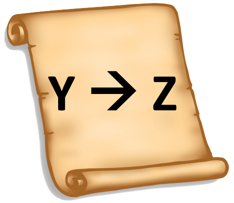
Tiramisu: Fast Multilayer Network Verification

Anubhavnidhi "Archie" Abhashkumar[*], Aaron Gember Jacobson[#], and Aditya Akella[*]

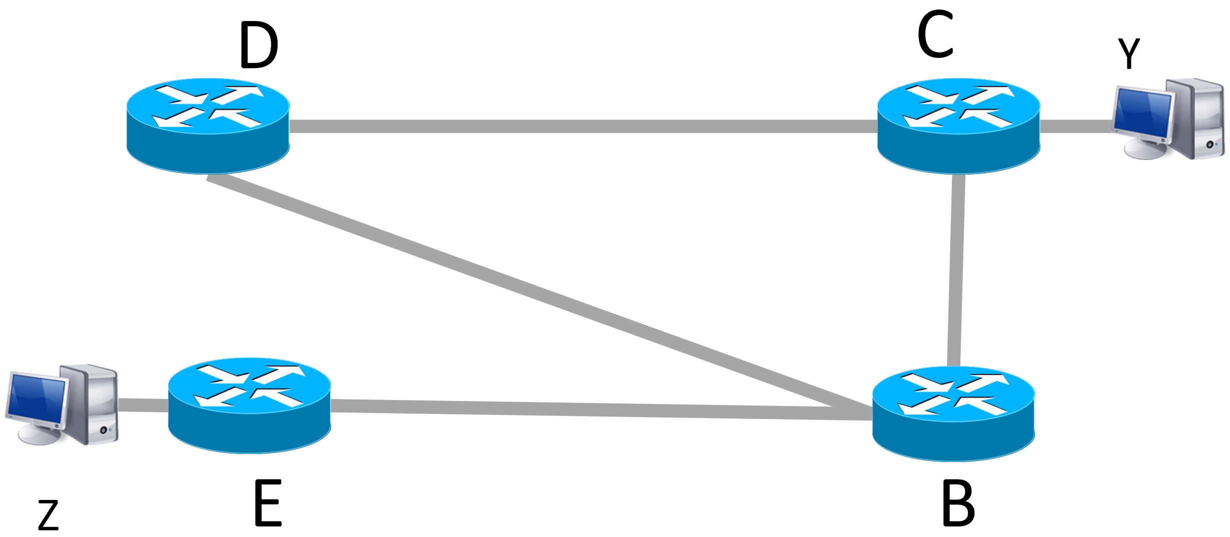
[*]University of Wisconsin-Madison, [#] Colgate University

Distributed Control Planes

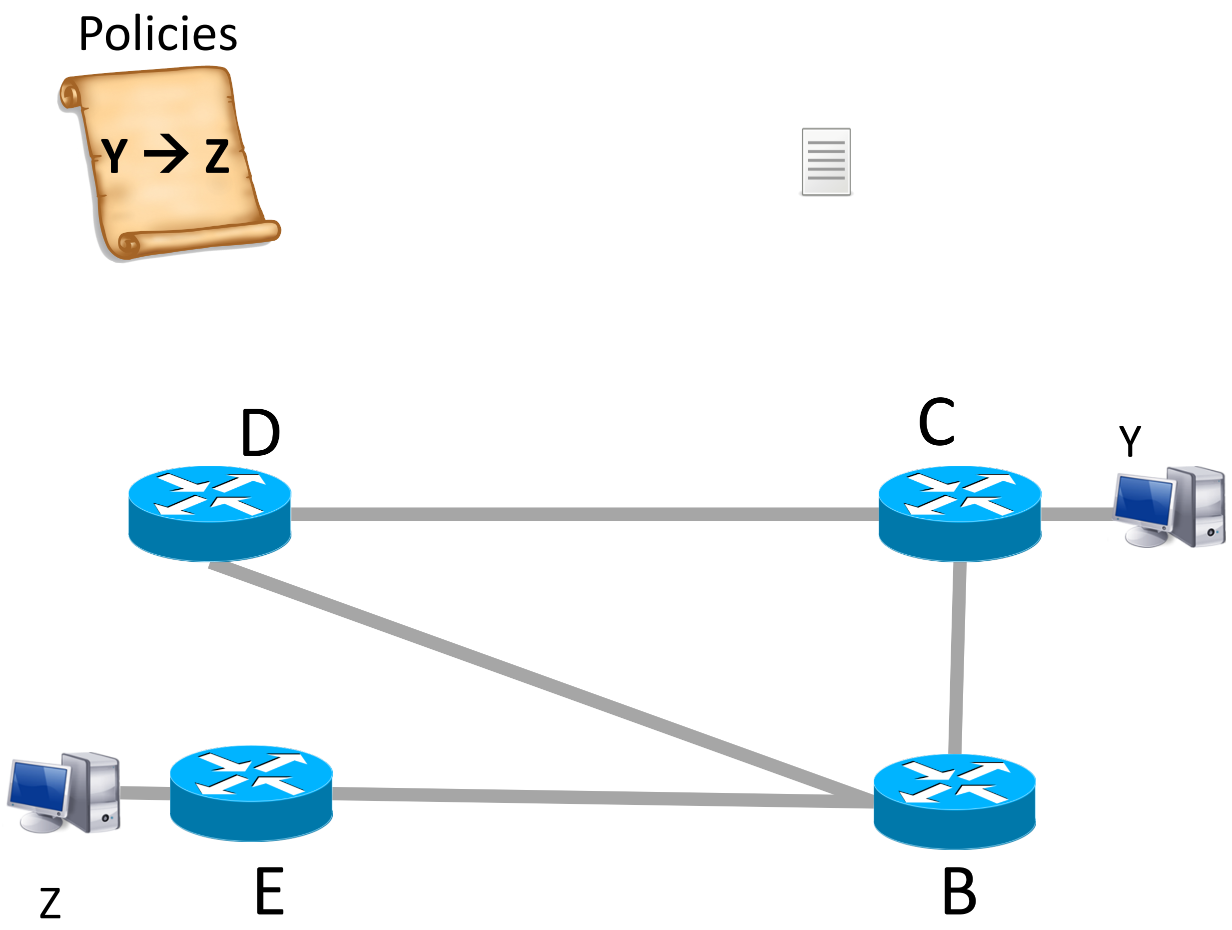


Configuration Example

```
interface GigabitEthernet0/1
	ip address 1.0.1.1 255.255.0.0
	ip ospf cost 1
router ospf 10
	network 3.0.1.2 0.0.255.255 area 0
```

