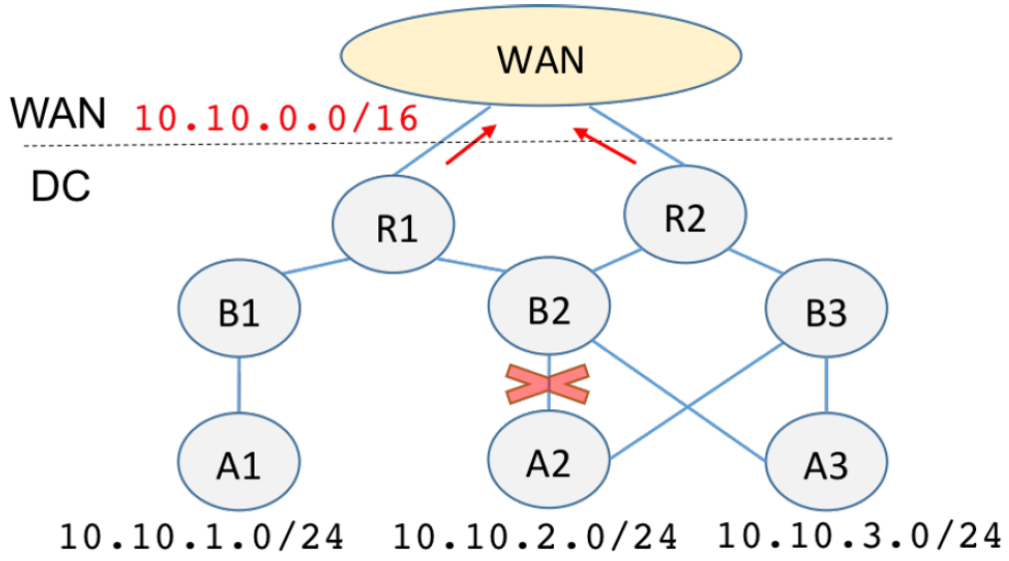


Figure 5: Problem to Route Aggregation

Setup

Routers belong to different *levels*. R, B and A are different levels.

Route Aggregation

A1:10.10.1.0/24

A2:10.10.2.0/24

A3:10.10.3.0/24

Change to Binary:

A1: 10.10.00000001.0/24

A2: 10.10.00000010.0/24

A3: 10.10.00000011.0/24

A : 10.10.00000000.0/22

Failure-Triggered Bugs

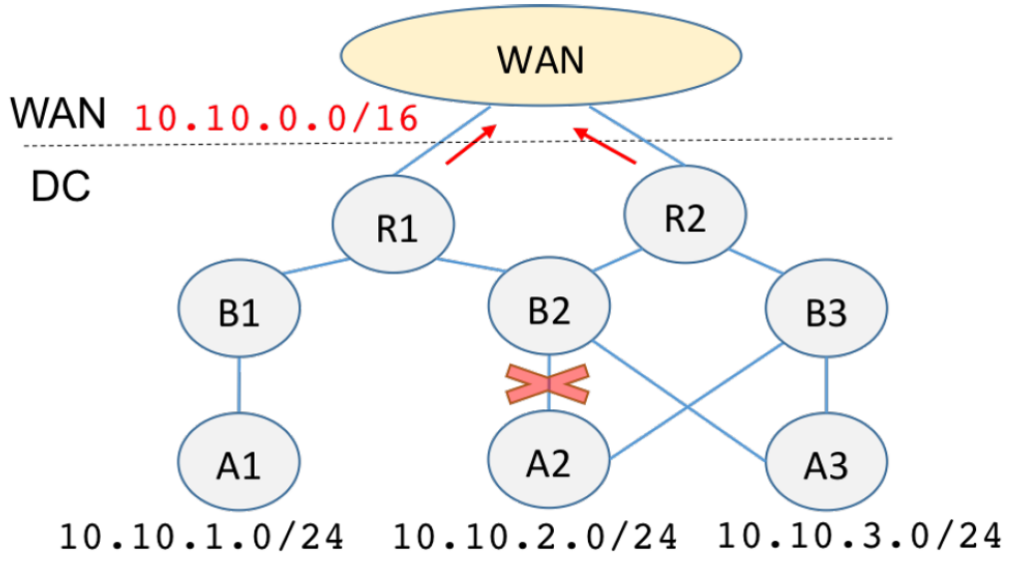


Figure 5: Problem to Route Aggregation

Problem

After the link between B_2 and A_2 goes down, some packets sent from *WAN* to A_2 are lost. Even though it is connected to *WAN* via R_2

Failure-Triggered Bugs

Route Aggregation

A1:10.10.1.0/24
A2:10.10.2.0/24
A3:10.10.3.0/24

Change to Binary:
A1: 10.10.00000001.0/24
A2: 10.10.00000010.0/24
A3: 10.10.00000011.0/24

A : 10.10.00000000.0/22

A1:10.10.1.0/24

A3:10.10.3.0/24

Change to Binary:
A1: 10.10.00000001.0/24

A3: 10.10.00000011.0/24

A : 10.10.00000000.0/22
Same as Before

Failure-Triggered Bugs

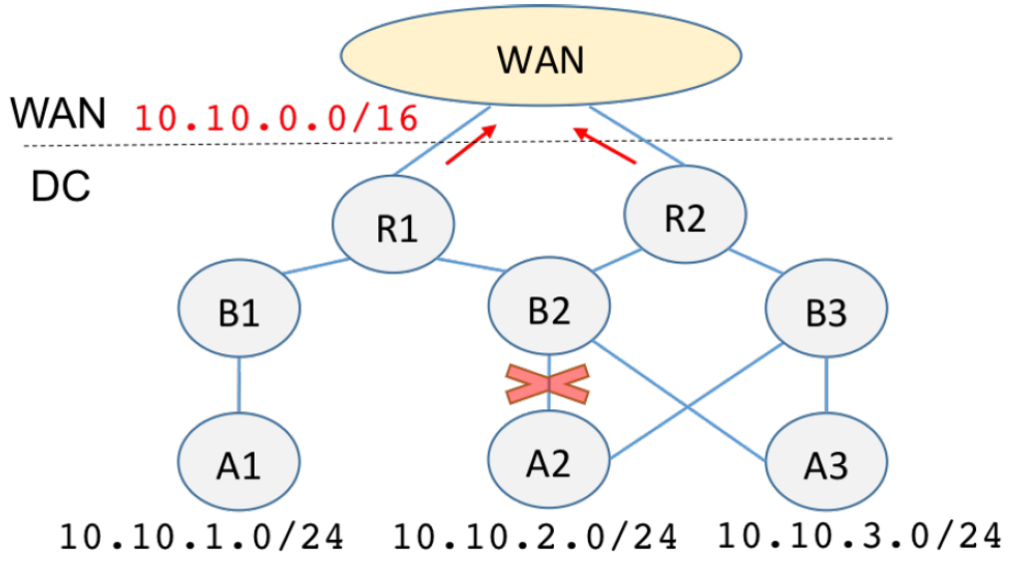


Figure 5: Problem to Route Aggregation

Problem

After the link between B_2 and A_2 goes down, some packets sent from *WAN* to A_2 are lost. Even though it is connected to *WAN* via R_2

Announcement-Triggered Bugs

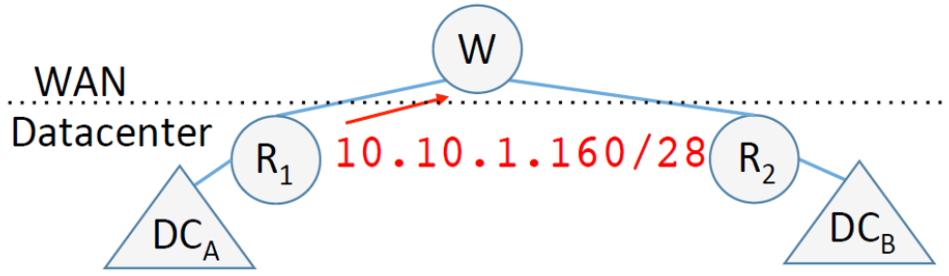


Figure 6: Problem to BGP Announcement

Setup

DC_A hosts some service on 10.10.0.0/16 prefix and DC_B can access them using the *WAN*.

Announcement-Triggered Bugs

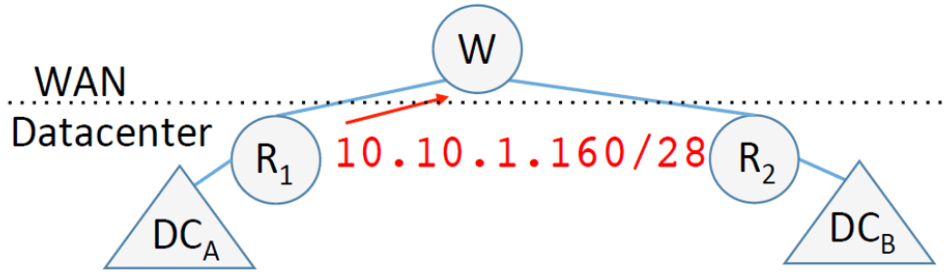


Figure 6: Problem to BGP Announcement

Problem

Some services are started in DC_A on 10.10.1.160/28 prefix, because of services hosted earlier on 10.10.0.0/16 become unreachable.

Announcement-Triggered Bugs

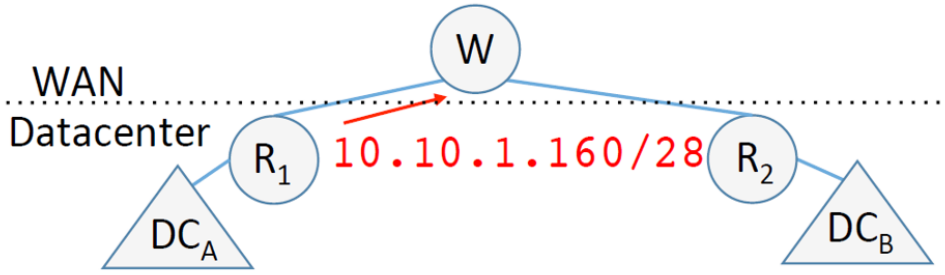


Figure 6: Problem to BGP Announcement

Reason

1. R_1 did not filter out 10.10.1.160/28 from its announcement.

Announcement-Triggered Bugs

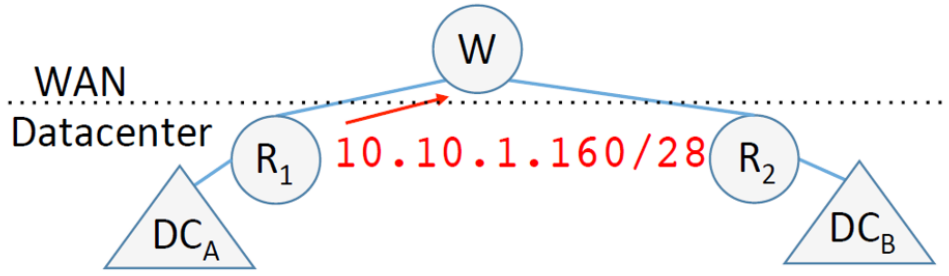


Figure 6: Problem to BGP Announcement

Reason

1. R_1 did not filter out 10.10.1.160/28 from its announcement.
2. Aggregate route on R_2 for 10.10.0.0/16 with next hop as DC_B

Announcement-Triggered Bugs

Aggregate Routes

"An aggregate route becomes active when it has one or more contributing routes. A contributing route is an active route that is a *more specific* match for the aggregate destination. For example, for the aggregate destination 128.100.0.0/16, routes to 128.100.192.0/19 and 128.100.67.0/24 are contributing routes, but routes to 128.0.0.0./8 and 128.0.0.0/16 are not."

- www.juniper.net

Announcement-Triggered Bugs

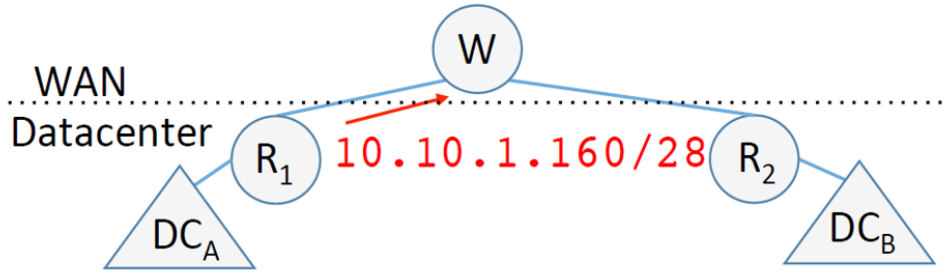


Figure 6: Problem to BGP Announcement

Reason

1. R_1 did not filter out 10.10.1.160/28 from its announcement.
2. Aggregate route on R_2 for 10.10.0.0/16 with next hop as DC_B

Control Plane

1. Control Plane analysis is preferable over Data Plane analysis.
2. Provides more powerful analysis, as data planes are generated by control plane.
3. It does not change much !!!

Control Plane

1. Control Plane analysis is preferable over Data Plane analysis.
2. Provides more powerful analysis, as data planes are generated by control plane.
3. It does not change much !!!

Clean-State Control plane design

This approach aim to build, correct-by-design control plane.
Useful in long run, but not in checking *reachability.*

ERA (Effective Reachability Analysis) Overview

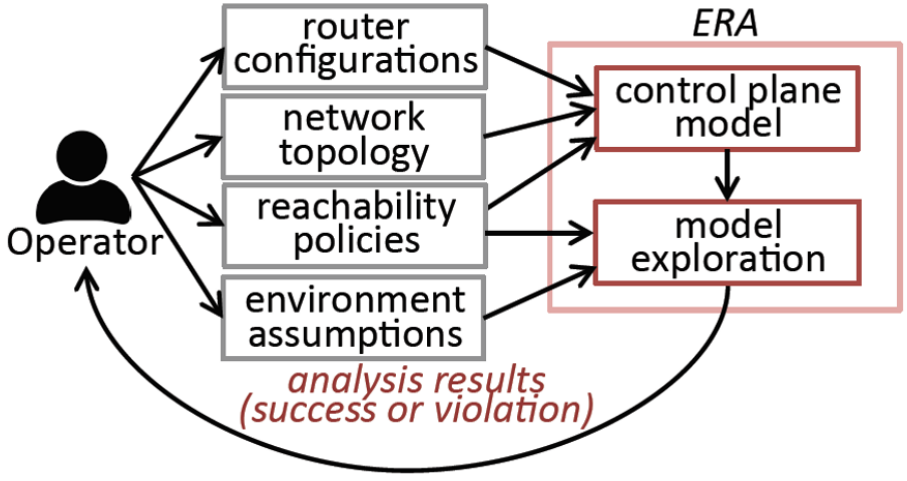


Figure 7: ERA Overview

1. Environment: Datacenter, enterprise, ISP.
2. Network Admin defines, router config, topology, reachability policy and environment assumptions.
3. ERA returns whether those policies hold or not.