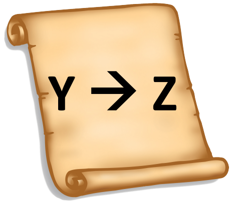


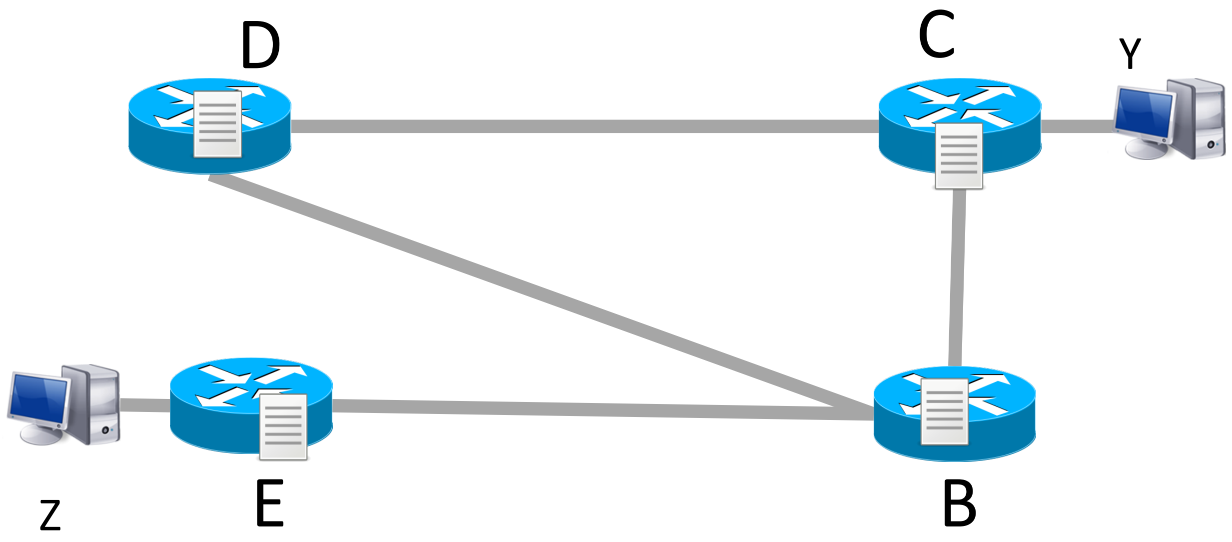
Distributed Control Planes



Distributed Control Planes

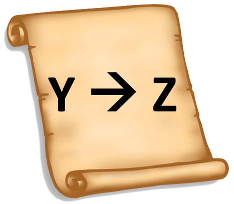
Policies

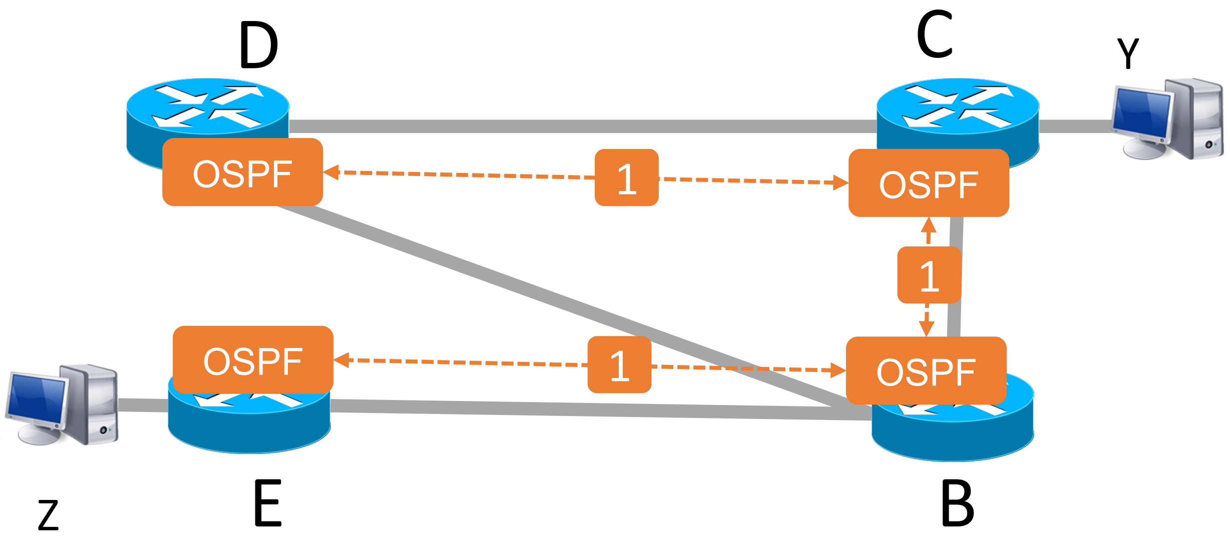




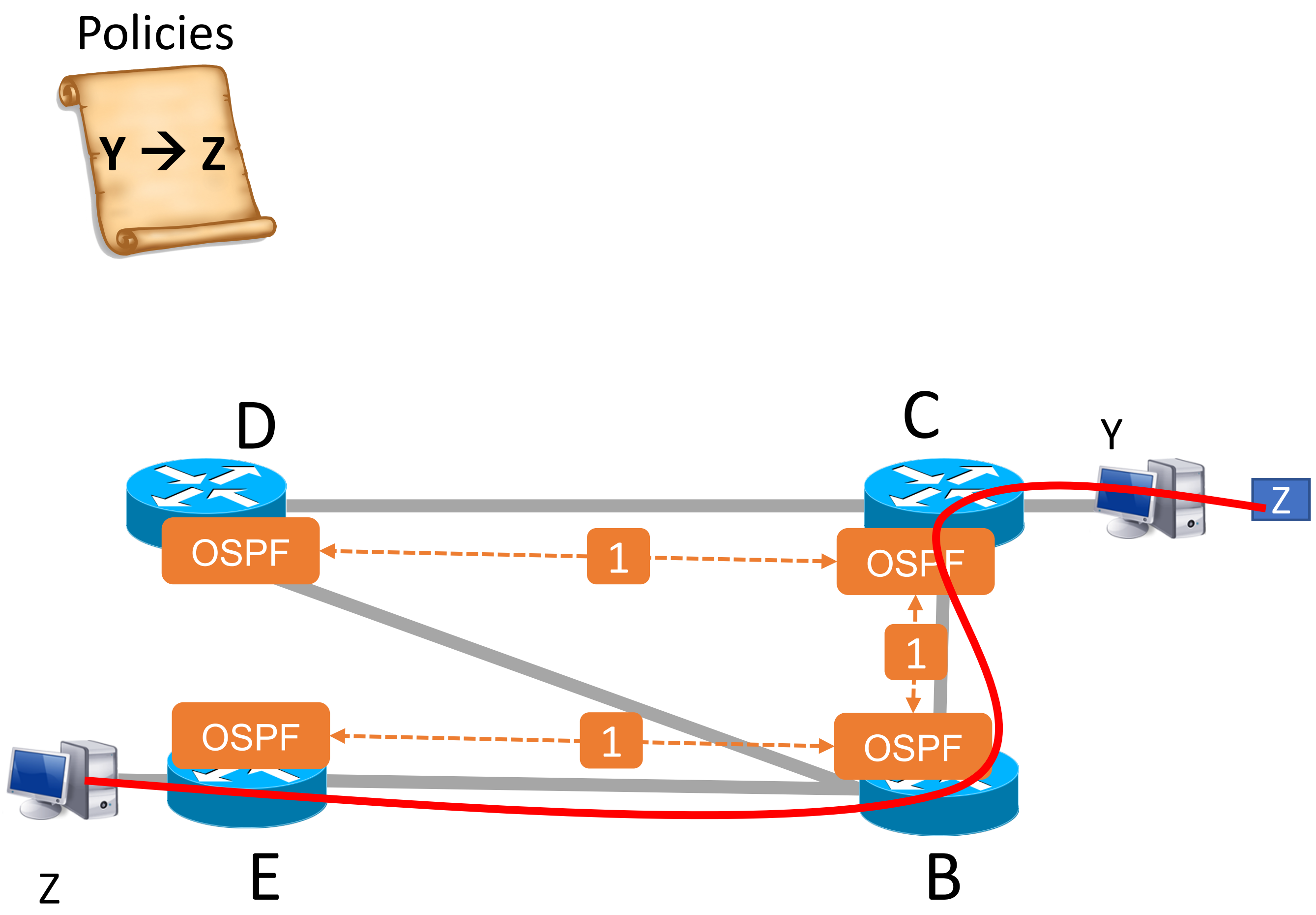
Distributed Control Planes

Policies





Distributed Control Planes



Router Configurations are Complex

```
interface GigabitEthernet0/1
ip address 1.0.1.1 255.255.0.0
ip ospf cost 1
!
interface GigabitEthernet0/2
ip address 10.11.11.1 255.255.0.0
!
router bgp 300
neighbor 2.2.2.2 route-map COMM out
!
route-map COMM permit 10
set community 1:1 additive
set local-preference 150
!
router ospf 10
network 3.0.1.2 0.0.255.255 area 0
```

- Multiple routing protocols
  - BGP
  - OSPF
  - …

Router Configurations are Complex

```
interface GigabitEthernet0/1
ip address 1.0.1.1 255.255.0.0
ip ospf cost 1
!
interface GigabitEthernet0/2
ip address 10.11.11.1 255.255.0.0
!
router bgp 300
neighbor 2.2.2.2 route-map COMM out
!
route-map COMM permit 10
set community 1:1 additive
set local-preference 150
!
router ospf 10
network 3.0.1.2 0.0.255.255 area 0
```

- Multiple routing protocols
  - BGP
  - OSPF
  - …
- Multiple routing metrics
  - ospf cost
  - local preference
  - …

Router Configurations are Complex

```
interface GigabitEthernet0/1
ip address 1.0.1.1 255.255.0.0
ip ospf cost 1
!
interface GigabitEthernet0/2
ip address 10.11.11.1 255.255.0.0
!
router bgp 300
neighbor 2.2.2.2 route-map COMM out
!
route-map COMM permit 10
set community 1:1 additive
set local-preference 150
!
router ospf 10
network 3.0.1.2 0.0.255.255 area 0
```

- Multiple routing protocols
  - BGP
  - OSPF
  - …
- Multiple routing metrics
  - ospf cost
  - local preference
  - …
- Multiple filters
  - Community
  - …

Configuration Complexity Make Errors Common

BGP errors are to blame for Monday's Twitter outage, not DDoS attacks

No, your toaster didn't kill Twitter, an engineer did

Level 3 blames huge network outage on human error

Level 3 says new prevention measures being taken; now on to Hurricane Matthew

Google made a tiny error and it broke half the internet in Japan

by MIX — 7 weeks ago in GOOGLE

Xbox Live outage caused by network configuration problem

BY TODD BISHOP on April 15, 2013 at 9:27 am

Microsoft: misconfigured network device led to Azure outage

30 July 2012 | By Yevgeniy Sverdlik

United says router issue responsible for grounding all flights

Configuration Complexity Make Errors Common

BGP errors are to blame for Monday's Twitter outage, not DDoS attacks

No, your toaster didn't kill Twitter, an engineer did

Level 3 blames huge network outage on human error

Level 3 says new prevention measures being taken; now on to ...w

Policy violations manifest under failures

Google made a t... and it broke half the internet in Japan

by MIX — 7 weeks ago in GOOGLE

Xbox Live outage caused by network configuration problem

BY TODD BISHOP on April 15, 2013 at 9:27 am

Microsoft: misconfigured network device led to Azure outage

30 July 2012 | By Yevgeniy Sverdlik

United says router issue responsible for grounding all flights

Configuration Complexity Make Errors Common

BGP errors are to blame for Monday's Twitter outage, not DDoS attacks

No, your toaster didn't kill Twitter, an engineer did

Level 3 blames huge network outage on human error

Level 3 says new prevention measures being taken; now on to

Google ma and it broke half the internet in Japan

by MIX — 7 weeks ago in GOOGLE

Verification tools can proactively check for failures

Xbox Live outage caused by network configuration problem

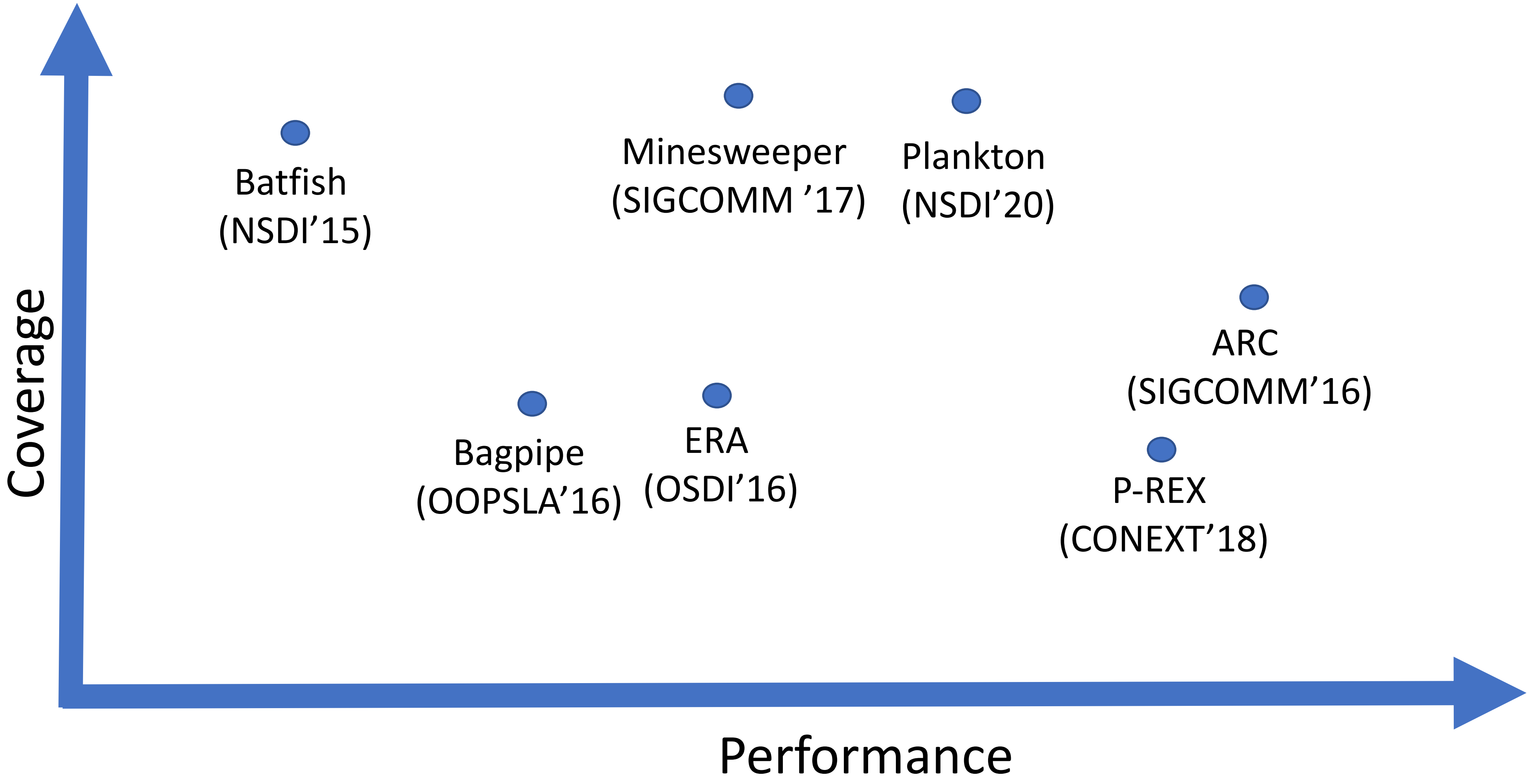
BY TODD BISHOP on April 15, 2013 at 9:27 am

Microsoft: misconfigured network device led to Azure outage

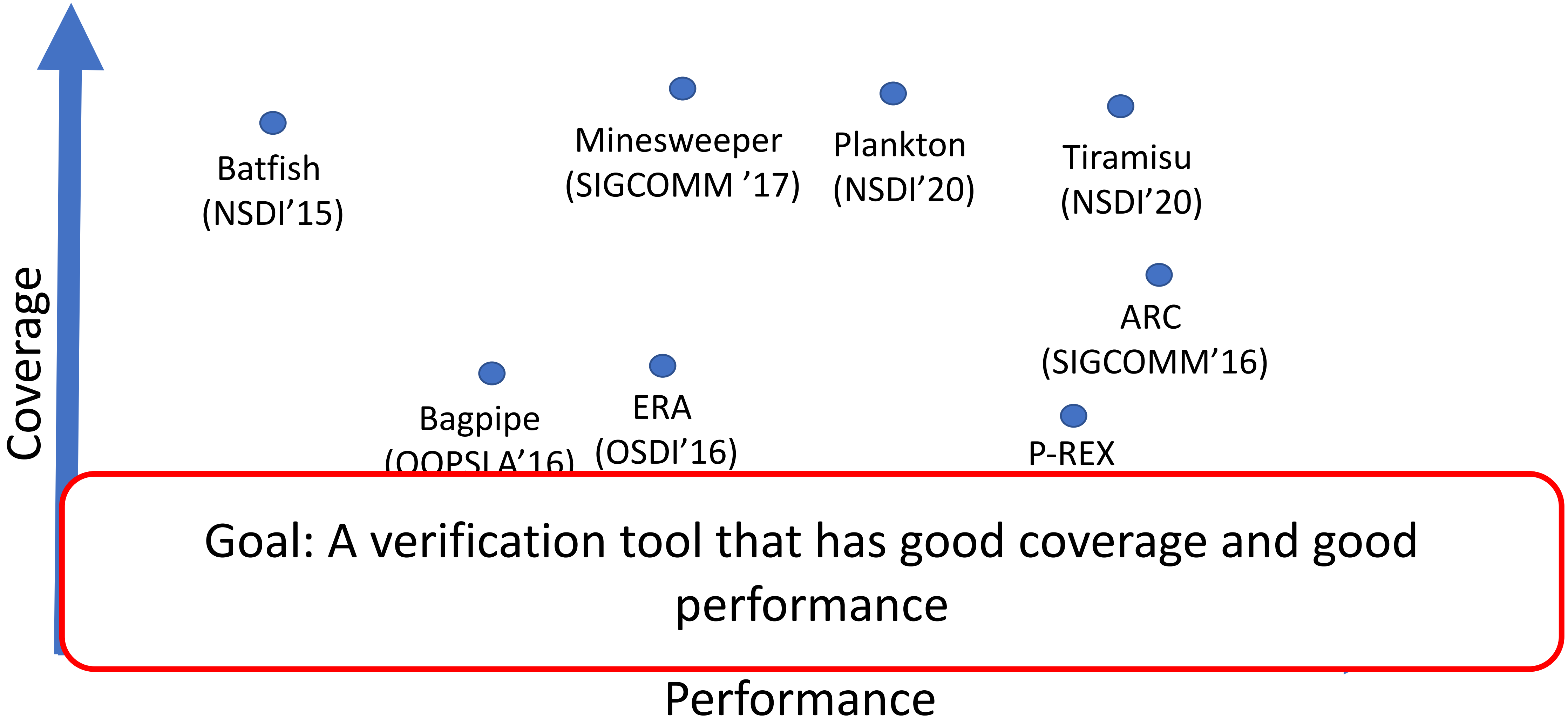
30 July 2012 | By Yevgeniy Sverdlik

United says router issue responsible for grounding all flights

Multiple Network Verification tools

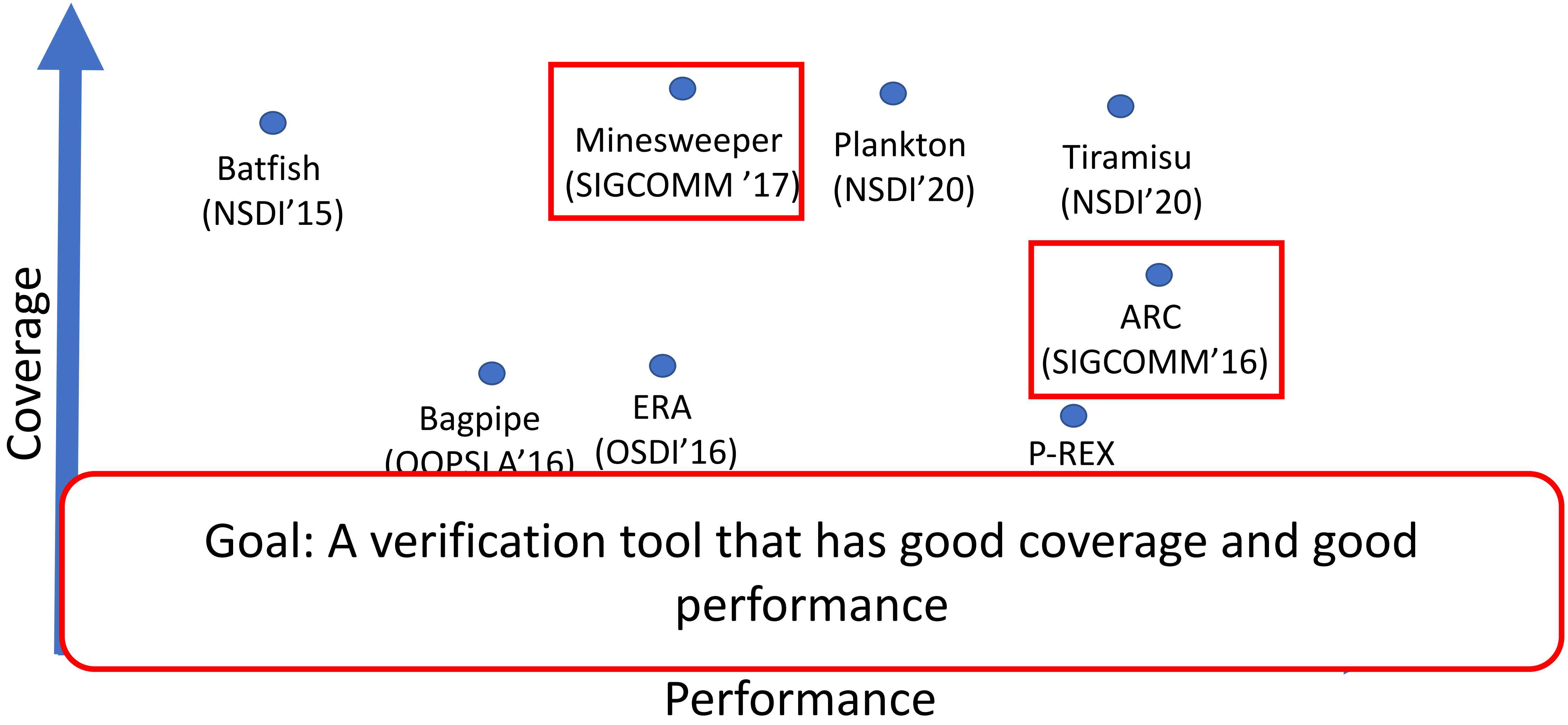


Multiple Network Verification tools



Goal: A verification tool that has good coverage and good performance

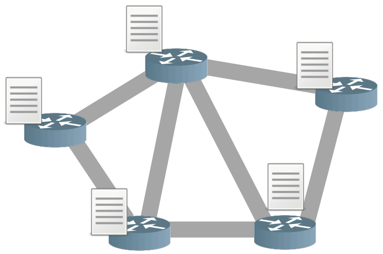
Multiple Network Verification tools



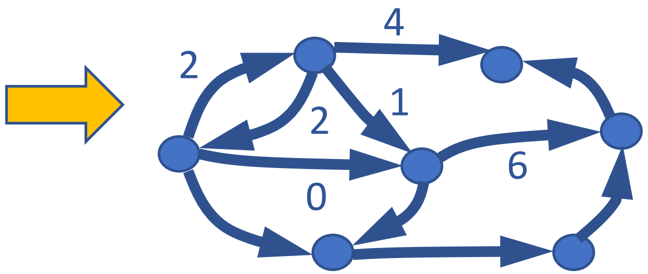
Goal: A verification tool that has good coverage and good performance