Note: Dependent on Environment assumptions!!

ERA: Approach

Relationship between Control Plane and the Data Plane:

1. Data Plane definition: Receive input on one of its port and then produce the output on its own or neighbours port:

$$DP : (pkt, port) \rightarrow (pkt, port)$$

ERA: Approach

Relationship between Control Plane and the Data Plane:

1. Data Plane definition: Receive input on one of its port and then produce the output on its own or neighbours port:

$$DP : (pkt, port) \rightarrow (pkt, port)$$

2. Reachability Policy, modelled as:

$$\phi_{A \rightarrow B}$$

$\phi_{A \rightarrow B}$ is a predicate that indicates whether a packet from router port A should be able to reach router port B.

ERA: Approach

Relationship between Control Plane and the Data Plane:

1. Data Plane definition: Receive input on one of its port and then produce the output on its own or neighbours port:

$$DP : (pkt, port) \rightarrow (pkt, port)$$

2. Reachability Policy, modelled as:

$$\phi_{A \rightarrow B}$$

$\phi_{A \rightarrow B}$ is a predicate that indicates whether a packet from router port A should be able to reach router port B.

3. A data plane DP is policy-compliant if $\phi_{A \rightarrow B}(pkt, DP)$ evaluates to true for all A-to-B packets.

ERA: Insight

1. Naive Approach
   - Generate all the data planes corresponding to an Environment and check *reachability* for all of them.
   - Back to Square One !!
2. Aim is to analyse the control plane WITHOUT generating all the data planes.

ERA: Insight

Key Insight: Analyse the control plane under an environment to determine

ERA: Insight

Key Insight: Analyse the control plane under an environment to determine

First

The *routes* that each router in the network learns via its neighbours or its configuration file.

ERA: Insight

Key Insight: Analyse the control plane under an environment to determine

First

The *routes* that each router in the network learns via its neighbours or its configuration file.

Second

The best route when multiple routes to the same prefix are learned.

ERA: Example

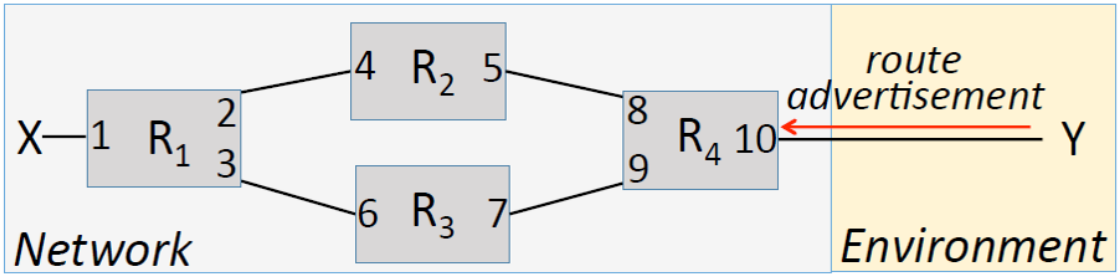


Figure 8: Reachability from X to Y

What traffic reaches from port X to port Y ?

ERA: Example

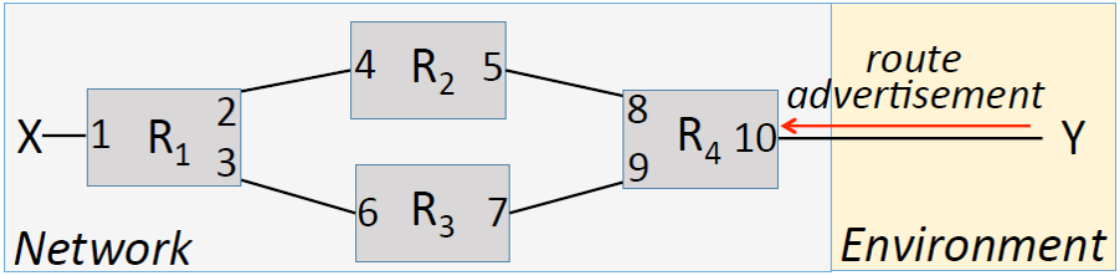


Figure 8: Reachability from X to Y

What traffic reaches from port X to port Y ?

Find the routes that traverse the opposite direction on each of the two paths.

ERA: Example

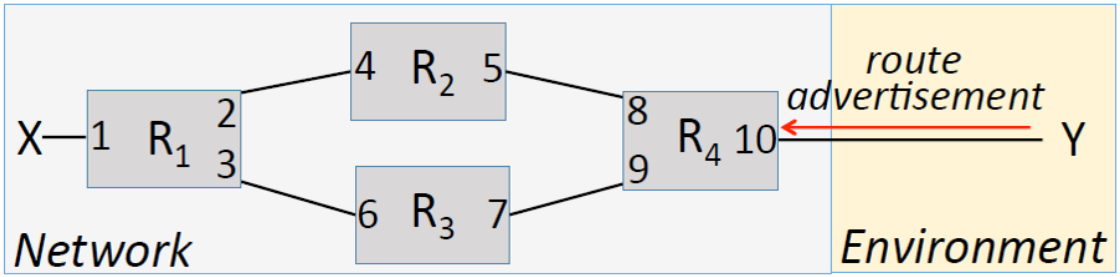


Figure 8: Reachability from X to Y

Output of router on port j, with *route* as input on port i

$$T^{i \to j}_{Router}(route)$$

ERA: Example

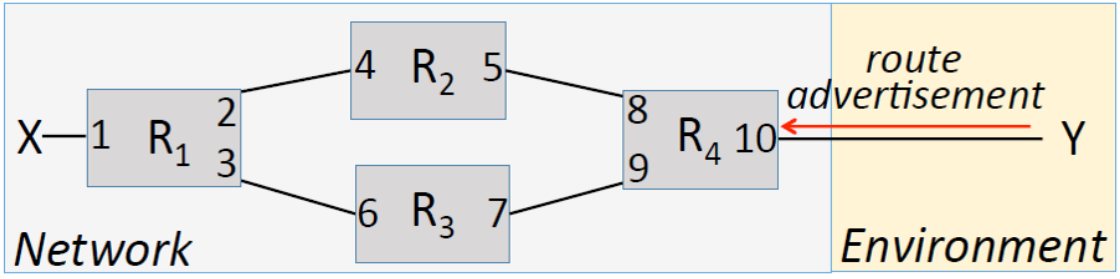


Figure 8: Reachability from X to Y

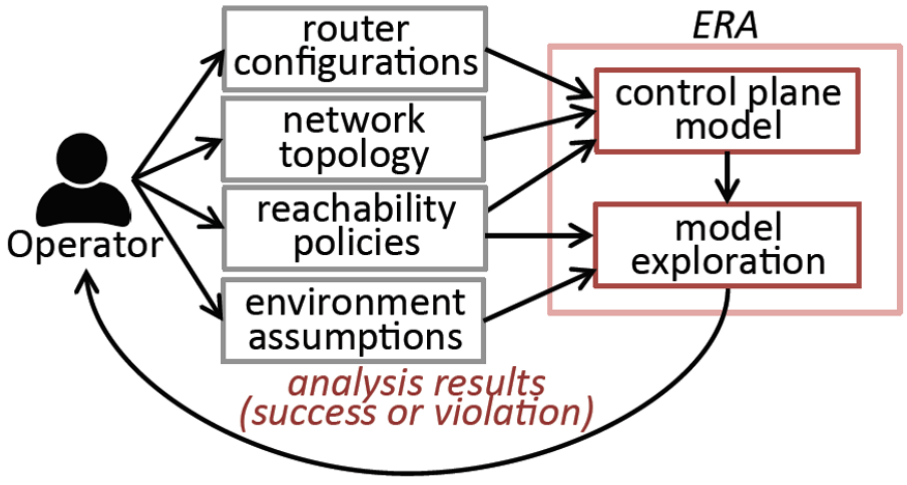
Output of router on port j, with *route* as input on port i

$$T^{i \to j}_{Router}(route)$$

Routes that reach from Y to X

$$T^{2\to1}_{R_1}(T^{5\to4}_{R_2}(T^{10\to8}_{R_4}(env))) \cup T^{3\to1}_{R_1}(T^{7\to6}_{R_3}(T^{10\to9}_{R_4}(env)))$$

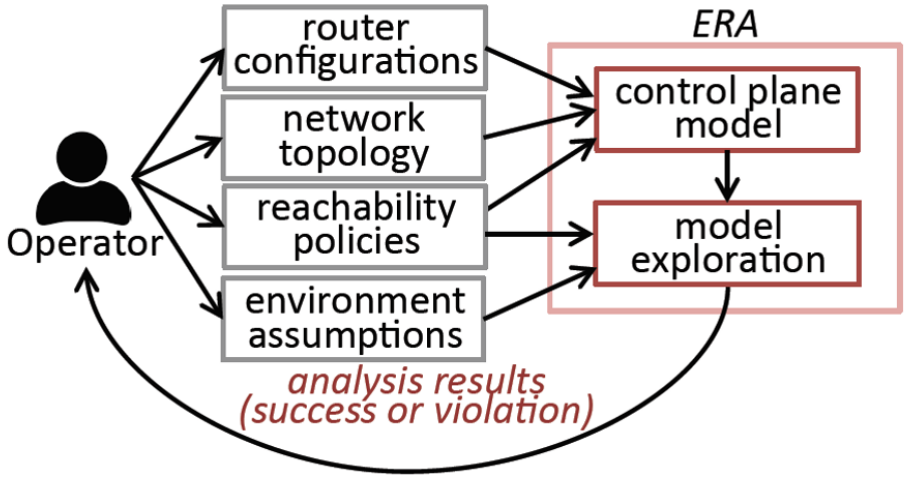
ERA: Challenges



Two main challenges for Control-based reachability:

1. An expressive and tractable control plane model.
   - Should be able to capture key behaviours of diverse protocols.
   - Should not be too general neither too specific.

ERA: Challenges



Two main challenges for Control-based reachability:

1. Scalable control plane exploration.
    - Should be easy to explore with respect to environment.

ERA : Limitations

ERA has some limitations because of its inherent design:

- User has to provide the *Environment Assumptions*.
  - This can be incorrect or overly permissive. This can lead to false positives.
  - This can lead to false negatives.
  - To guarantee freedom bugs, need to iterate over all the possibilities of environments.

ERA : Limitations

ERA has some limitations because of its inherent design:

- Cannot find all types of bugs
  - Focus on *Reachability* problem.
  - Assumes that the network converges.
  - Convergent errors and reachability in *transient* state is not guaranteed.

Convergence

The process where network sends update announcements to nearby routers. When it is finished, the network has said to be converged.

ERA : Limitations

ERA has some limitations because of its inherent design:

- It does not support certain directives such as regular expressions in routing filters.
    - However, this is a limitation of software, rather than the model of ERA.

ERA : Modeling the Control Plane

Main points to consider:

1. Capture all the routing protocols using a common abstraction.
2. Expressive with respect to routing behaviour of individuals protocol.
3. Scalable exploration.

ERA : Modeling the Control Plane

Main points to consider:

1. Capture all the routing protocols using a common abstraction.
2. Expressive with respect to routing behaviour of individuals protocol.
3. Scalable exploration.

Network Control plane is a composition of control plane of routers.
Problems to Model:

1. I/O unit of a Router.
2. Processing logic of a Router.

Modeling Control Plane I/O

Naive way: Use the actual specification of router advertisements of different protocols.

1. Include all the low level details (keep-alive messages, sequence numbers etc.).
2. This makes the model very expressive.
3. However, this makes it too cumbersome !!
4. It is not required for the *reachability* analysis,

Modeling Control Plane I/O

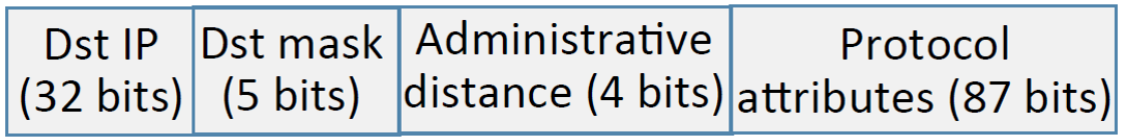
Another Naive way: Completely ignore differences across protocols to simplify our I/O unit model

1. Expressiveness is lost !!
2. May not be able to capture some key properties, like *Administrative Distance* which is a metric of priority among protocols.

Strike a balance between the expressiveness and tractability

Modeling Control Plane I/O

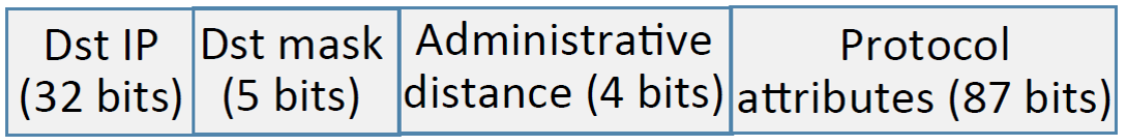
Abstract Route:



1. 128 bit vector for I/O unit for Control Plane.
2. Mimics a route advertisement.
3. Represented as Bit Vector, conveying key information in route advertisements that affects routing decisions of a router.
4. It abstracts away low level details of a route.

Modeling Control Plane I/O

Abstract Route:



1. Destination IP Address and Mask
2. Administrative Distance: Numerical Distance of the protocols. $AD_A < AD_B$ means, protocol A is preferred over B.
3. Protocol Attributes: Attributes related to protocol.

Encoded such that selecting on over other means choosing lower of the unsigned representation of $AD_1.attrs_1$ and $AD_2.attrs_2$.

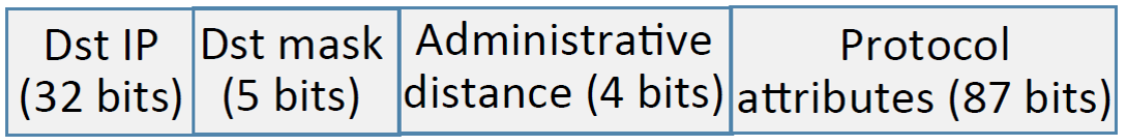
Modeling Control Plane I/O

Route Source	Default Distance Values
Connected interface	0
Static route	1
Enhanced Interior Gateway Routing Protocol (EIGRP) summary route	5
External Border Gateway Protocol (BGP)	20
Internal EIGRP	90
IGRP	100
OSPF	110
Intermediate System-to-Intermediate System (IS-IS)	115
Routing Information Protocol (RIP)	120
Exterior Gateway Protocol (EGP)	140
On Demand Routing (ODR)	160
External EIGRP	170
Internal BGP	200
Unknown*	255

Figure 9: Administrative Distance of Various Protocols

Modeling Control Plane I/O

Abstract Route:



1. Destination IP Address and Mask
2. Administrative Distance: Numerical Distance of the protocols. $AD_A < AD_B$ means, protocol A is preferred over B.
3. Protocol Attributes: Attributes related to protocol.

Encoded such that selecting on over other means choosing lower of the unsigned representation of $AD_1.attrs_1$ and $AD_2.attrs_2$.

ERA : Control Plane as Visibility Function

Processing Logic of the Control Plane (Routers):

ERA : Control Plane as Visibility Function

Processing Logic of the Control Plane (Routers):

Visibility Function

Model of the Router as set of operations, performed on *Input* (made visible to it) to get the *Output*.

ERA : Control Plane as Visibility Function

Processing Logic of the Control Plane (Routers):

Visibility Function

Model of the Router as set of operations, performed on *Input* (made visible to it) to get the *Output*.

5 Key Operations:

1. Input filtering

ERA : Control Plane as Visibility Function

Processing Logic of the Control Plane (Routers):

Visibility Function

Model of the Router as set of operations, performed on *Input* (made visible to it) to get the *Output*.

5 Key Operations:

1. Input filtering
2. Route redistribution
    - Capture cross protocol interaction

ERA : Control Plane as Visibility Function

Processing Logic of the Control Plane (Routers):

Visibility Function

Model of the Router as set of operations, performed on *Input* (made visible to it) to get the *Output*.

5 Key Operations:

1. Input filtering
2. Route redistribution
   - Capture cross protocol interaction
3. Route aggregation

ERA : Control Plane as Visibility Function

Processing Logic of the Control Plane (Routers):

Visibility Function

Model of the Router as set of operations, performed on *Input* (made visible to it) to get the *Output*.

5 Key Operations:

1. Input filtering
2. Route redistribution
   - Capture cross protocol interaction
3. Route aggregation
4. Route selection
   - Select the best route for a given destination prefix.

ERA : Control Plane as Visibility Function

Processing Logic of the Control Plane (Routers):

Visibility Function

Model of the Router as set of operations, performed on *Input* (made visible to it) to get the *Output*.

5 Key Operations:

1. Input filtering
2. Route redistribution
   - Capture cross protocol interaction
3. Route aggregation
4. Route selection
   - Select the best route for a given destination prefix.
5. Output Filtering

Visibility Function

V^{in}: set of input *routes* received from its neighbours and configured static routes.

$$V^{out}_{Router} = T_{Router}(V^{in}_{Router})$$

denotes the control plane visibility function of *Router.*

V^{out}: set of output routes sent by a particular *Router*.

Visibility Function: Operations

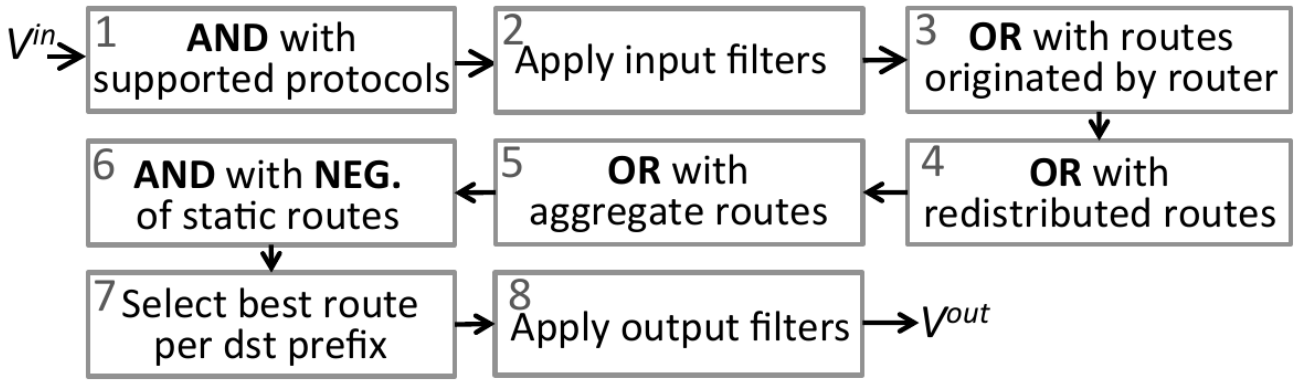


Figure 9: Visibility Function

Visibility Function: Route Encoding

Visibility Functions:

1. Processing on **one** *route* at a time is not scalable.
2. Works on a set of *routes*
3. Use BDD: Binary Decision Diagram to represent the encodings
4. BDD allows *fast exploration*

BDD: Binary Decision Diagram

$$f(x_1, x_2, x_3) = \bar{x_1}\bar{x_2}\bar{x_3} + x_1x_2 + x_2x_3$$

BDD: Binary Decision Diagram

$$f(x_1, x_2, x_3) = \bar{x_1}\bar{x_2}\bar{x_3} + x_1x_2 + x_2x_3$$

x_1	x_2	x_3	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Table 1: Truth Table

BDD: Binary Decision Diagram

$$f(x_1, x_2, x_3) = \bar{x_1}\bar{x_2}\bar{x_3} + x_1x_2 + x_2x_3$$

x_1	x_2	x_3	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Table 1: Truth Table

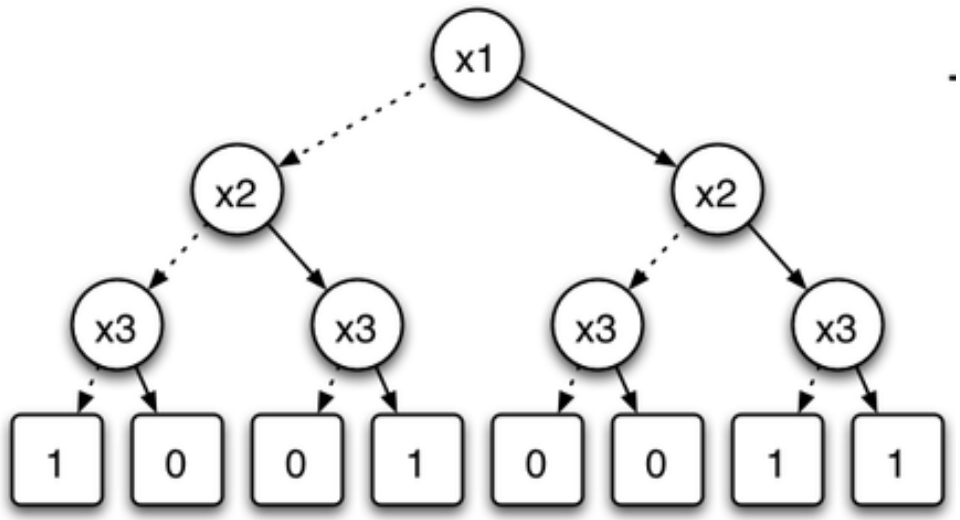


Figure 10: Binary Decision Tree

BDD: Binary Decision Diagram

$$f(x_1, x_2, x_3) = \bar{x_1}\bar{x_2}\bar{x_3} + x_1x_2 + x_2x_3$$

x_1	x_2	x_3	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Table 1: Truth Table

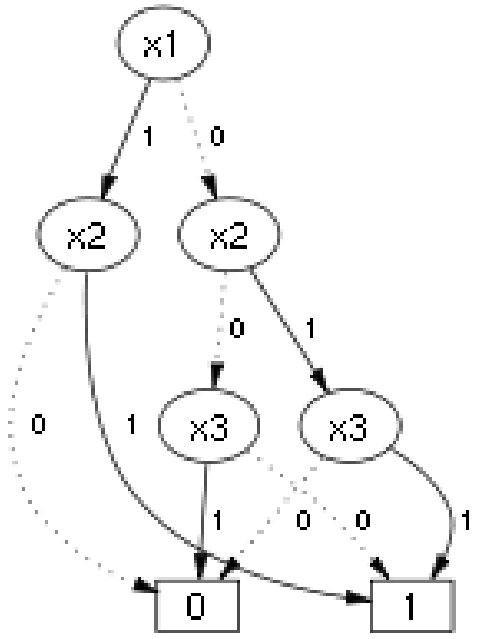


Figure 10: Binary Decision Diagram

Example

Setup:

1. For simplicity, route has 4 bits, $x_3x_2x_1x_0$
   - x_3x_2 represents the IP and the prefix.
   - x_1 for Administrative Distance (AD).
   - x_0 for protocol attribute.
2. Assume environment sends 1 (true) , which captures all the possibilities from the environment.
3. Router configured with static route and *RIP* with AD values 0 and 1 respectively.The static route is 10/2.

Example

Run:

- *RIP*: Presence of RIP on Router:
  $1 \wedge x_1$

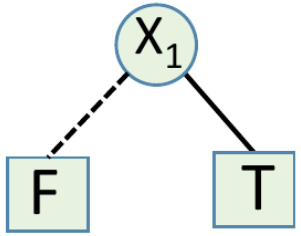


Figure 11: RIP BDD

Example

Run:

- *RIP*: Presence of RIP on Router: $1 \wedge x_1$
- *Static 10/2*: This will override routes with same prefix: $(\overline{x_3 \overline{x_2}})x_1 = \overline{x_3}x_1 \vee x_2 x_1$

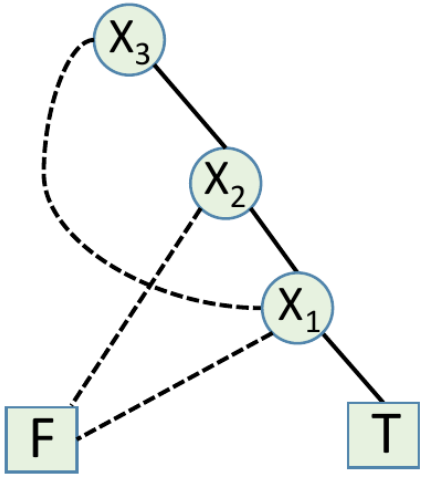


Figure 11: Static Route BDD

Example

Run:

- *RIP*: Presence of RIP on Router: $1 \wedge x_1$
- *Static 10/2*: This will override routes with same prefix: $(\overline{x_3 \overline{x_2}})x_1 = \overline{x_3}x_1 \vee x_2 x_1$
- *Output Filter*: If RIP is 0 make it 1. This replaces all the x_1 with $x_1 x_0$:
  $V^{out} = \overline{x_3}x_1 x_0 \vee x_2 x_1 x_0$
  $= (\overline{x_3} \vee x_2) \wedge x_1 x_0$

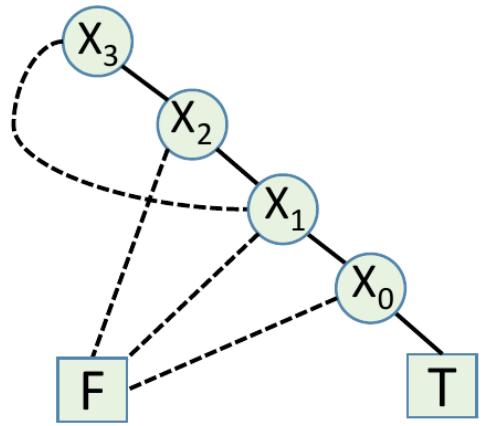


Figure 11: Output Filter BDD

Example

Run:

- *RIP*: Presence of RIP on Router: $1 \wedge x_1$
- *Static 10/2*: This will override routes with same prefix: $(\overline{x_3 \overline{x_2}})x_1 = \overline{x_3}x_1 \vee x_2 x_1$
- *Output Filter*: If RIP is 0 make it 1. This replaces all the x_1 with $x_1 x_0$:
  $V^{out} = \overline{x_3}x_1 x_0 \vee x_2 x_1 x_0$
  $= (\overline{x_3} \vee x_2) \wedge x_1 x_0$

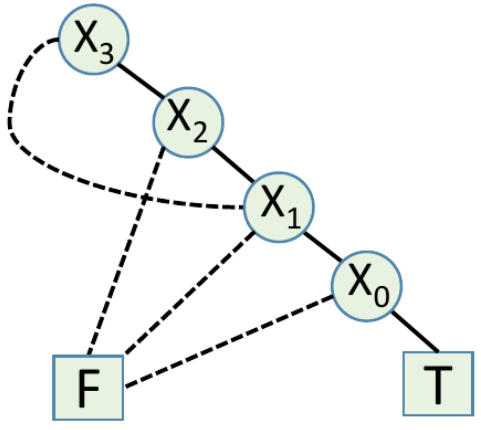


Figure 11: Output Filter BDD

Given *every* environment, Router outputs RIP with *attribute* as 1.

Destination prefix can be 00,01 or 11.