Performance VS Coverage

- ARC (SIGCOMM'16)
  - Graph algorithms



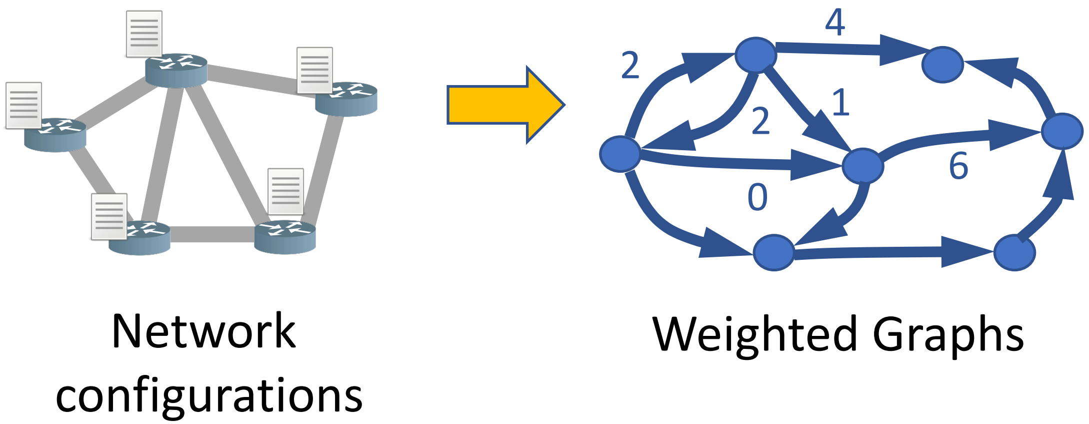
Network configurations



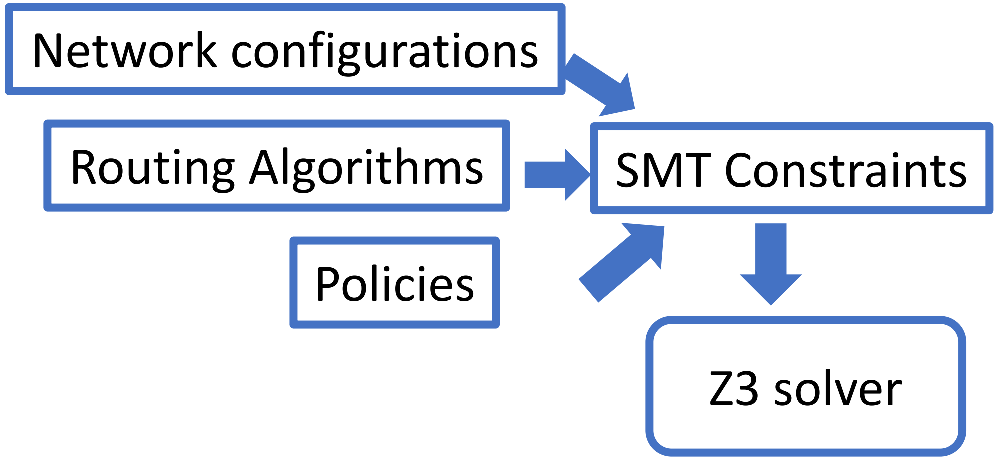
Weighted Graphs

Performance VS Coverage

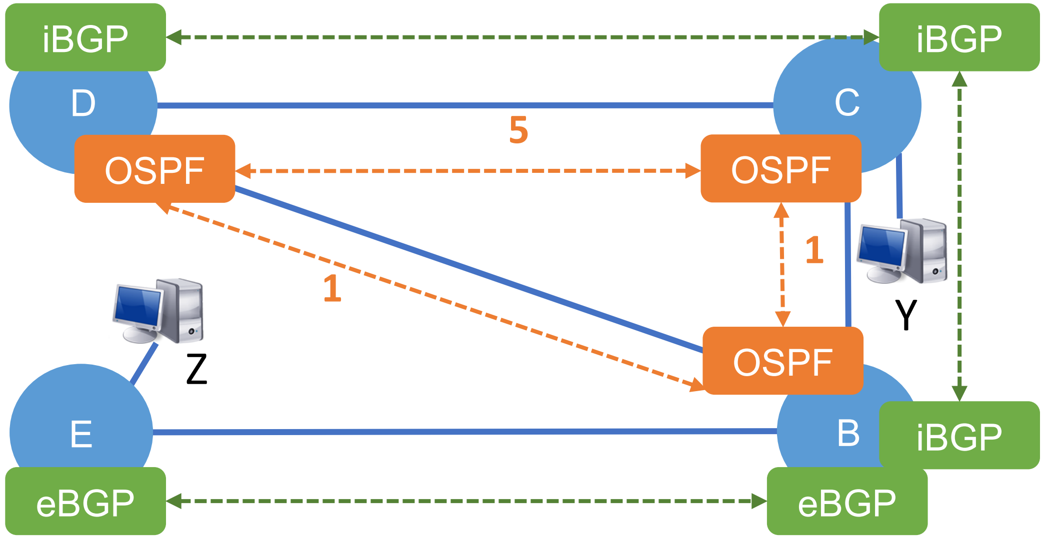
- ARC (SIGCOMM'16)
  - Graph algorithms



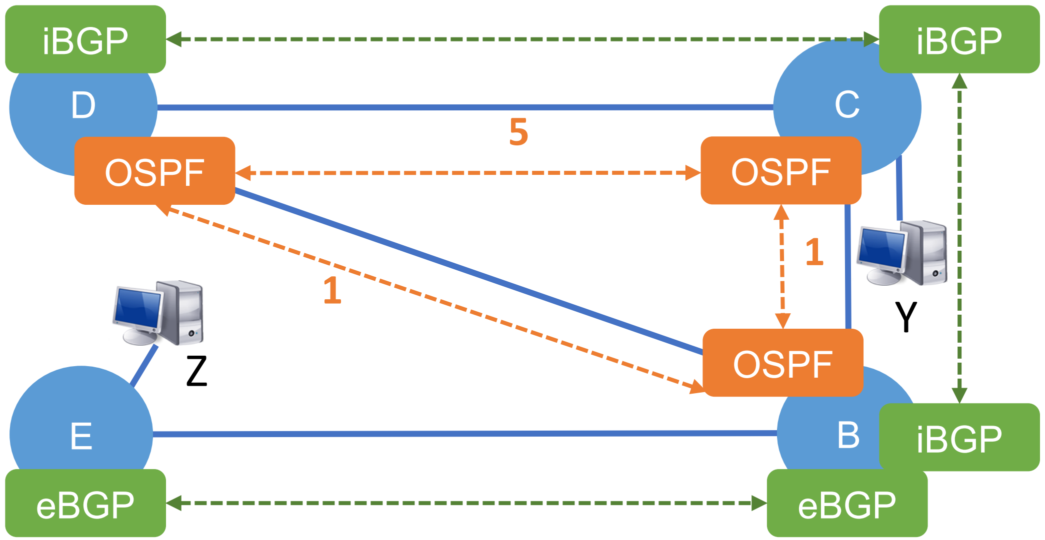
- Minesweeper (SIGCOMM'17)
  - Symbolic encoding



Cross Layer Dependency



Cross Layer Dependency

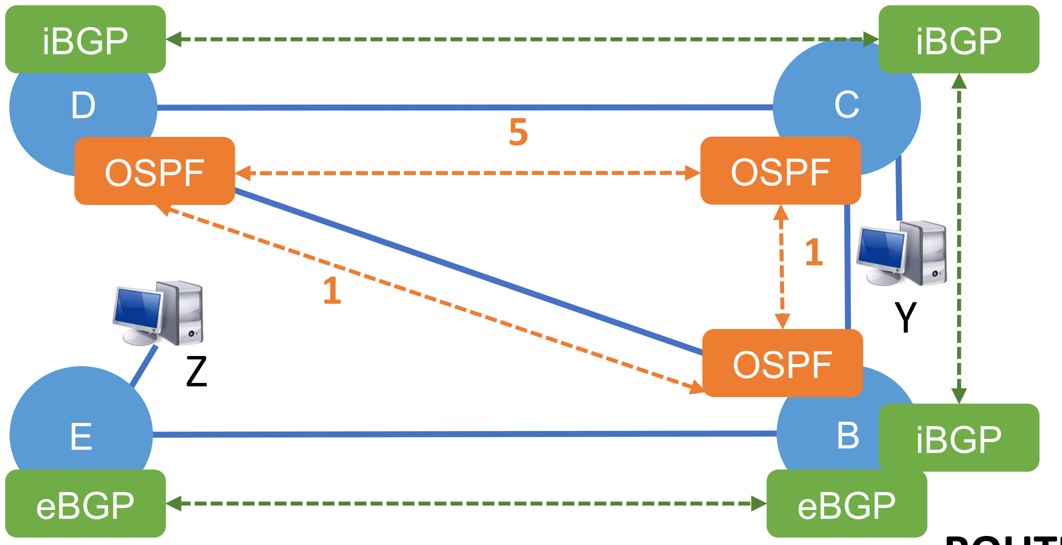


IBGP uses OSPF computed route to reach next hop router

Cross Layer Dependency

ROUTER D

Dst	NextHop
B	B
C	C



ROUTER C

Dst	NextHop
E	B
B	B
D	D

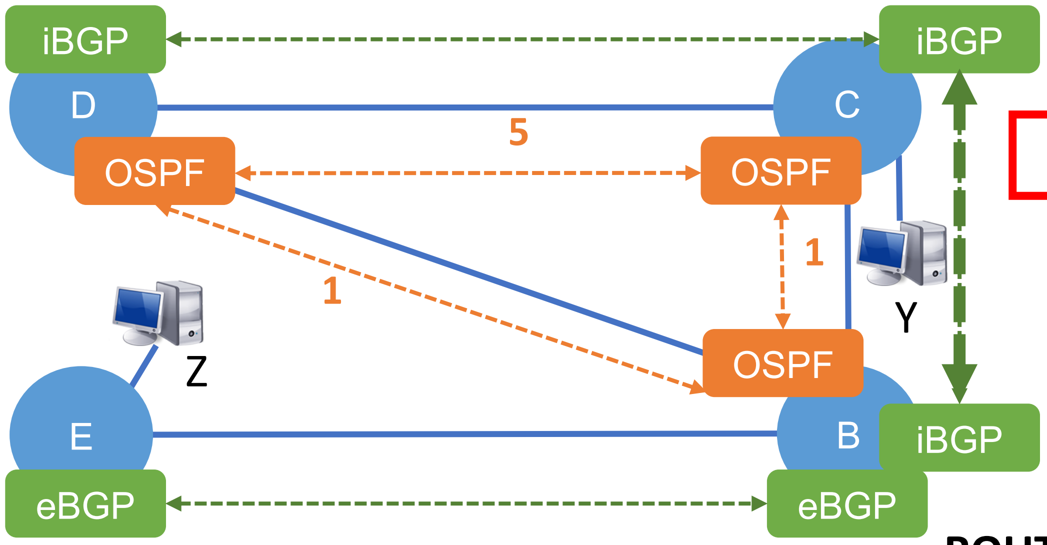
ROUTER B

Dst	NextHop
E	E
C	C
D	D

Cross Layer Dependency

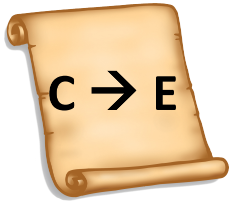
ROUTER D

Dst	NextHop
B	B
C	C



ROUTER C

Dst	NextHop
E	B
B	B
D	D



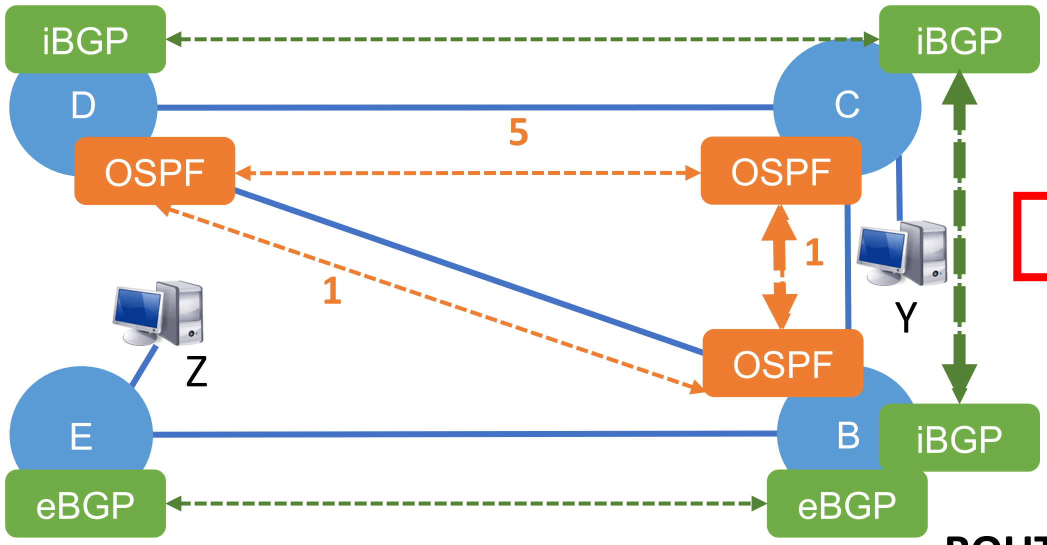
ROUTER B

Dst	NextHop
E	E
C	C
D	D

Cross Layer Dependency

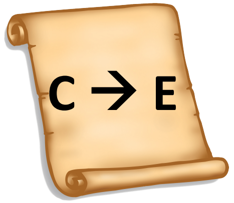
ROUTER D

Dst	NextHop
B	B
C	C



ROUTER C

Dst	NextHop
E	B
B	B
D	D



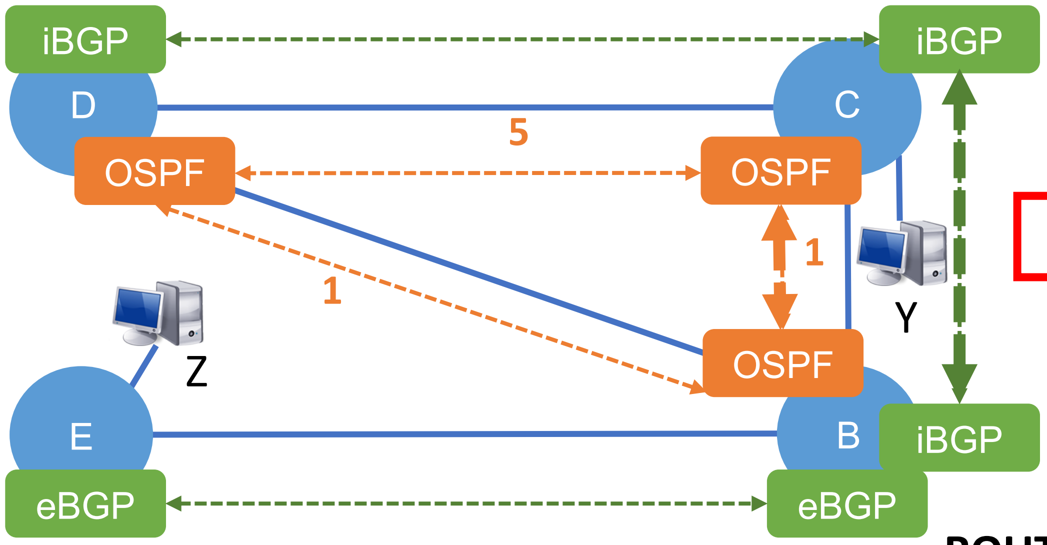
ROUTER B

Dst	NextHop
E	E
C	C
D	D

Cross Layer Dependency

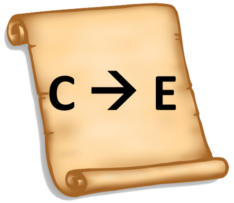
ROUTER D

Dst	NextHop
B	B
C	C



ROUTER C

Dst	NextHop
E	B
B	B
D	D



ROUTER B

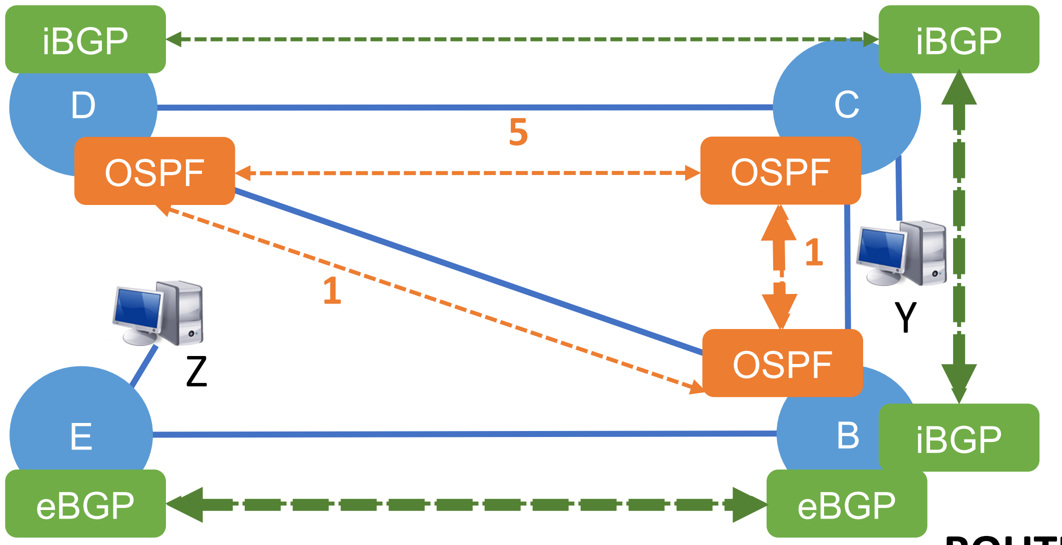
Dst	NextHop
E	E
C	C
D	D

Cross Layer Dependency:
iBGP - OSPF

Cross Layer Dependency

ROUTER D

Dst	NextHop
B	B
C	C

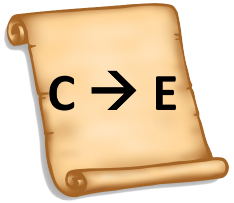


ROUTER C

Dst	NextHop
E	B
B	B
D	D

ROUTER B

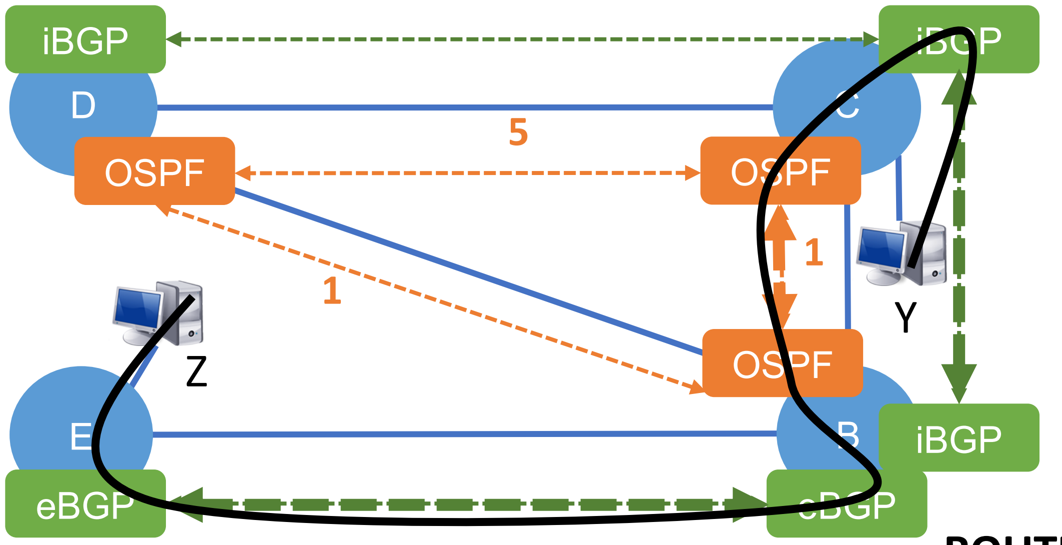
Dst	NextHop
E	E
C	C
D	D



Cross Layer Dependency

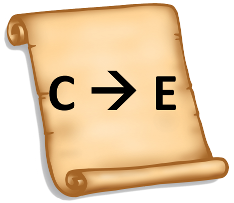
ROUTER D

Dst	NextHop
B	B
C	C



ROUTER C

Dst	NextHop
E	B
B	B
D	D



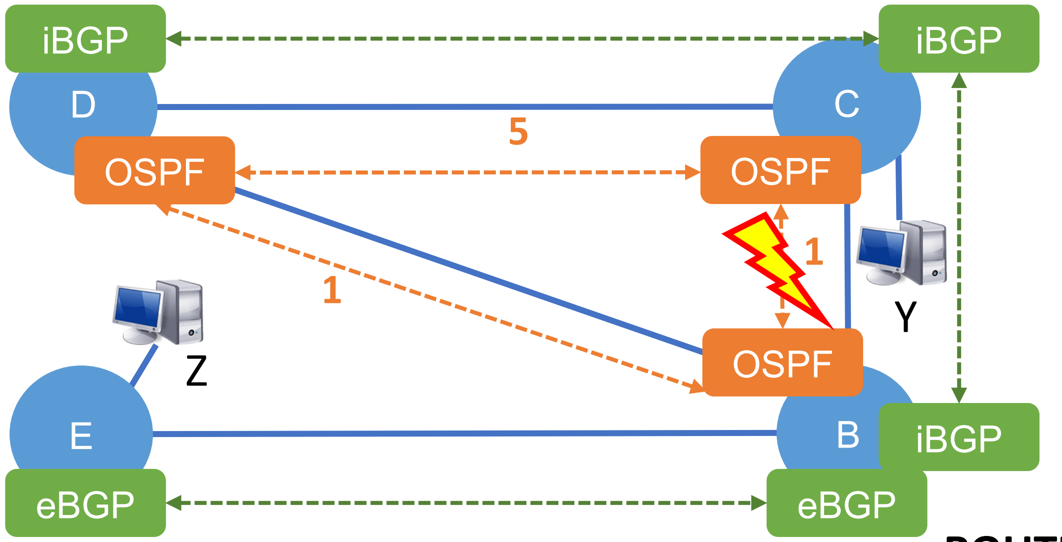
ROUTER B

Dst	NextHop
E	E
C	C
D	D

Cross Layer Dependency

ROUTER D

Dst	NextHop
B	B
C	C



ROUTER C

Dst	NextHop
E	B
B	B
D	D

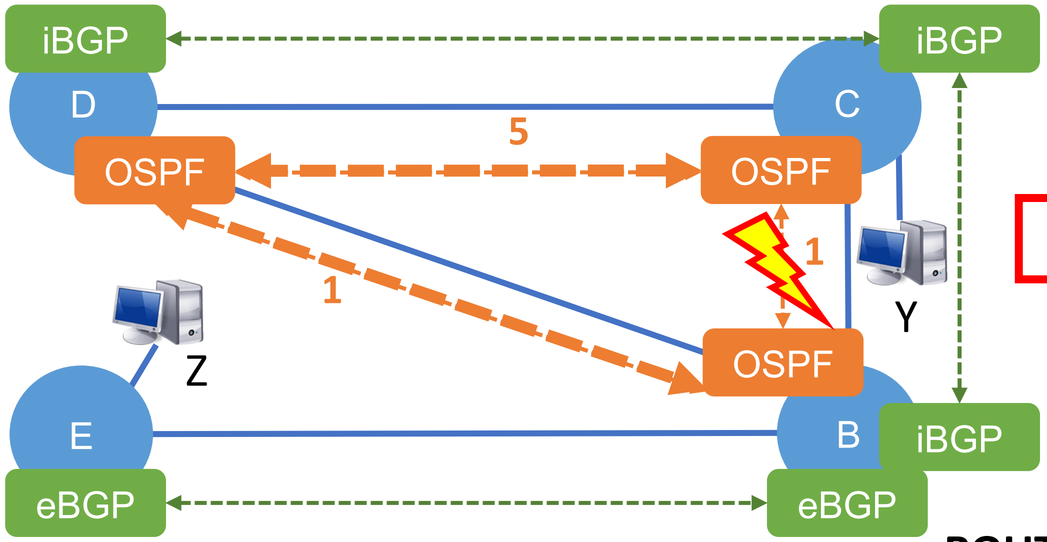
ROUTER B

Dst	NextHop
E	E
C	C
D	D

Cross Layer Dependency

ROUTER D

Dst	NextHop
B	B
C	C



ROUTER C

Dst	NextHop
E	B
B	D
D	D

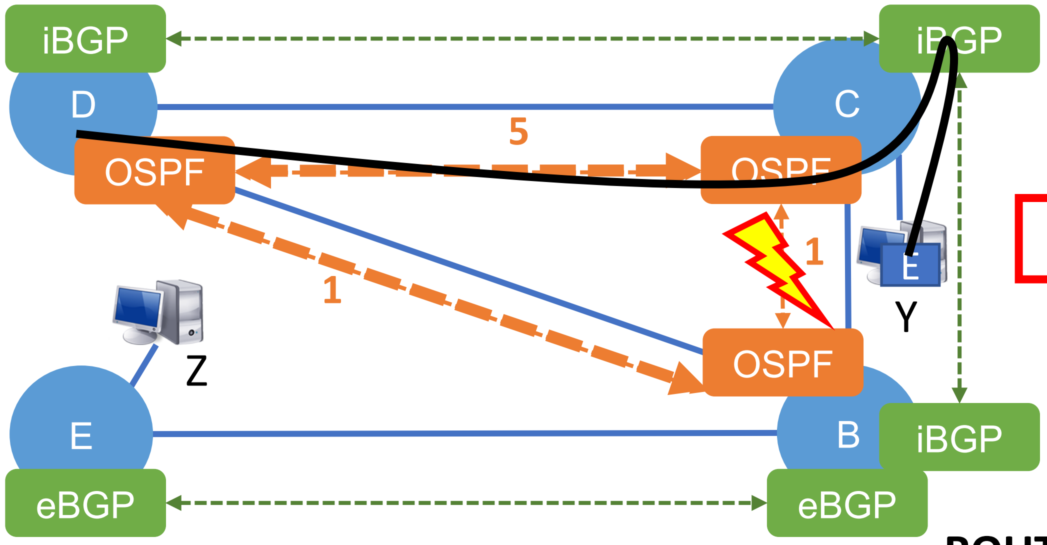
ROUTER B

Dst	NextHop
E	E
C	C
D	D

Cross Layer Dependency

ROUTER D

Dst	NextHop
B	B
C	C



ROUTER C

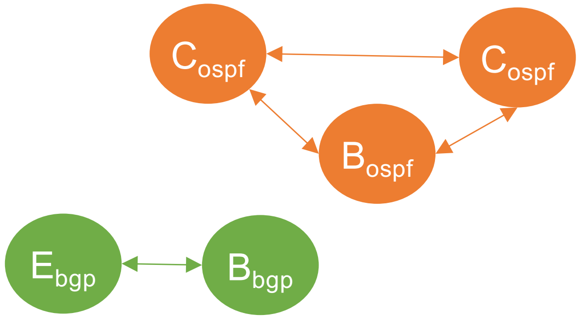
Dst	NextHop
E	B
B	D
D	D

ROUTER B

Dst	NextHop
E	E
C	C
D	D

ARC and Minesweeper

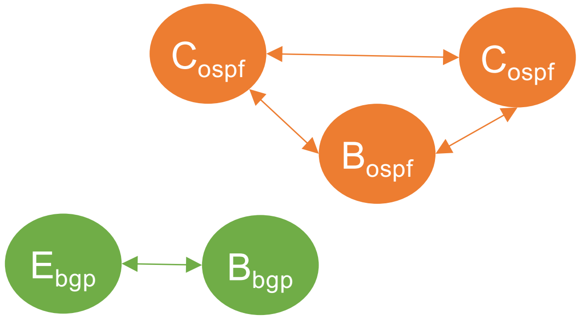
- ARC (SIGCOMM'16)



- Insufficient feature coverage
  - No IBGP, local preference, community, ….