Visibility Function: Operations

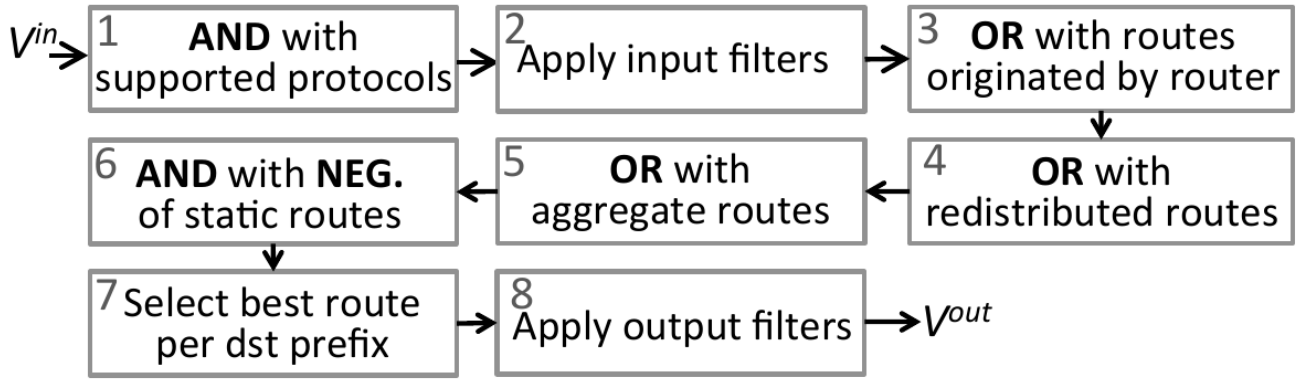


Figure 12: Visibility Function

Input

1 ▷ Inputs: (1) Configuration information pertaining to router output port $Router_{port}$ including: static routes $sr[.]$, route redistribution $rr[.]$, route aggregation $ra[.]$, supported routing protocols $proto[.]$, input filters $if[.]$, output filters $of[.]$
(2) Input to the router is a boolean function in DNF form:
$V^{in} = X_1^{in} \vee \cdots \vee X_N^{in}$

2 ▷ Output: Boolean representation of $Router_{port}$ in DNF

Visibility Function: Operations

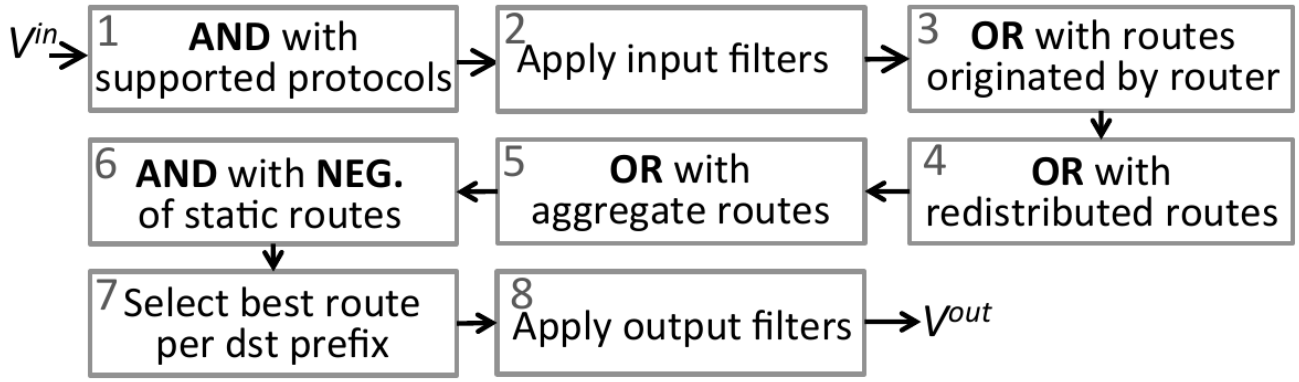


Figure 12: Visibility Function

Initializations

3 ▷ Route bit vector from Figure 7, denoted by X, is concatenation of 3 fields: $X = X_{prefix}.X_{proto}.X_{attr}$

4 ▷ We show the length of an array $array$ by $size(array[.])$

5 $V^{out} = V^{in}$ ▷ Initializing the output

Visibility Function: Operations

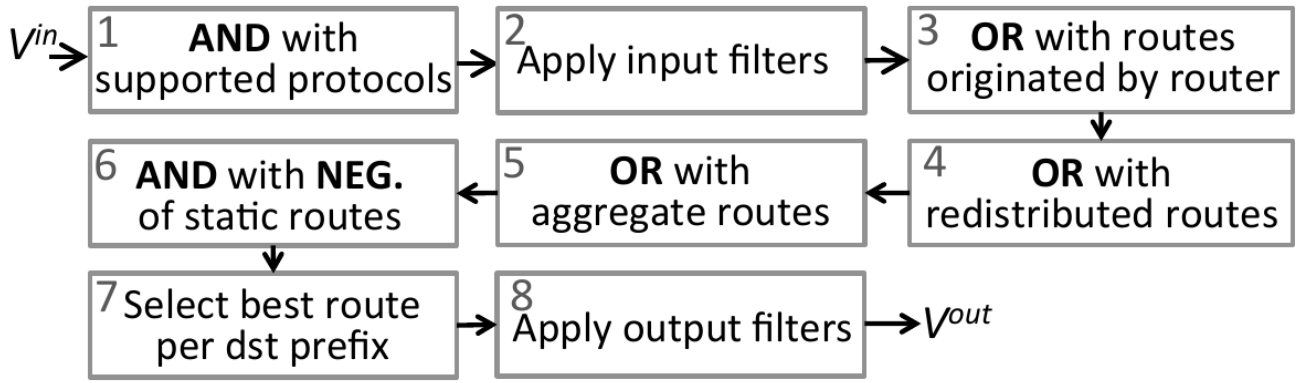


Figure 12: Visibility Function

1. AND with Supported protocols

6 ▷ Applying supported routing protocols
7 $V^{out} = V^{out} \wedge \{\bigvee_i X_{proto[i]}\}$

Visibility Function: Operations

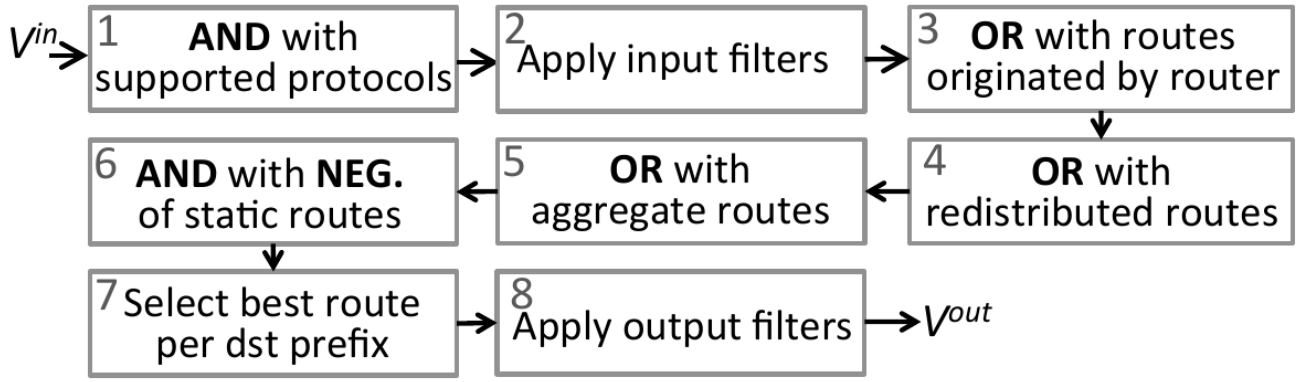


Figure 12: Visibility Function

2. Input Filters

```
 8  ▷ Applying input filters
 9  for i = 1 to size(if[.])
10     for each disjunctive term of V^out, denoted by V_j^out
11        if V_j^out matches if[i].condition
12           apply action if[i].action
```

Visibility Function: Operations

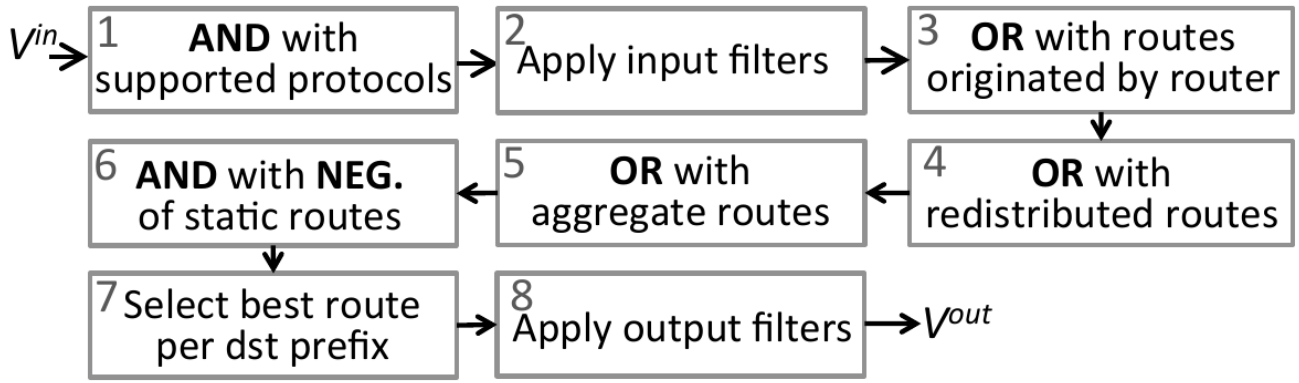


Figure 12: Visibility Function

3. OR with Originated Routes

13 ▷ Accounting for routes that $Router$ originates, denoted by V^{local}
14 $V^{out} = V^{out} \vee V^{local}$

Visibility Function: Operations

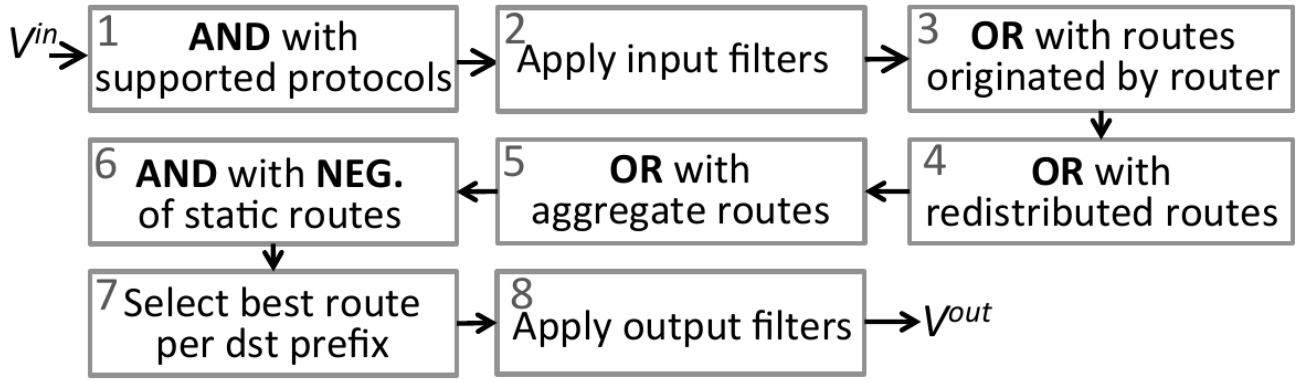


Figure 12: Visibility Function

4. OR with Redistributed Routes

15 ▷ Applying route redistribution
16 **for** $i = 1$ to $size(rr[.])$
17 **for each** disjunctive term of V^{out}, denoted by V_j^{out}
18 **if** $V_j^{out}.X_{proto} == rr[i].fromProto$
19 $newTerm = V_j^{out}$
20 $newTerm.X_{proto} = rr[i].toProto$
21 $newTerm.X_{attr} = defaultAttr[proto]$
22 $V^{out} = V^{out} \vee newTerm$

Visibility Function: Operations

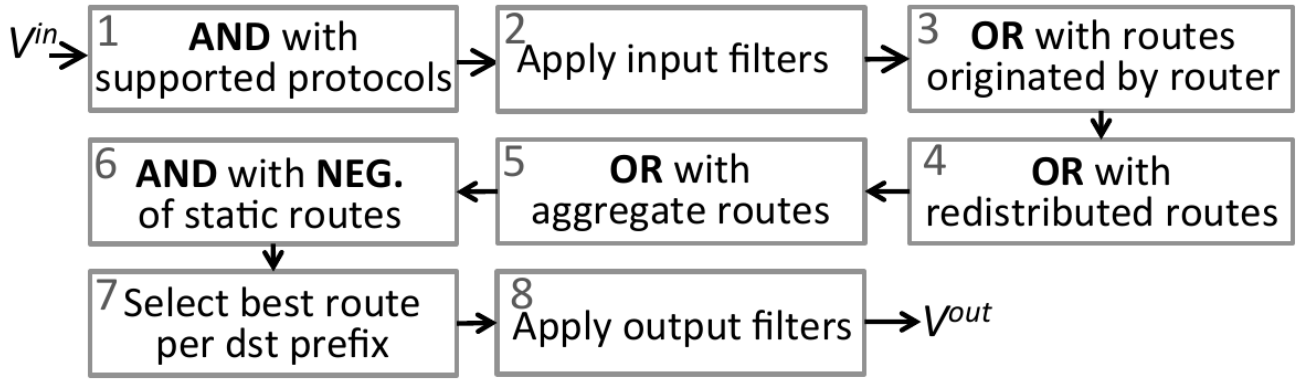


Figure 12: Visibility Function

5. OR with aggregate routes

```
23  ▷ Applying route aggregation
24  for i = 1 to size(ra[.])
25      newTerm.X_prefix = ra[i].prefix
26      newTerm.X_proto = ra[i].proto
27      newTerm.X_attr = defaultAttr[proto]
28      V^out = V^out ∨ newTerm
```

Visibility Function: Operations

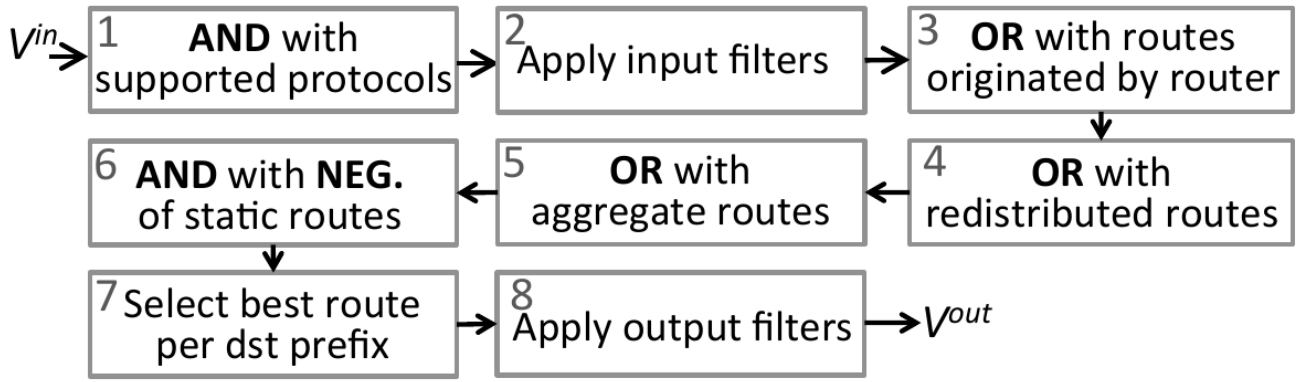


Figure 12: Visibility Function

6. AND with NEG of Static routes

29 ▷ Applying static routes
30 **for** $i = 1$ to $size(sr[.])$
31 **for each** disjunctive term of V^{out}, denoted by V_j^{out}
32 **if** $AD(V_j^{out}.X_{proto}) > AD(static)$
33 $V_j^{out} = V_j^{out} \wedge (\overline{sr[i].prefix})$