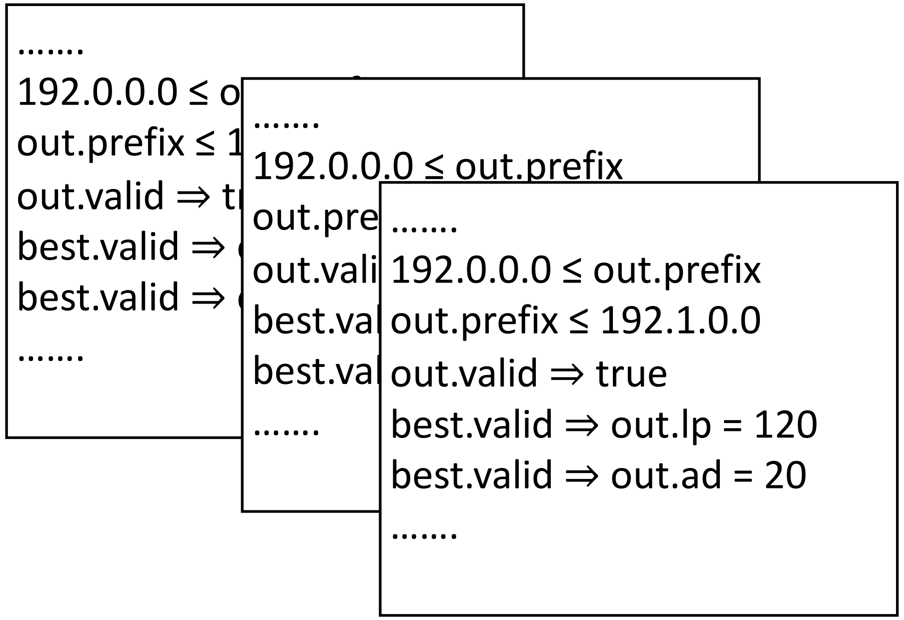
ARC and Minesweeper

- ARC (SIGCOMM'16)



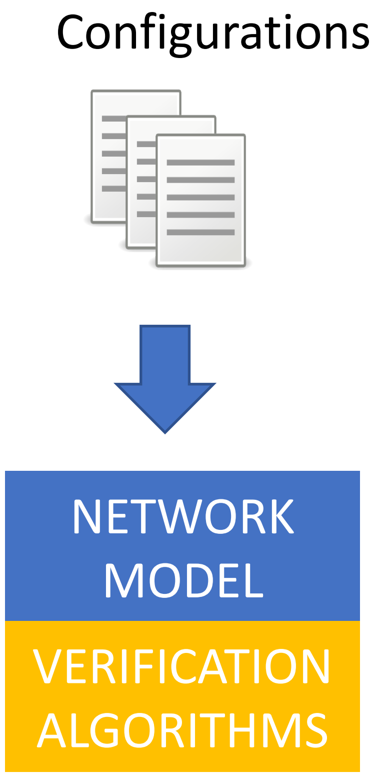
  - Insufficient feature coverage
    - No IBGP, local preference, community, ….

- Minesweeper (SIGCOMM'17)

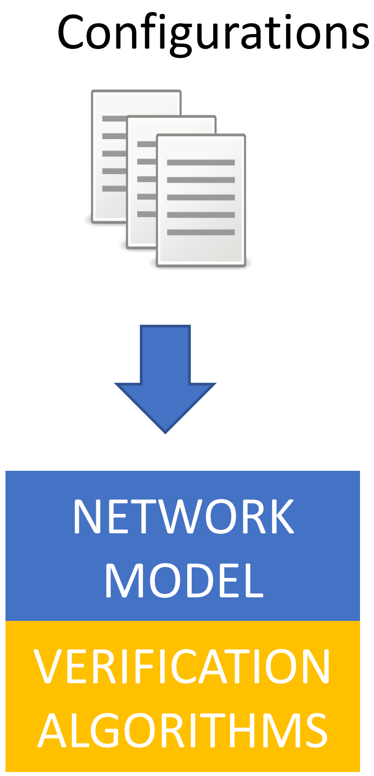


  - Poor performance
    - replicate model for iBGP

Insights



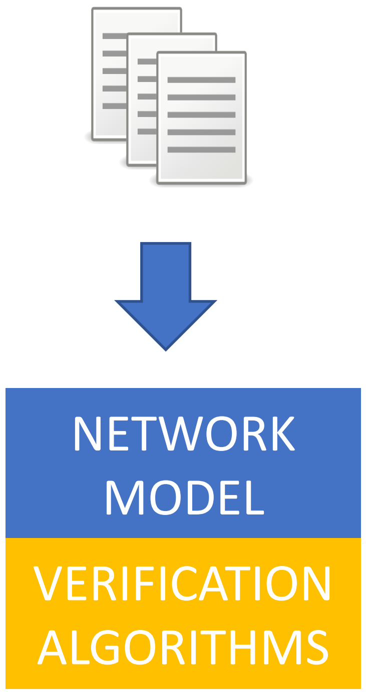
Insights



	Model	**Verification**
ARC	Graph	Graph
Minesweeper	SMT	SMT

Insights

Configurations



	Model	**Verification**
ARC	Graph	Graph
Minesweeper	SMT	SMT

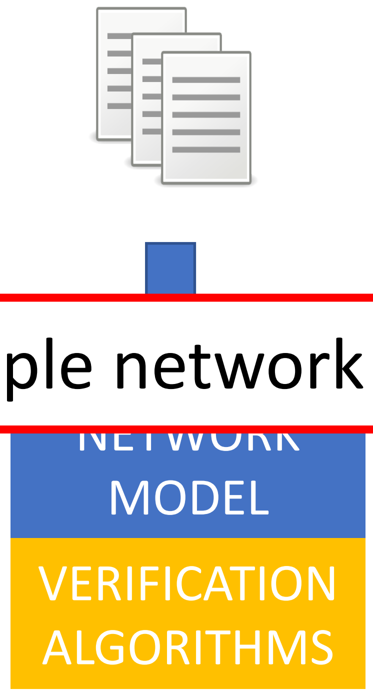
low coverage

high coverage

high performance

low performance

Insights

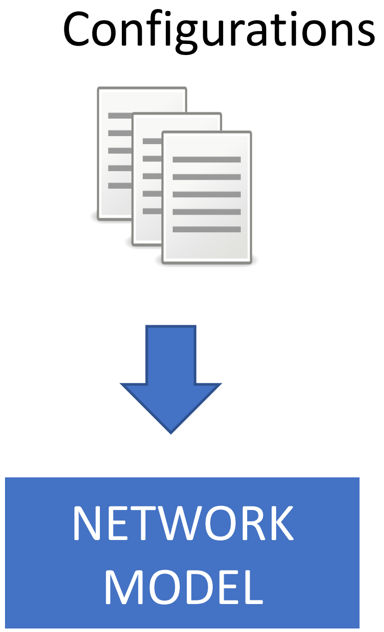


Configurations	Model	Verification
ARC	Graph	Graph
Minesweeper	SMT	SMT

low coverage (ARC model) · high coverage (Minesweeper model) · high performance (Graph verification) · low performance (SMT verification)

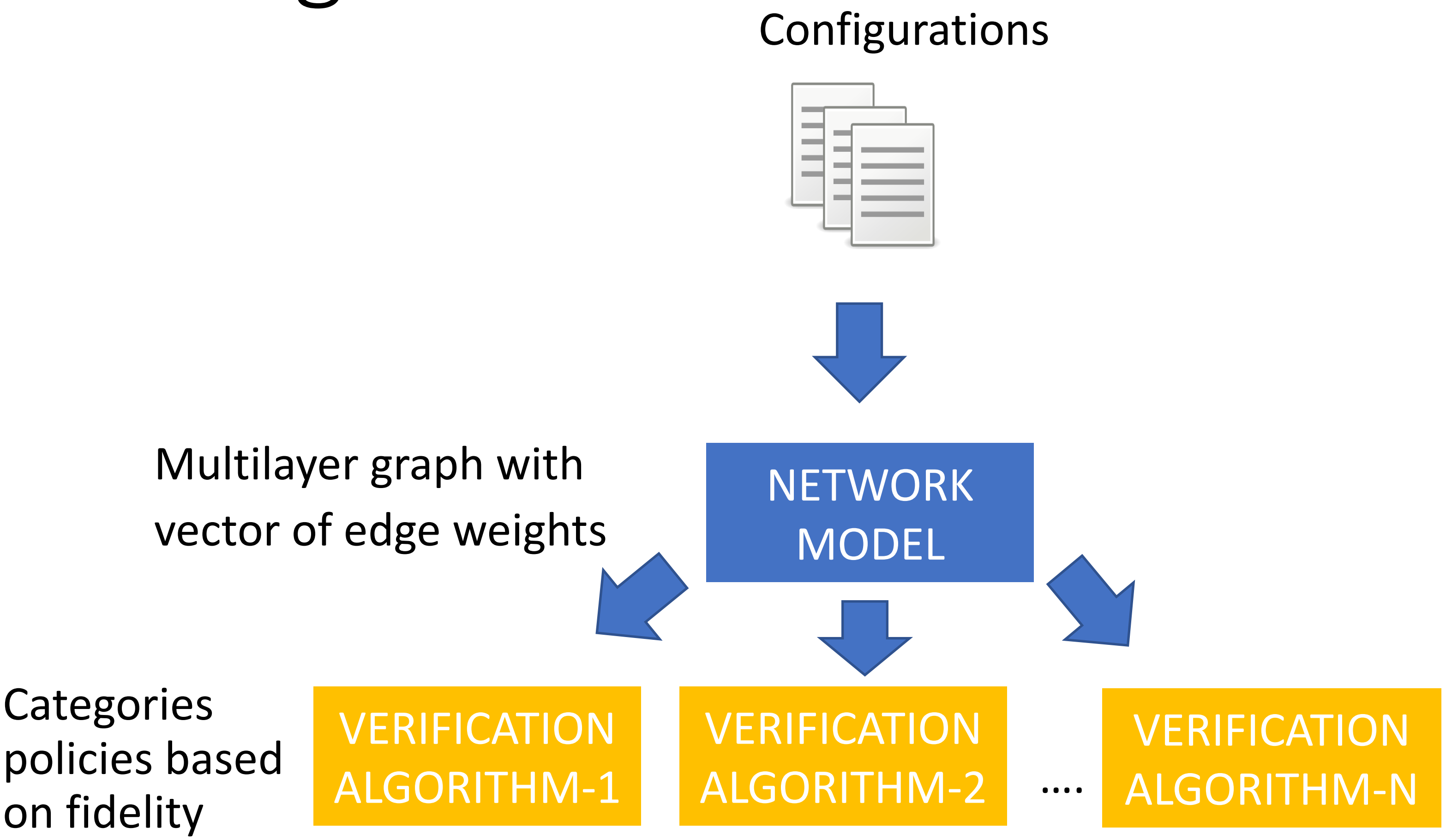
Insight 1: Decouple network encoding from verification algorithm