Visibility Function: Operations

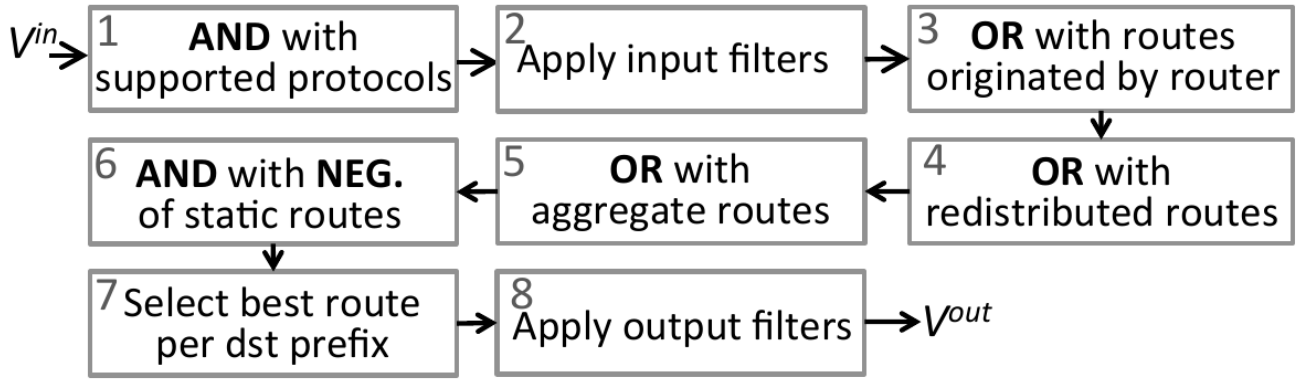


Figure 12: Visibility Function

7. Route Selection

34 ▷ Applying route selection
35 **for each** prefix $prfx$ present in V^{out}
36 $precedence = +\infty$
37 **for each** disjunctive term of V^{out}, denoted by V_j^{out}
38 **if** $(V_j^{out}.prefix == prfx)\&\&(V_j^{out}.AD.attr < precedence)$
39 $precedence = V_j^{out}.AD.attr$ ▷ Finding best route
40 **for each** disjunctive term of V^{out}, denoted by V_j^{out}
41 **if** $(V_j^{out}.prefix == prfx)\&\&(V_j^{out}.AD.attr > precedence)$
42 Eliminate V_j^{out} from V^{out} ▷ Eliminating others
43 $V_j^{out} = V_j^{out} \vee prfx.precedence$

Visibility Function: Operations

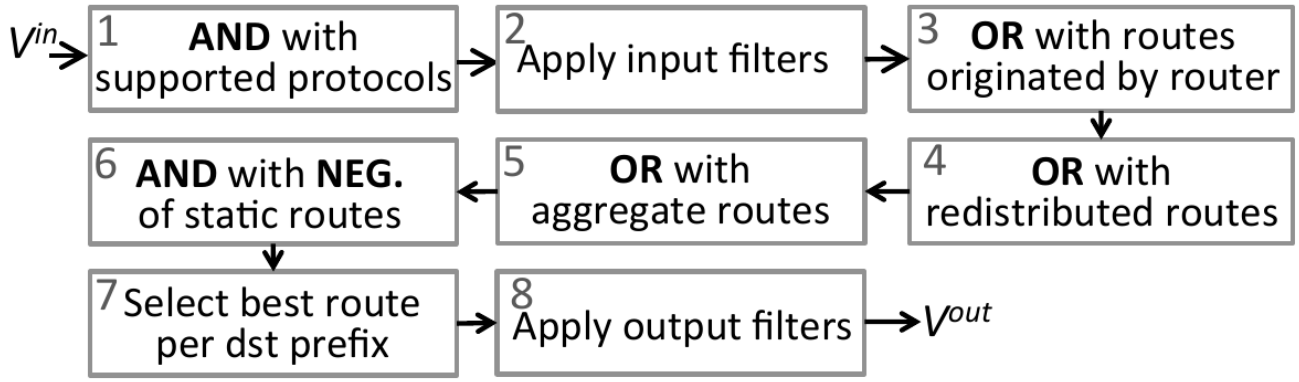
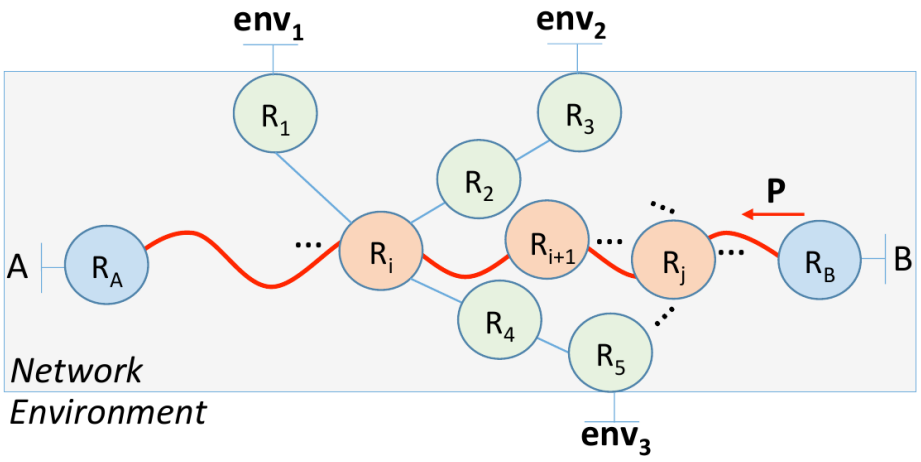


Figure 12: Visibility Function

8. Output Filters

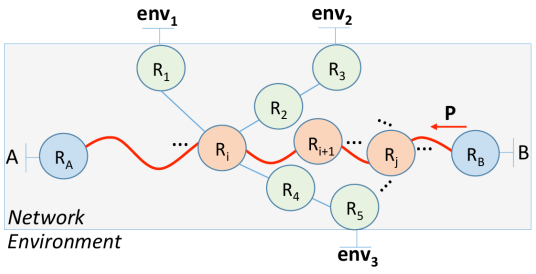
```
44  ▷ Applying output filters
45  for i = 1 to size(of[.])
46     for each disjunctive term of V^out, denoted by V_j^out
47        if V_j^out matches of[i].condition
48           apply action of[i].action

49  return V^out
```

Exploring the Model



- A to B path involves routers: $R_A, ..., R_i, R_{i+1}, ..., R_B$.
- Assuming only one path from A to B.
- Routers R_1, R_3 *and* R_5 connected to environment.
- Default value of $env = true$, unless administrator specifies one.

Exploring the Model

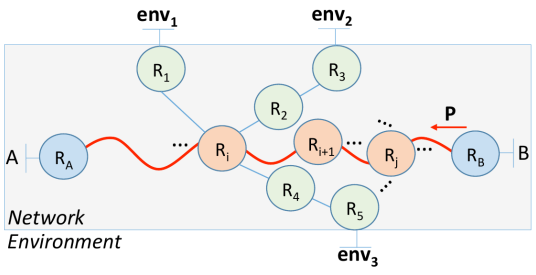


1 ▷ Inputs: (1) router-level topology of network
(2) Set of router ports facing environment Env
(3) routers configurations
2 ▷ Output: Prefix(es) of traffic reaching from router port A to router port B

3 Parse router configurations into boolean functions (using Figure 18)
4 Initialize $assumed_e$ on port e (by default, $true$)
5 initialize $assumed_B$ on port B (by default, $true$)

- A to B path involves routers: $R_A, ..., R_i, R_{i+1}, ..., R_B$.
- Assuming only one path from A to B.
- Routers R_1, R_3 *and* R_5 connected to environment.
- Default value of *env* = *true*, unless administrator specifies one.

Exploring the Model



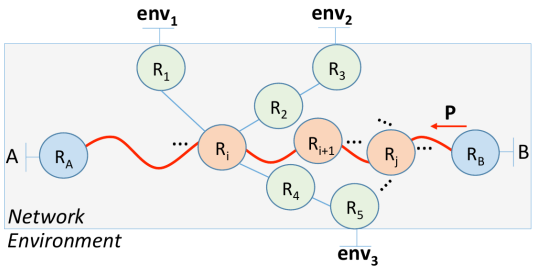
6 ▷ Accounting for effect of environment on routers on an A-to-B path
7 **for each** router $router_i$ on an $A - to - B$ path
8 **for each** environment-facing port $e \in Env$
9 **for each** path p from port e to $router_i$
10 ▷ $router_j$ is the jth router on $e \rightsquigarrow i$, where $1 \leq j \leq M(j)$
11 $E^{in}_{e \to i,p} = E^{in}_{e \to i,p} \vee T_{M(j)}(\ldots(T_1(assumed_e))\ldots)$
12 $E^{in}_{e \to i} = E^{in}_{e \to i} \vee V^{in}_{e \to i,p}$
13 $E^{in}_i = E^{in}_i \vee E^{in}_{e \to i}$

Computing traffic reachable from A to B:

- Apply the effect of environment. Router R_i receives the environment input E_i^{in}, where

$$E_i^{in} = T_1(env_1) \vee T_2(T_3(env_2)) \vee T_4(T_5(env_3))$$

Exploring the Model



14 ▷ Compute per-path reachability

15 Find all paths from B to A in G:
$Path_{B \rightsquigarrow A} = \{path^1_{B \rightsquigarrow A}, \ldots, path^N_{B \rightsquigarrow A}\}$

16 ▷ $router^j_i$ is the jth router on $path^i_{B \rightsquigarrow A}$, where $1 \leq j \leq M(j)$

17 $reachability^{path^i_{B \rightsquigarrow A}}_{B \rightsquigarrow A} =$
$T_{M(j)}(\ldots(T_2(E^{in}_2 \vee (T_1(E^{in}_1 \vee assumed_B))))$

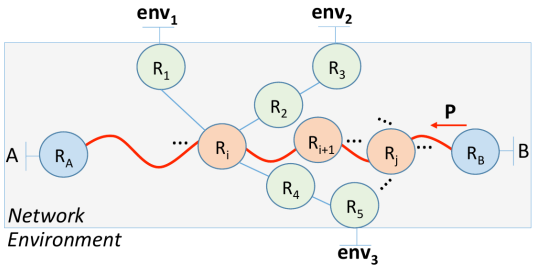
Computing traffic reachable from A to B:

- Computing routes reachable from B to A, by checking route prefixes are made visible from B to A.

  $reach_{A \rightsquigarrow B} = T_A(E^{in}_A \vee \ldots(T_{i+1}(E^{in}_{i+1} \vee \ldots T_N(E^{in}_B \vee assumed_B)\ldots)))$

  "$assumed_B$" show the input the operator assumes about what port B receives from the environment

Exploring the Model

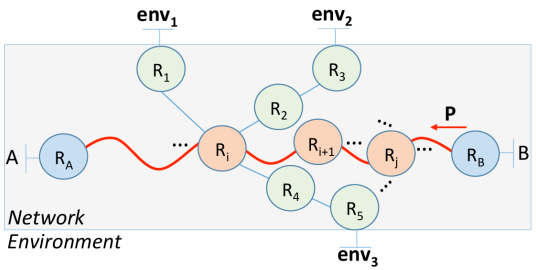


18 Eliminate binary variables in $reachability_{A \rightsquigarrow B}$ except those corresponding to X_{prefix}

Computing traffic reachable from A to B:

- Extracting prefixes reachable from A to B:
  Drop binary variables in the route fields that do not correspond to *prefix* (i.e., AD and protocol attributes) in all boolean terms of $reach_{A \rightsquigarrow B}$.

Exploring the Model



19 ▷ Accounting for static routes

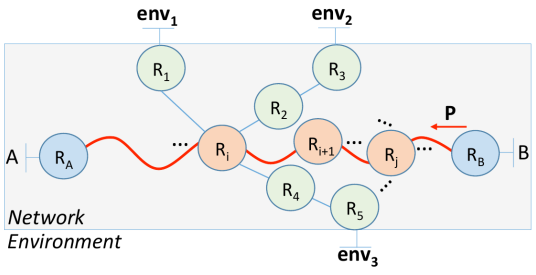
20 $static_{A \rightsquigarrow B} = \bigvee_i (\bigwedge_k (StaticPrefix_{Router_i^k}))$

21 $reachability_{A \rightsquigarrow B} = reachability_{A \rightsquigarrow B} \vee static_{A \rightsquigarrow B}$

Computing traffic reachable from A to B:

- Account for Static routes:
  Traffic from A to B can reach using *static routes* also, which are not announced by the routers. **OR** them with result from previous step.

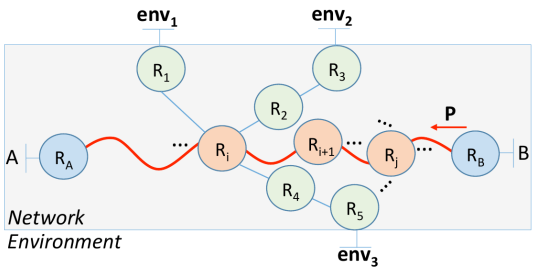
Exploring the Model



22 ▷ Accounting for on-path ACLs. $Router_i^k$ is the kth router on $path_{A \rightsquigarrow B}^i$

23 $$reachability_{B \rightsquigarrow A}^{path_{B \rightsquigarrow A}^i} = reachability_{B \rightsquigarrow A}^{path_{B \rightsquigarrow A}^i} \wedge (\bigvee_k ACLs_{Router_i^k})$$

Exploring the Model



24 ▷ Compute all paths reachability

25 $reachability_{A \rightsquigarrow B} = \bigvee_{i} reachability^{path^{i}_{B \rightsquigarrow A}}_{A \rightsquigarrow B}$

26 **return** $reachability_{A \rightsquigarrow B}$

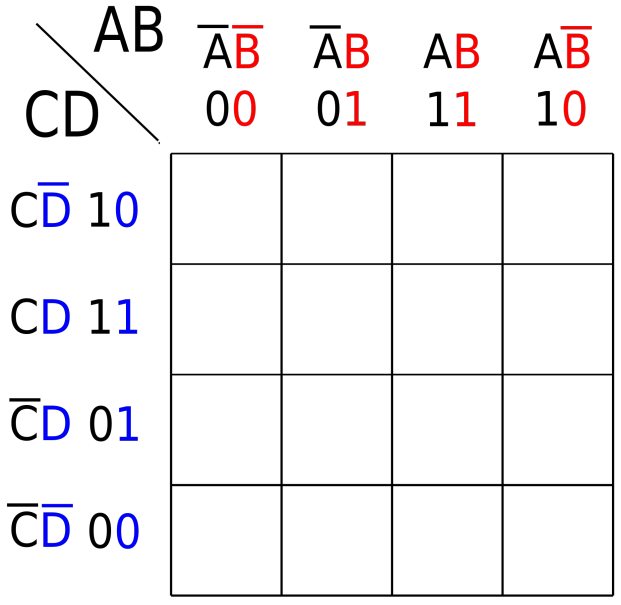
Prefixes reaching from B to A, if they are equal to $\phi_{A \rightarrow B}$ then it is policy compliant, else,

ERA: performs a *K Map* operation on the violating prefixes and outputs the result.

Scalability Optimizations

Aim to compute the reachability from A to B in seconds. Naive implementation of the control mode and exploration will not achieve this.

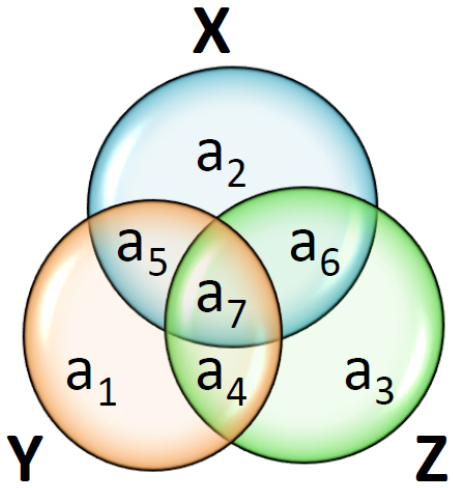
- K-Map



Scalability Optimizations

Same, IP or prefix appears multiple times in Network, which are treated equivalently. ERA removes this redundant data.

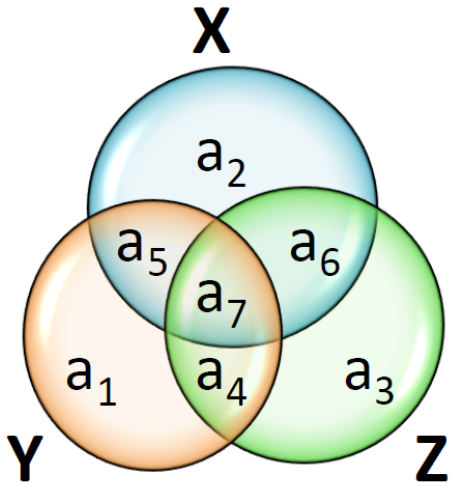
- Equivalence Class



Scalability Optimizations

Same, IP or prefix appears multiple times in Network, which are treated equivalently. ERA removes this redundant data.

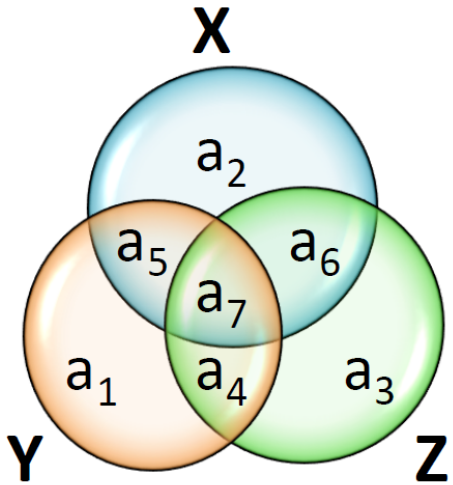
- Equivalence Class
  - Need to find: $X \wedge Y \wedge Z$



Scalability Optimizations

Same, IP or prefix appears multiple times in Network, which are treated equivalently. ERA removes this redundant data.

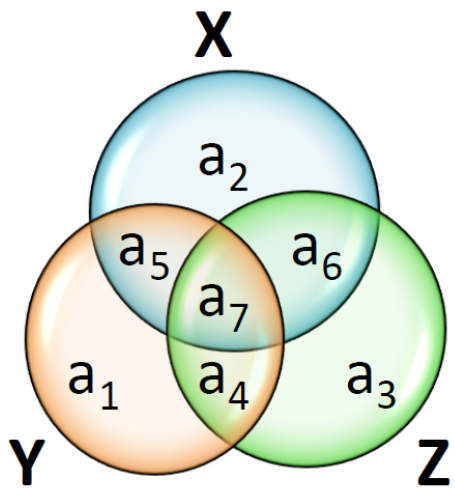
- Equivalence Class
  - Need to find: $X \wedge Y \wedge Z$
  - Express each term, in terms of equivalence class:
    $X = a_2 \vee a_5 \vee a_6 \vee a_7$



Scalability Optimizations

Same, IP or prefix appears multiple times in Network, which are treated equivalently. ERA removes this redundant data.

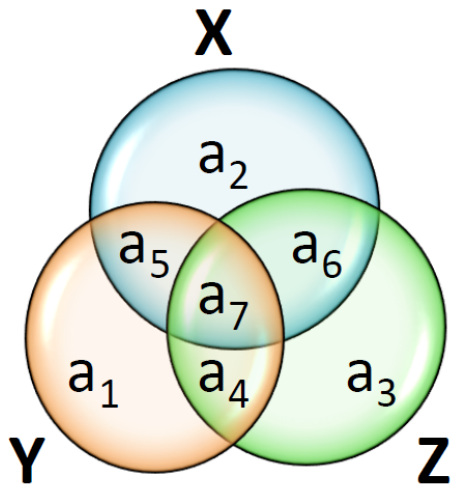
- Equivalence Class
  - Need to find: $X \wedge Y \wedge Z$
  - Express each term, in terms of equivalence class: $X = a_2 \vee a_5 \vee a_6 \vee a_7$
  - Represent each term using indices of members of equivalence class.
  - X is union of members: $2, 5, 6$ *and* 7



Scalability Optimizations

Same, IP or prefix appears multiple times in Network, which are treated equivalently. ERA removes this redundant data.

- Equivalence Class
  - Need to find: $X \wedge Y \wedge Z$
  - Express each term, in terms of equivalence class: $X = a_2 \vee a_5 \vee a_6 \vee a_7$
  - Represent each term using indices of members of equivalence class.
  - X is union of members: $2, 5, 6$ *and* 7



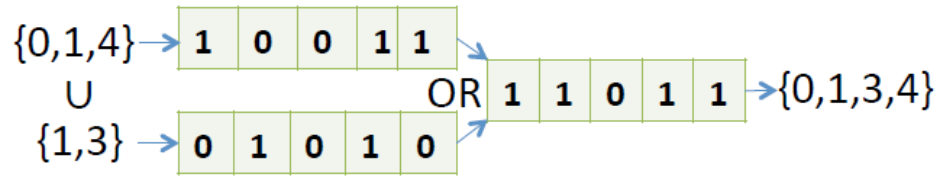
$X \wedge Y \wedge Z = \{2, 5, 6, 7\} \cap \{1, 4, 5, 7\} \cap \{3, 4, 6, 7\}$

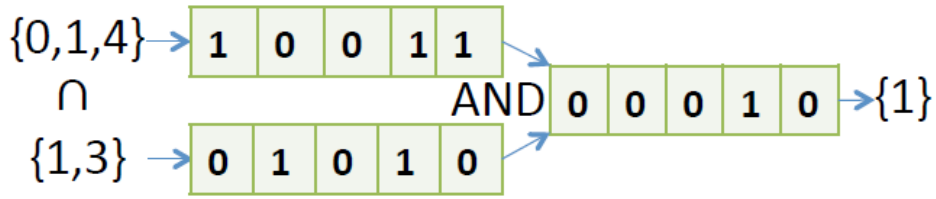
$X \wedge Y \wedge Z = 7$ which is, a_7

Scalability Optimizations

In previous case, the analysis involves computing union and intersection of sets of integers.

- Fast Set Operation
  - Vectorized Instructions on processors.
  - Set is represented as bit vector
  - Union/Intersection is reduced to OR/AND operation.





Beyond Reachability

Along with *Reachability* some similar kind of problems can be solved using ERA.

Beyond Reachability

Along with *Reachability* some similar kind of problems can be solved using ERA.

1. Valley-free Routing: Should not advertise routes learned from one provider/peer to another provider/peer.

Beyond Reachability

Along with *Reachability* some similar kind of problems can be solved using ERA.

1. Valley-free Routing: Should not advertise routes learned from one provider/peer to another provider/peer.
2. Equivalence of two Routers: If two boolean function defined over n boolean variables are equivalent their Reduced Order BDDs (ROBDDs) are identical.

Beyond Reachability

Along with *Reachability* some similar kind of problems can be solved using ERA.

1. Valley-free Routing: Should not advertise routes learned from one provider/peer to another provider/peer.
2. Equivalence of two Routers: If two boolean function defined over n boolean variables are equivalent their Reduced Order BDDs (ROBDDs) are identical.
3. Black hole freeness: A router unintentionally drops traffic.

Beyond Reachability

Along with *Reachability* some similar kind of problems can be solved using ERA.

1. Valley-free Routing: Should not advertise routes learned from one provider/peer to another provider/peer.
2. Equivalence of two Routers: If two boolean function defined over n boolean variables are equivalent their Reduced Order BDDs (ROBDDs) are identical.
3. Black hole freeness: A router unintentionally drops traffic.
4. Way pointing: Traffic from A to B should go through intended set of routers.

Beyond Reachability

Along with *Reachability* some similar kind of problems can be solved using ERA.

1. Valley-free Routing: Should not advertise routes learned from one provider/peer to another provider/peer.
2. Equivalence of two Routers: If two boolean function defined over n boolean variables are equivalent their Reduced Order BDDs (ROBDDs) are identical.
3. Black hole freeness: A router unintentionally drops traffic.
4. Way pointing: Traffic from A to B should go through intended set of routers.
5. Loop Freeness: ERA can check whether same router port appears again in the path.

Implementation

- Supports several configuration language (e.g., Cisco IOS, JunOS, Arista).
- Uses *Batfish* configuration parser to make it vendor agnostic.
- Control Plane Model, K-map and atomic predicates in *Java*.
- *JDD* Library for BDD.
- Fast set intersection and Union algorithm in *C* using Intel AVX2, where integer operations is done over 256 bits.

Two authors took few hours to model the common routing protocols. As there are not many protocols and key difference is how protocol prefers one route over the other.

Evaluation

Setup:

- Finds new and known bugs across real and synthetic scenarios.
- Focus on latent bugs, get triggered in specific situations.
  - Existing verification tools lack this feature.
  - Scaling these tools to do this is a challenge.
- Assumption that environment sends *all* possible route announcements.
  - This does not guarantee that all possible environments are covered.
  - This provides a *maximal* kind of setting.
  - Compared to *Batfish* where user has to craft the environment assumptions very carefully.

Known bug in Synthetic scenario

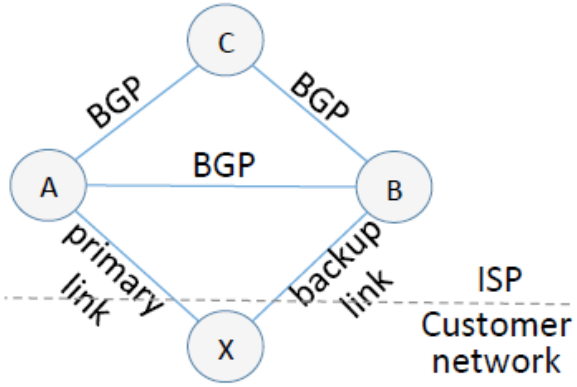


Figure 13: Known Bug in Synthetic setting: Waypoint

Setup

Customer wants to waypoint its traffic through X - A - C and use X - B - C as the backup path.

Known bug in Synthetic scenario

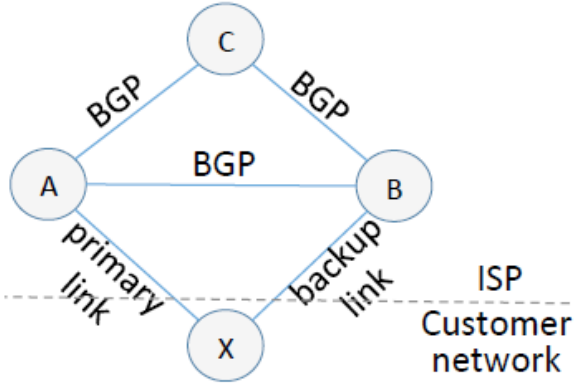


Figure 13: Known Bug in Synthetic setting: Waypoint

Problem

A static routes on A and B are redistributed into BGP and ISP forwards it to rest of Internet. Because of which B - X acts as primary link.

Known bug in Synthetic scenario

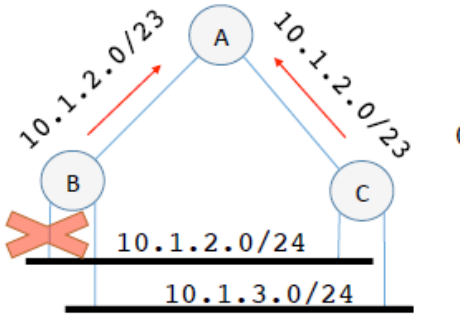


Figure 13: Known Bug in Synthetic setting: Blackhole

Setup

Both routers B and C are configured to announce aggregate route 10.1.2.0/23 to router A.

Known bug in Synthetic scenario

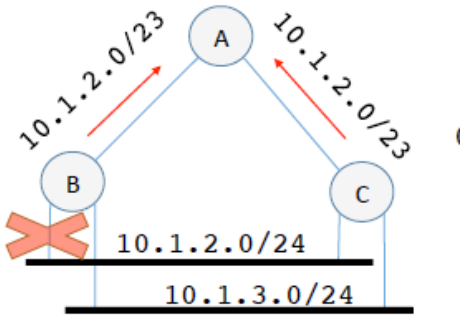


Figure 13: Known Bug in Synthetic setting: Blackhole

Problem

After the marked interface of B fails, B continues to announce the aggregate route, which causes A to send packets destined to 10.1.2.0/24 to B. B will drop this traffic.