Insights



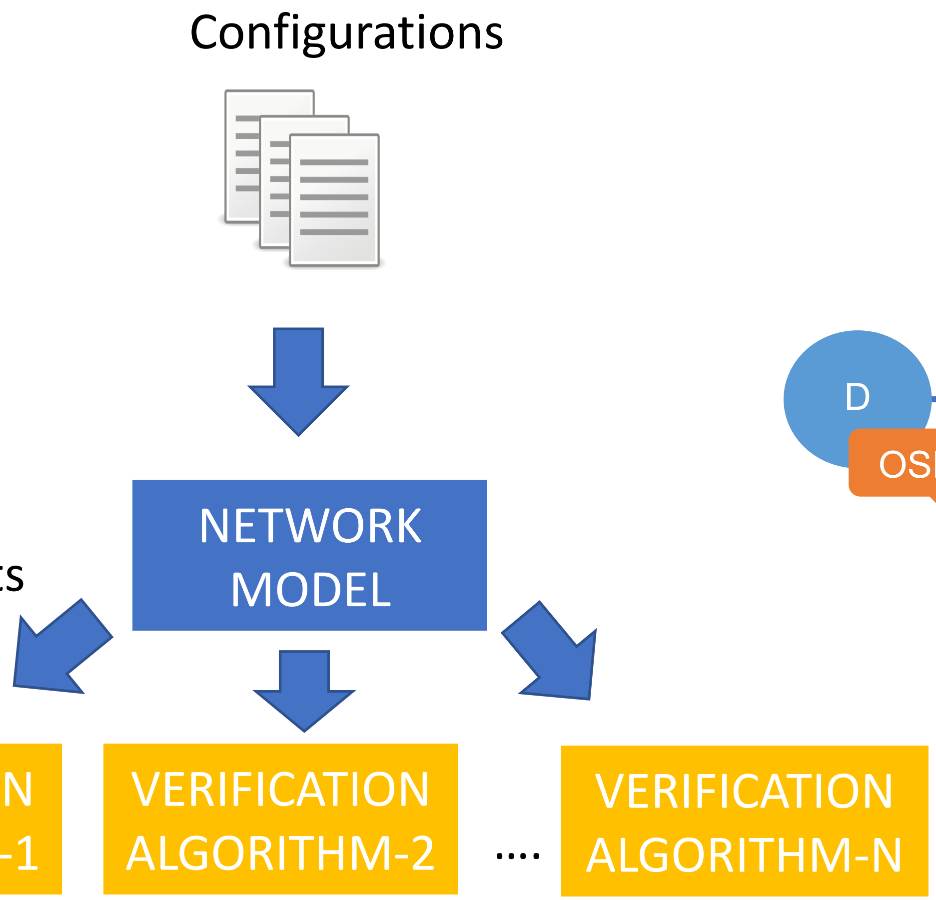
Multilayer graph with vector of edge weights

Insights



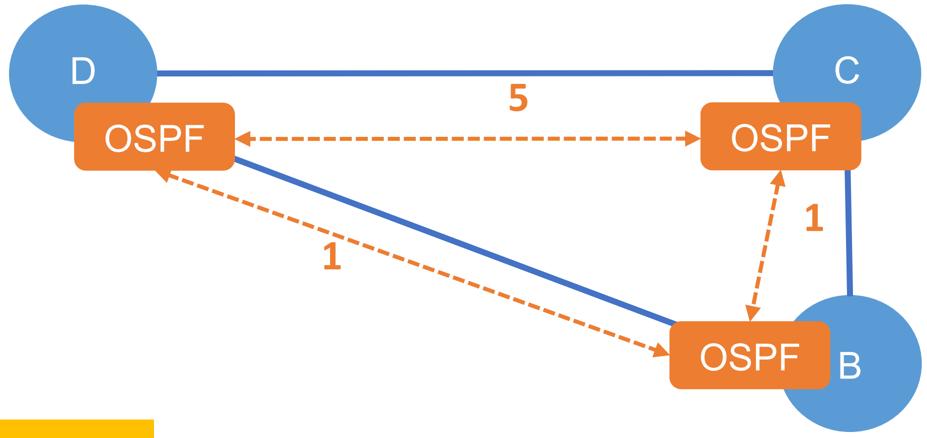
Configurations
Multilayer graph with vector of edge weights
NETWORK MODEL
Categories policies based on fidelity
VERIFICATION ALGORITHM-1
VERIFICATION ALGORITHM-2
….
VERIFICATION ALGORITHM-N

Insights

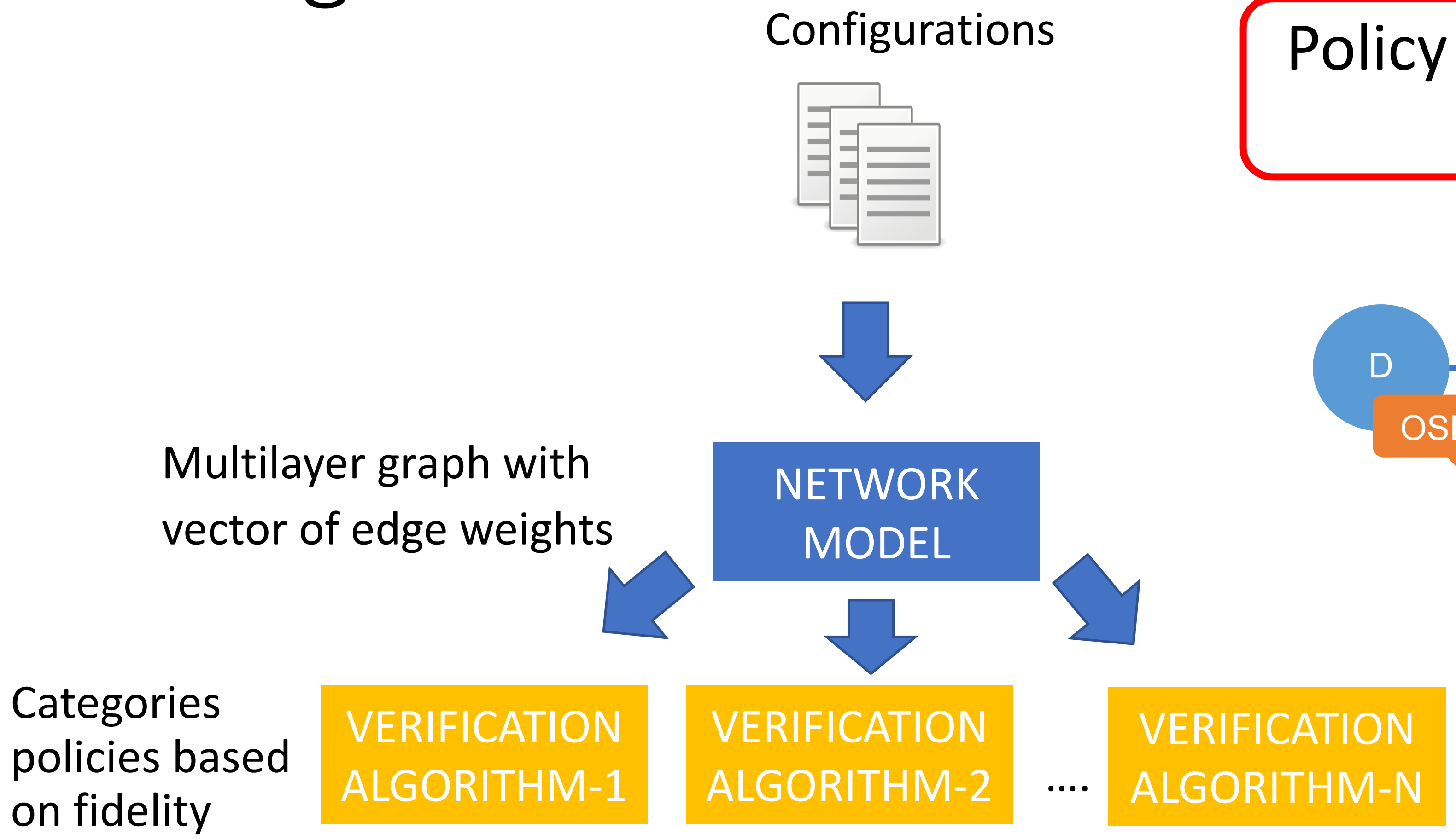


Multilayer graph with vector of edge weights

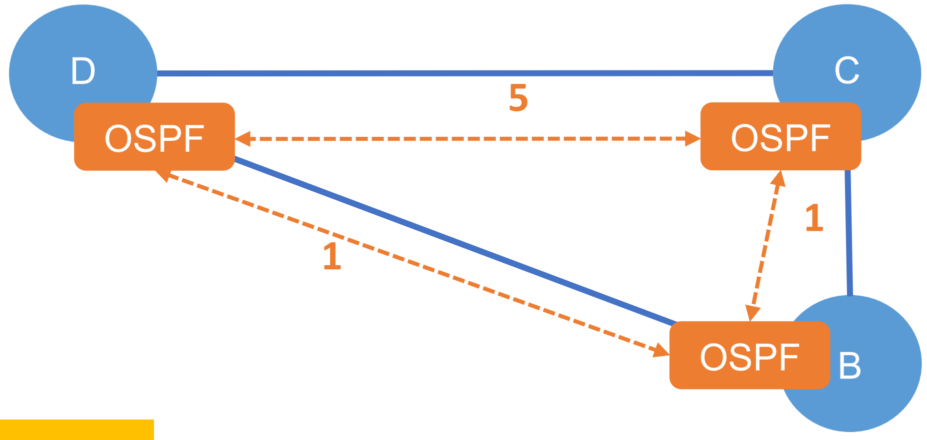
Categories policies based on fidelity



Insights



Policy 1: Which path is preferred?
C→B << C→D→B



Insights

Configurations

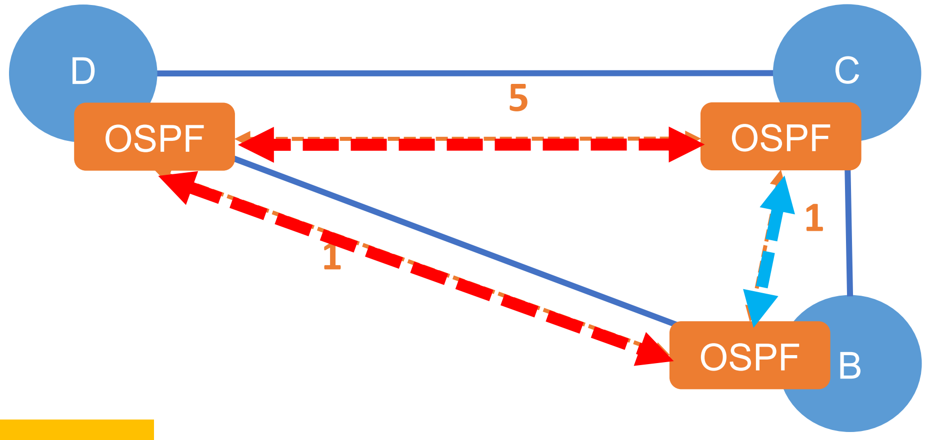
Multilayer graph with vector of edge weights