Known bug in Synthetic scenario

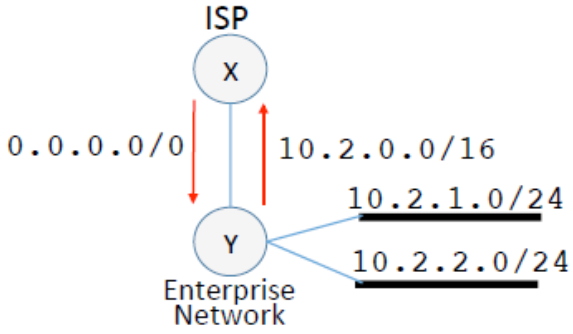


Figure 13: Known Bug in Synthetic setting: Loop

Setup

ISP router X advertises the default route 0.0.0.0/0 to router Y . Even though Y has connectivity to only 10.2.1.0/24 and 10.2.2.0/24, it has been configured to advertise to the ISP the aggregate route for the entire 10.2.0.0/16 prefix.

Known bug in Synthetic scenario

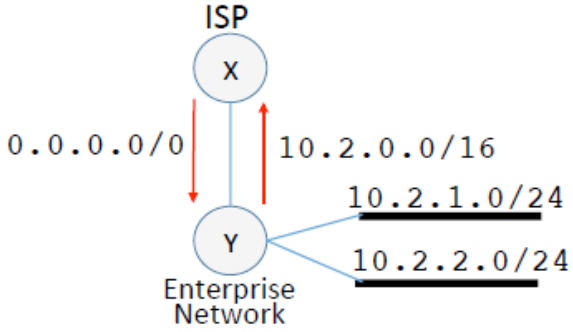


Figure 13: Known Bug in Synthetic setting: Loop

Problem

As, 10.3.0.0/24 is as subprefix of 10.2.0.0/16, the ISP may send traffic to destination prefix 10.3.0.0/24 to Y . Since Y does not know how to reach 10.3.0.0/24, this traffic will match its default route entry and be bounced back to the ISP. Hence, stuck in a loop.

Red-Blue team Bugs

Red team introduced bugs in the network and *Blue* team tries to find it.

Red-Blue team Bugs

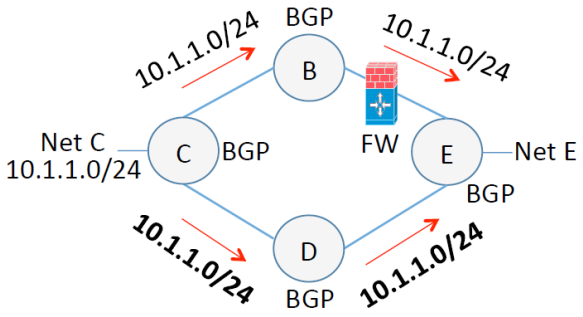


Figure 14: Red Blue team: Waypoint

Setup

The intended policy is to ensure traffic originating from network E destined to network C goes through path E - B - C (so that it is scrubbed by the firewall).

Red-Blue team Bugs

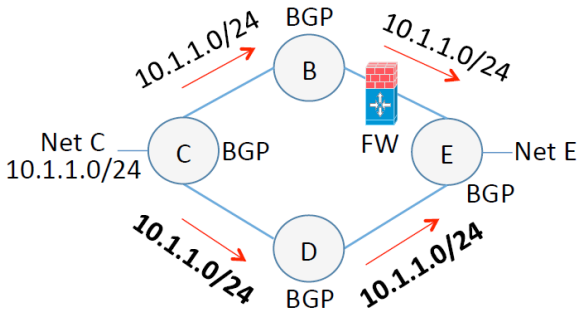


Figure 14: Red Blue team: Waypoint

Problem

However, as C and D fails to filter out the announcement **10.1.1.0/24** some traffic might end up taking the path, E - D - C.

Red-Blue team Bugs

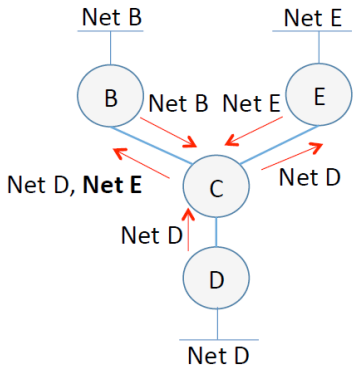


Figure 14: Red Blue team: Valley-Free

Setup

B and E are providers for C, which in turn, is a provider for D.

Red-Blue team Bugs

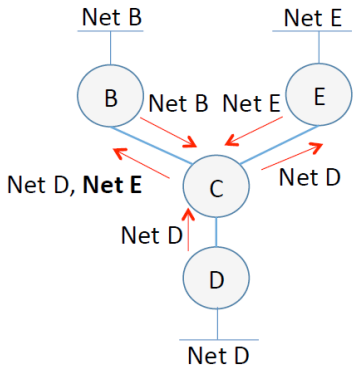


Figure 14: Red Blue team: Valley-Free

Problem

A missing export filter on C caused C to advertise the prefix for NetE to B. A violation of the valley-free routing property.

Red-Blue team Bugs

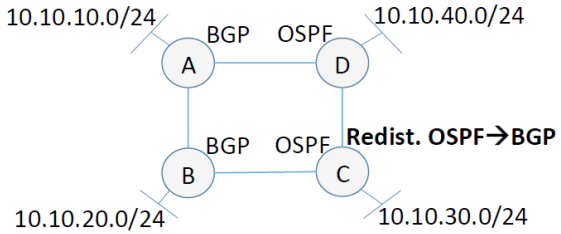


Figure 14: Red Blue team: Network Isolation

Setup

We want the traffic from segments $\{A, B\}$ (running BGP) and $\{D, C\}$ (running OSPF) to remain isolated from each other.

Red-Blue team Bugs

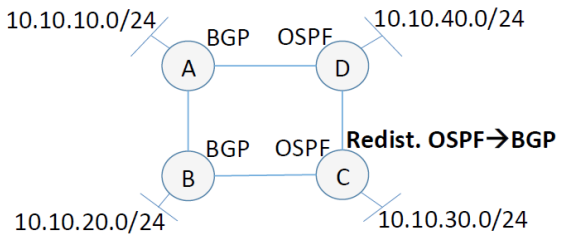


Figure 14: Red Blue team: Network Isolation

Problem

This policy is violated due to a misconfiguration on C whereby OSPF is redistributed into BGP, that will allow traffic from$\{A, B\}$ to reach $\{C, D\}$.

Red-Blue team Bugs

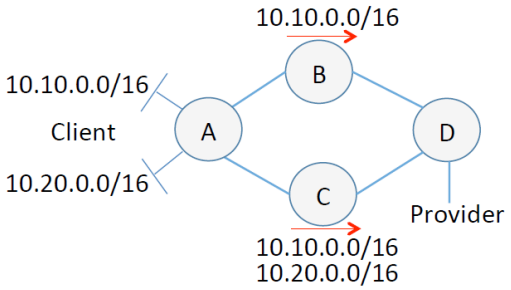


Figure 14: Red Blue team: Backup

Setup

The client has two /16 networks connected to A and intends to maintain two paths to the provider to ensure reachability in case of failure on one of them.

Red-Blue team Bugs

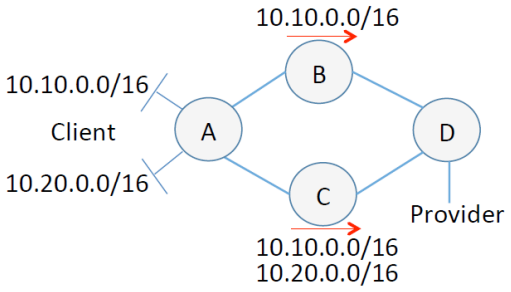


Figure 14: Red Blue team: Backup

Problem

The policy is violated because of an incorrect filter configured on B that drops the advertisement for the 10.20.0.0/16 network. As a result, if path D - C - A fails, the 10.20.0.0/16 network will be unreachable from the provider.

Hybrid Network and Fibbing

- Hybrid Network: SDN (Software Defined Networking) is deployed along side traditional network routing infrastructure for scalability and fault tolerance.
- Fibbing: Allows an operator to use an SDN controller to flexibly enforce way-pointing policies in a network running OSPF.

Fibbing Example

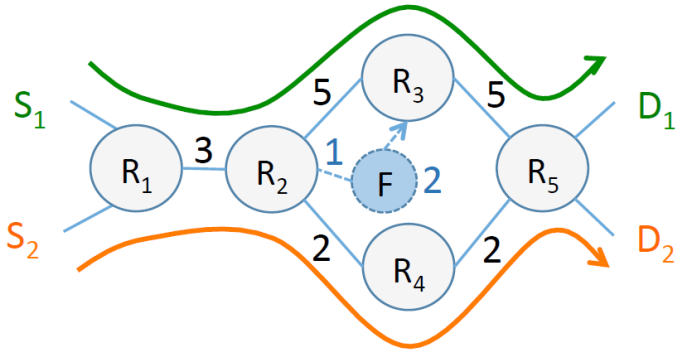


Figure 15: Fibbing Example

Fibbing Example

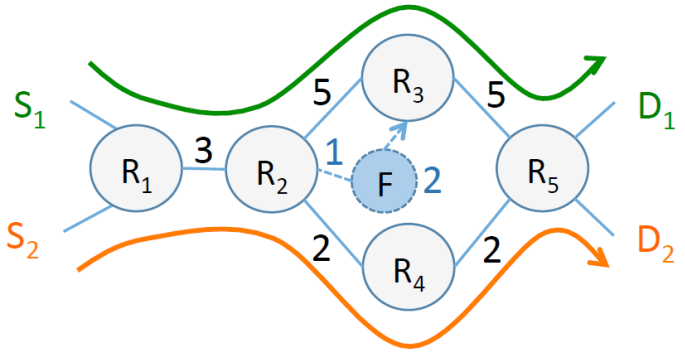


Figure 15: Fibbing Example

- In OSPF, source to destination path will take cheaper path $R_1 - R_2 - R_2 - R_4 - R_5$.

Fibbing Example

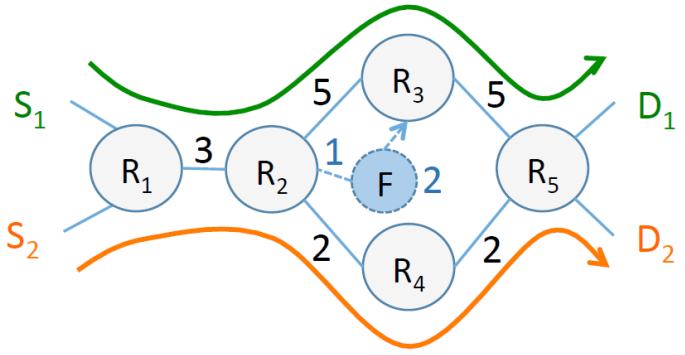


Figure 15: Fibbing Example

- In OSPF, source to destination path will take cheaper path $R_1 - R_2 - R_2 - R_4 - R_5$.
- For load balancing, traffic from $S_1 - D_1$ should take path$R_1 - R_2 - R_2 - R_3 - R_5$.

Fibbing Example

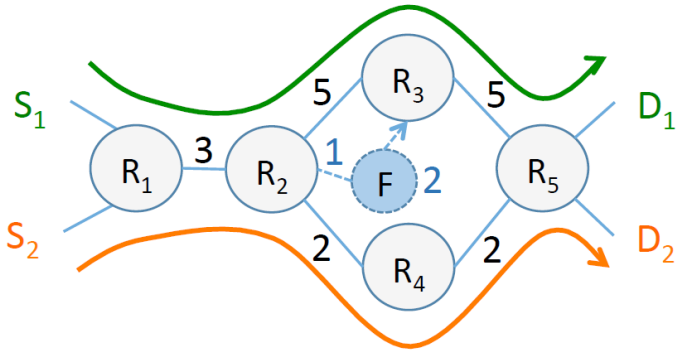


Figure 15: Fibbing Example

- In OSPF, source to destination path will take cheaper path $R_1 - R_2 - R_2 - R_4 - R_5$.
- For load balancing, traffic from $S_1 - D_1$ should take path$R_1 - R_2 - R_2 - R_3 - R_5$.
- Add a fake router F, which announce that it can reach D_1 at a cost of 2.

Fibbing Example

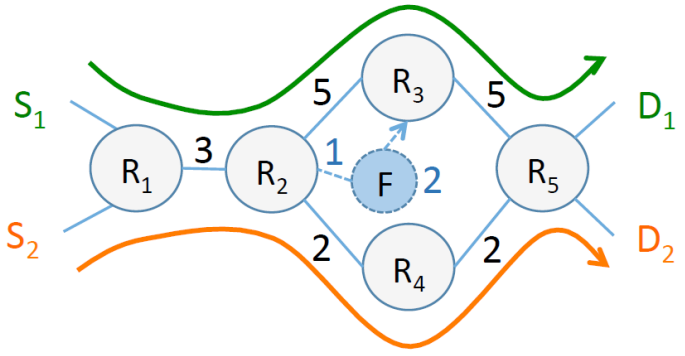


Figure 15: Fibbing Example

- In OSPF, source to destination path will take cheaper path $R_1 - R_2 - R_2 - R_4 - R_5$.
- For load balancing, traffic from $S_1 - D_1$ should take path$R_1 - R_2 - R_2 - R_3 - R_5$.
- Add a fake router F, which announce that it can reach D_1 at a cost of 2.

These networks are error prone !!

New bugs: Synthetic Scenario

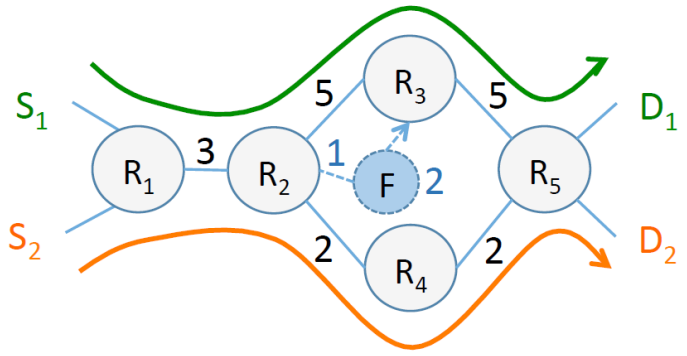


Figure 16: Route aggregation on R2

Setup

Aim is to use fibbing to enforce the waypointing denoted by green and orange paths.

New bugs: Synthetic Scenario

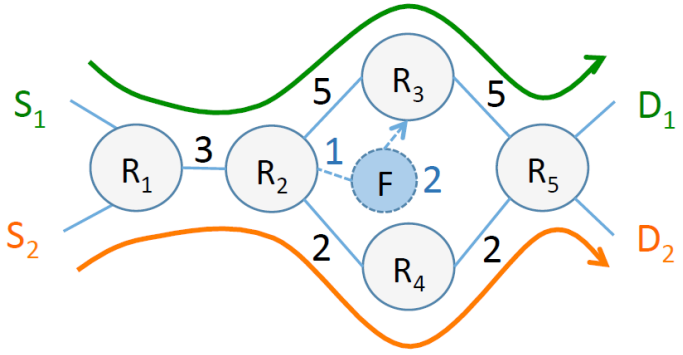


Figure 16: Route aggregation on R2

Problem

There is a aggregate route configured on R_2 to destination prefix $D_1 \cup D_2$ pointing to R_4 as its next hop. As a result, both $S1 \rightarrow D1$ and $S2 \rightarrow D2$ traversed the orange path, which violated the policy.

New bugs: Synthetic Scenario

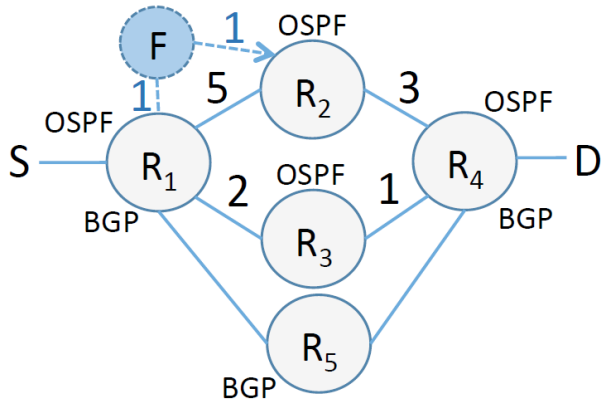


Figure 16: Cross Protocol Effects

Setup

Aim is to use fibbing to waypoint traffic to D through $R_1 - R_2 - R_4$ by using a fake router F.

New bugs: Synthetic Scenario

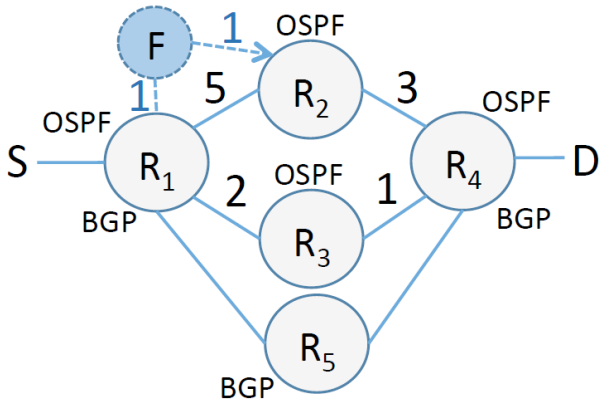


Figure 16: Cross Protocol Effects

Problem

There is a static route between R_4 to D, that is redistributed into BGP and OSPF. R_1 receives announcement from both OSPF (R_2 and R_3) and BGP (R_5). BGP has a AD value less than OSPF, hence that route is preferred leading to a violation.

New bugs: Synthetic Scenario

Takeaway:

1. Fibbing is proved to be correct if the network only uses OSPF.
2. Hybrid networks to be practical, and realistic router configurations has to be accounted for.
3. ERA cannot handle random SDN bugs.
4. ERA handles SDN only if its behavior can be abstracted in control plane model defined earlier.

Known Bug in Real Scenarios

1. Route Leak, involves
   - A router incorrectly advertising the destination prefix.
   - Another router accepting it.
   - High impact, e.g. Google and Amazon outages in 2015.

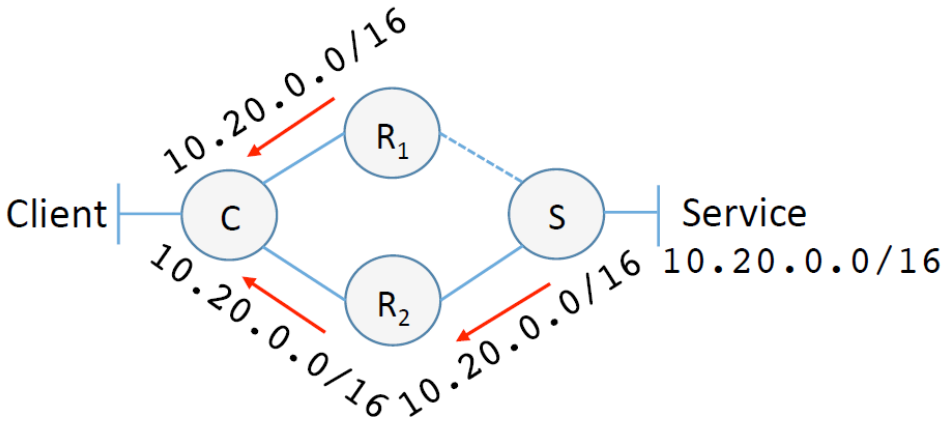


Figure 16: Route Leak

Known Bug in Real Scenarios

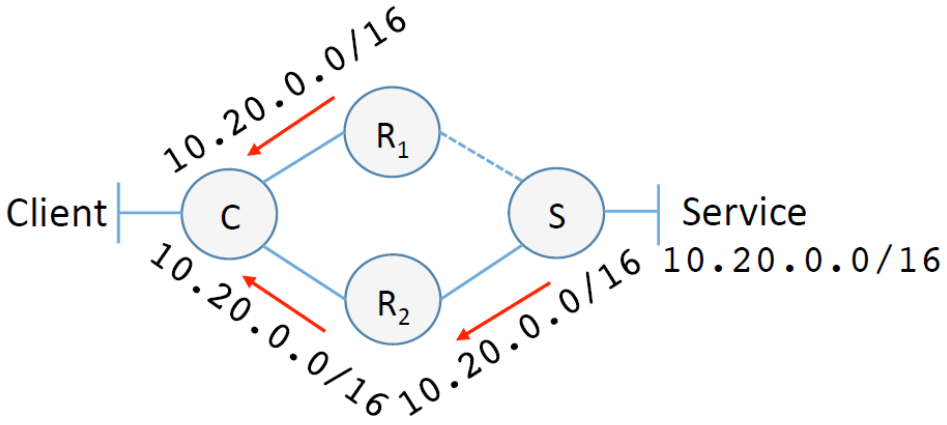


Figure 16: Route Leak

Setup

The intended path from the client to the service is through R_2;

Known Bug in Real Scenarios

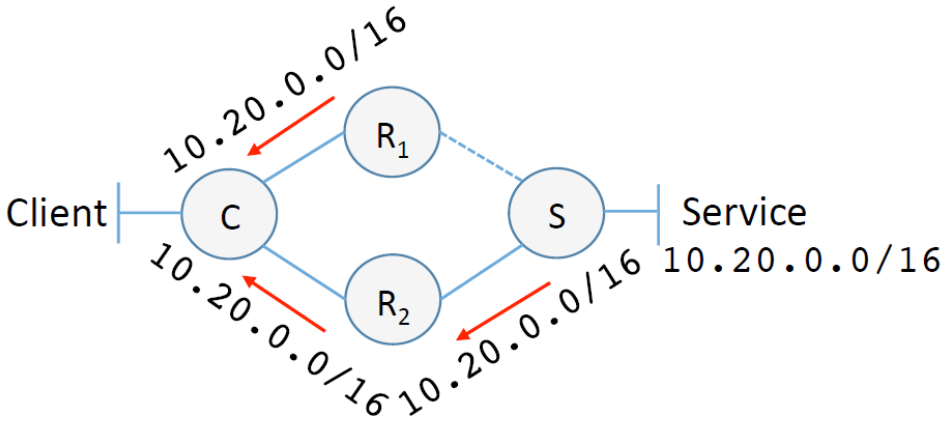


Figure 16: Route Leak

Problem

The client's traffic ends up taking the wrong path $C \rightarrow R_1$ because

- R_1 incorrectly advertises the service prefix.
- C prefers the route advertisement made by R_1 over the one made by R_2.

New bugs in Real Scenarios

Campus Network:

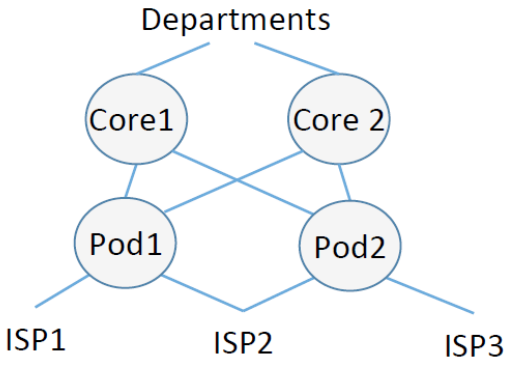


Figure 17: Router Equivalence

Setup

Core2 is meant to be *Core1's* backup. Ideally, they should be equivalent to each other.

New bugs in Real Scenarios

Campus Network:

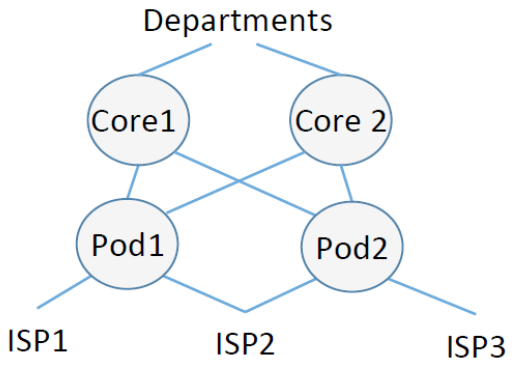


Figure 17: Router Equivalence

Problem

Core1 has OSPF configured on one of its interfaces, which is missing on *Core2*. As a result, if *Core1* fails, the departments that rely on OSPF will be disconnected from the Internet.

New bugs in Real Scenarios

Campus Network:

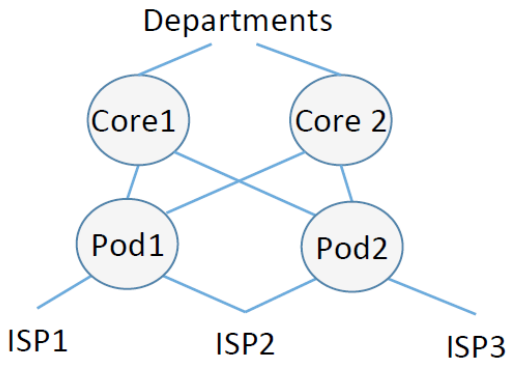


Figure 17: Router Equivalence

Setup

*Pod*1 and *Pod*2, connecting the campus to the Internet, are both connected to ISP2 with the intention that link $Pod1 - ISP2$ is active and $Pod2 - ISP2$ is its backup.

New bugs in Real Scenarios

Campus Network:

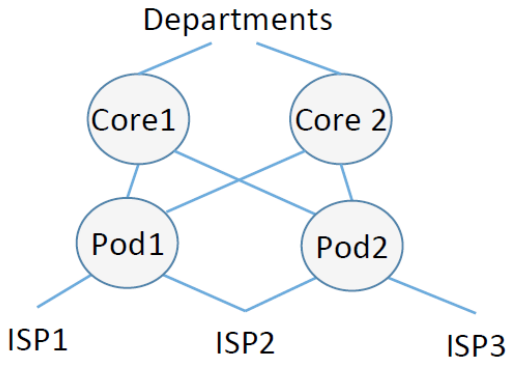


Figure 17: Router Equivalence

Problem

ACLs on Pod1 and Pod2 affecting their respective links with ISP2 are different. Pod2 has more restrictive ACLs than Pod1. If link $Pod1 - ISP2$ fails, a subset of $campus - to - ISP2$ traffic will be mistakenly dropped by Pod2.

New bugs in Real Scenarios

Bugs in *CloudNet*;

- Check the equivalence of same-tier routers, on configuration of seven production datacenters of a large cloud provider.
- Seven routers in two datacenters had a total of 19 static routes responsible for violations of equivalence policies.

Scalability of ERA

- **TestBed:** Desktop machine (4-core 3.50GHz, 16GB RAM).
- Compare with Batfish
  - Takes concrete network environment.
  - Runs a high fidelity model of the control plane to generate data plane.
  - Performs data reachability analysis.
- Batfish took about 4 seconds. ERA took 0.17 seconds to analyze the same network (a 23X speedup over Batfish).
- Batfishs performance will degrade as the size of the environment increases.
- BDD-based approach allows it to naturally handle even the maximal environment, represented by the BDD true.

Effect of Optimizations

Topo.	#routers/ave path len.	Reachability analysis latency (sec)			
		baseline	kmap	kmap+EC	ERA
Stanford	16/2	5	1.8	0.30	0.29
OTEglb	92/3.3	7.8	3.5	1.97	1.84
FatTree	1,024/5.89	13.8	7.01	6.1	5.4
Purdue	1,646/6.8	15	8	6.5	6

Figure 17: Effect of Optimizations

- Optimizations yield a speedup of 2.5× to 17× making ERA sufficiently fast to be interactively usable.

Conclusion

- Constantly changing network. Tool required to reason about reachability policy across these changes.
- Current tool focuses on a part of control plane or on data plane.
- ERA: Models complete control plane and then argues *Reachability* in that plane.
- The model expresses key behaviours and a scalable mode using a protocol invariant *route* abstraction, BDD and scalable boolean operation.
- ERA provide near-real-time analysis capabilities which can scale to datacenter and enterprise networks.
- It DOES NOT automatically reason about all the environments.
- User has to specify the environment using BDD.

Future Work

- Extend ERA to cover all the environments automatically.
- Bug fixing prioritization.

Future Work

Using Verification Techniques

A General Approach to Network Configuration Verification

Ryan Beckett
Princeton University

Aarti Gupta
Princeton University

Ratul Mahajan
Microsoft Research

David Walker
Princeton University

Abstract— We develop an approach to verify network configurations that is (1) *general* whereas prior work is limited in terms of protocols or features, (2) *accurate* whereas prior work approximates, (3) *complete* in that it analyzes all possible sets of routing announcements from external sources rather than just one or some, (4) *powerful* in that it can verify a wide range of properties such as reachability, path length,

Thus, it is highly desirable to perform control plane *verification*, which aims to analyze correctness in the face of *all* possible environments. Verification is more challenging because one cannot simply simulate the control plane with concrete environments. Rather, one must build a *model* in which the environment is *symbolic* (*i.e.*, represented as a variable).

Several researchers have considered the network verifica-

plane behaviors. It then uses an SMT solver to determine if the paths that can emerge when the control plane converges satisfy properties of interest. We implement our approach in

https://batfish.github.io/minesweeper/