NETWORK MODEL

Categories policies based on fidelity

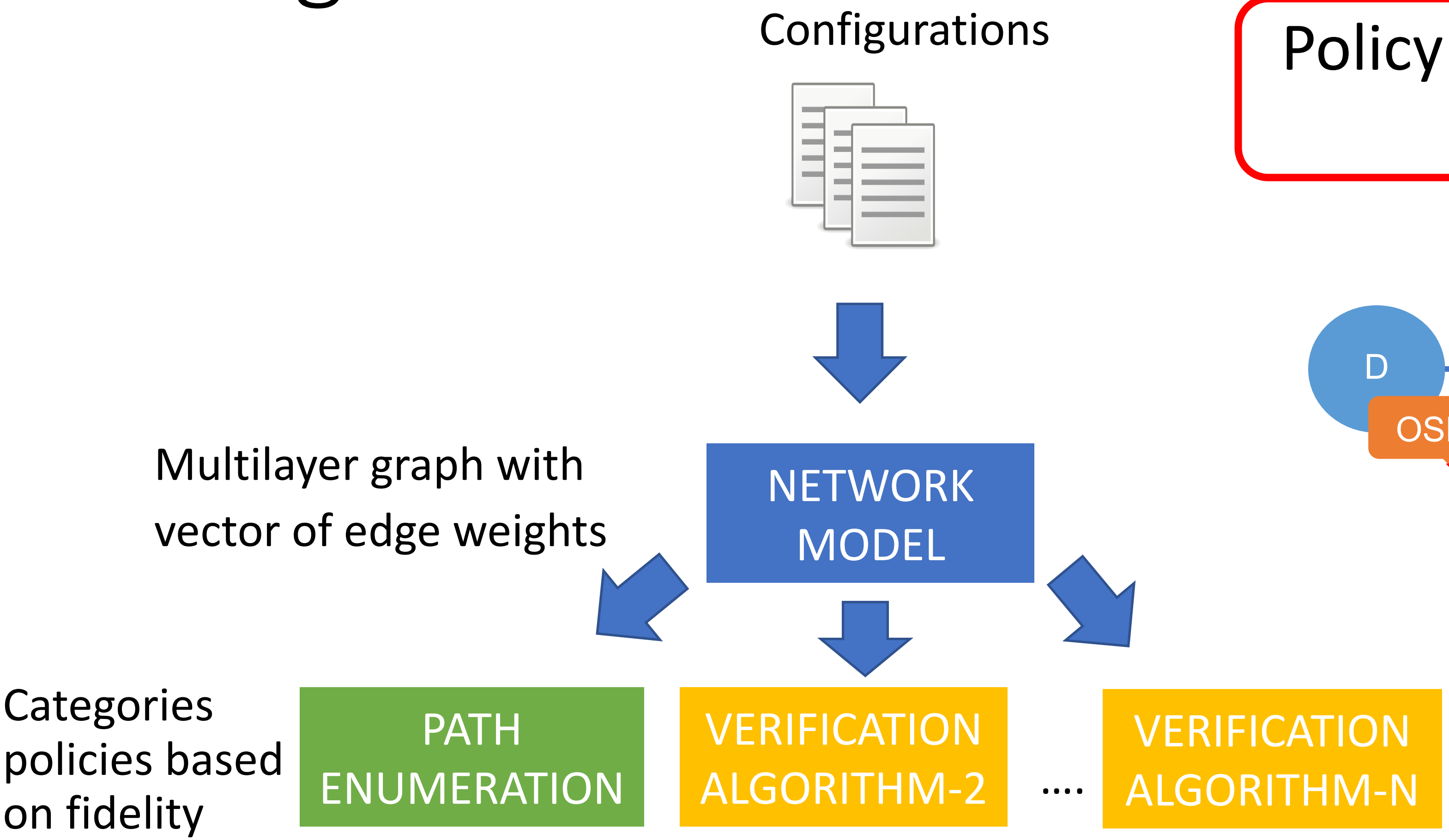
VERIFICATION ALGORITHM-1 | VERIFICATION ALGORITHM-2 | .... | VERIFICATION ALGORITHM-N

Policy 1: Which path is preferred?
C→B << C→D→B

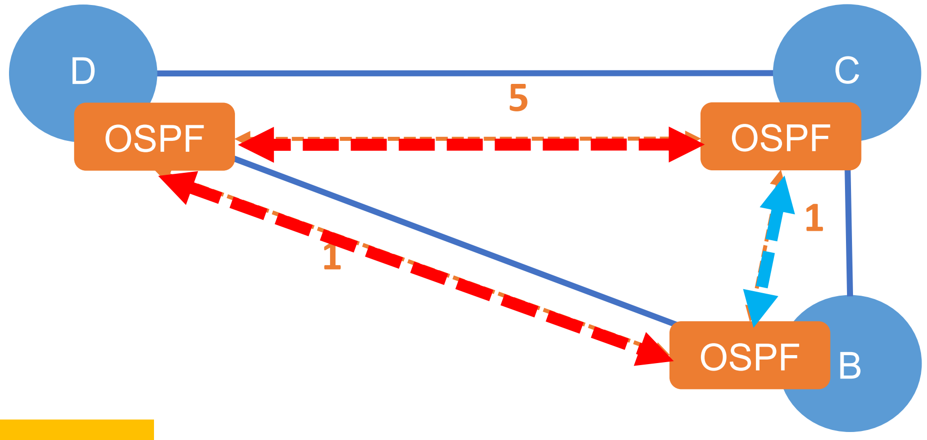


C→B : ospf cost = 1
C→D→B : ospf cost = 6

Insights



Policy 1: Which path is preferred?
C→B << C→D→B



C→B : ospf cost = 1
C→D→B : ospf cost = 6

Insights

Configurations

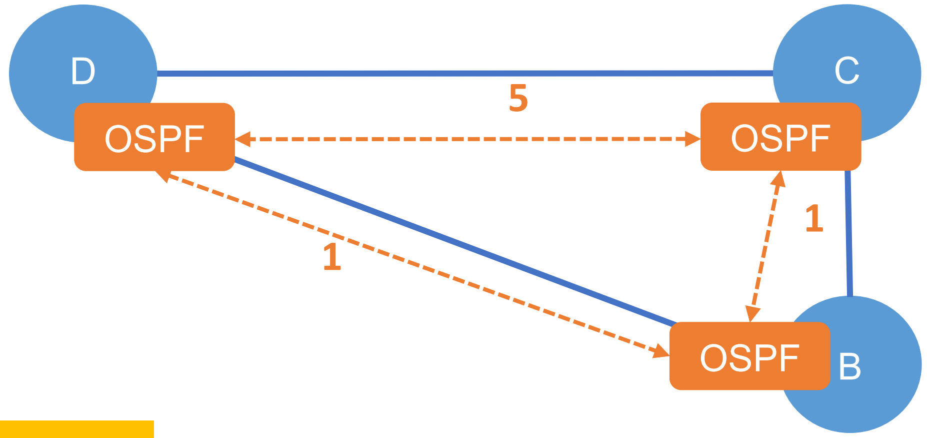
Multilayer graph with vector of edge weights

NETWORK MODEL

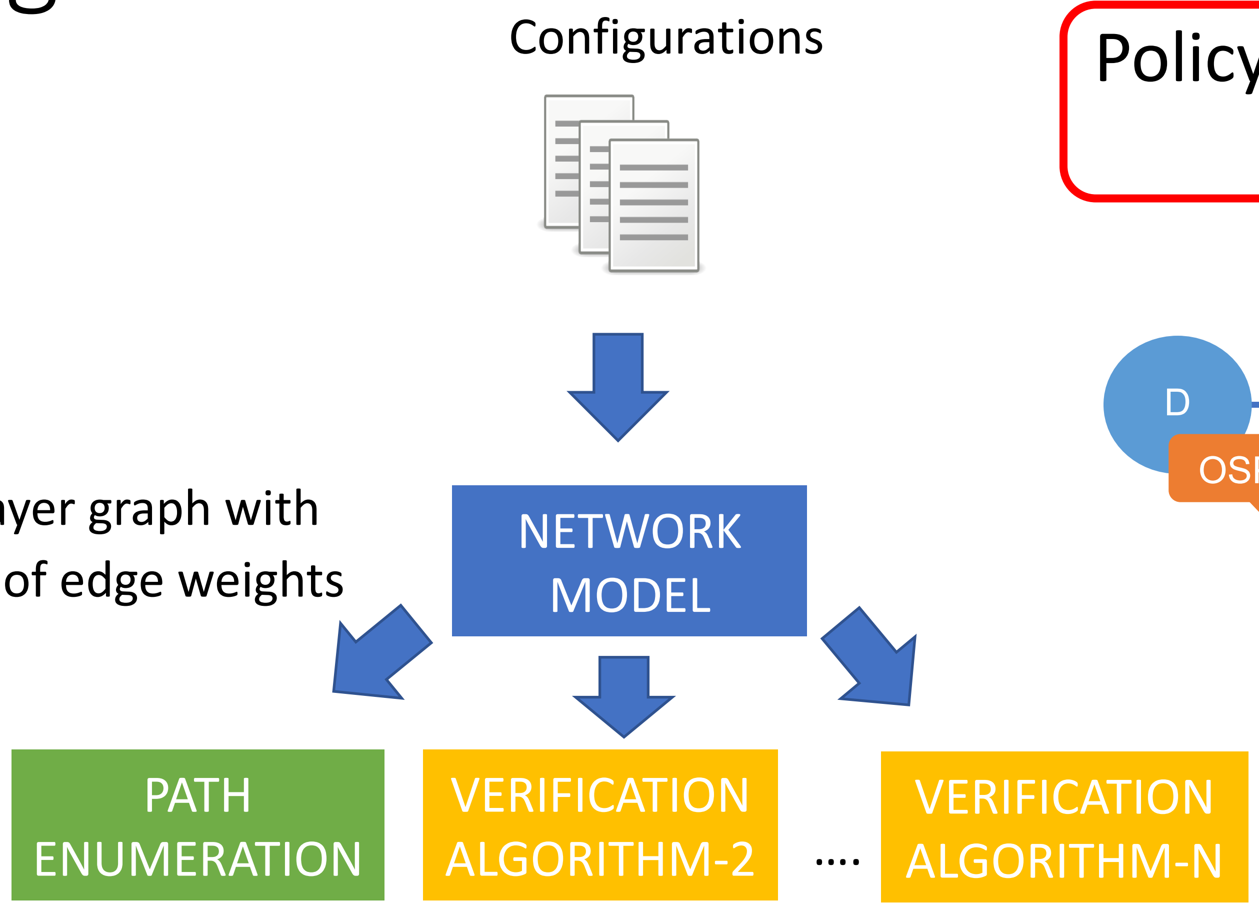
Categories policies based on fidelity

PATH ENUMERATION | VERIFICATION ALGORITHM-2 | .... | VERIFICATION ALGORITHM-N

Policy 2: Can C reach B with 1 link failure?



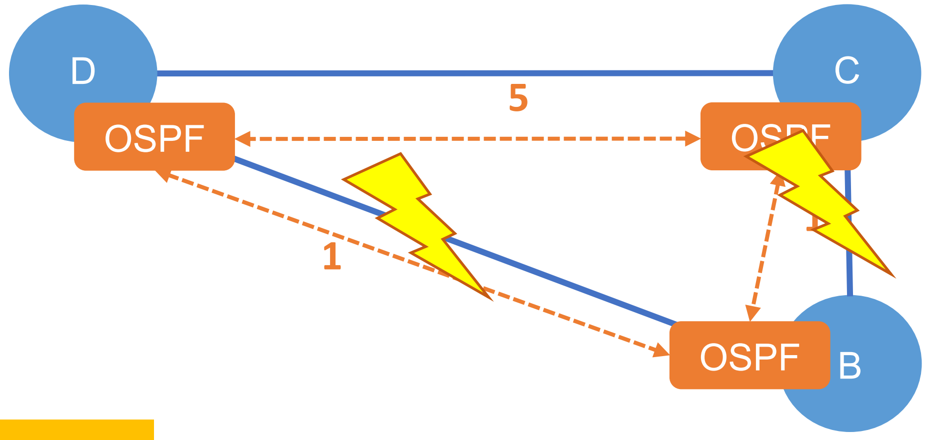
Insights



Multilayer graph with vector of edge weights

Categories policies based on fidelity

Policy 2: Can C reach B with 1 link failure?



Insights

Configurations

Multilayer graph with vector of edge weights

NETWORK MODEL

Categories policies based on fidelity

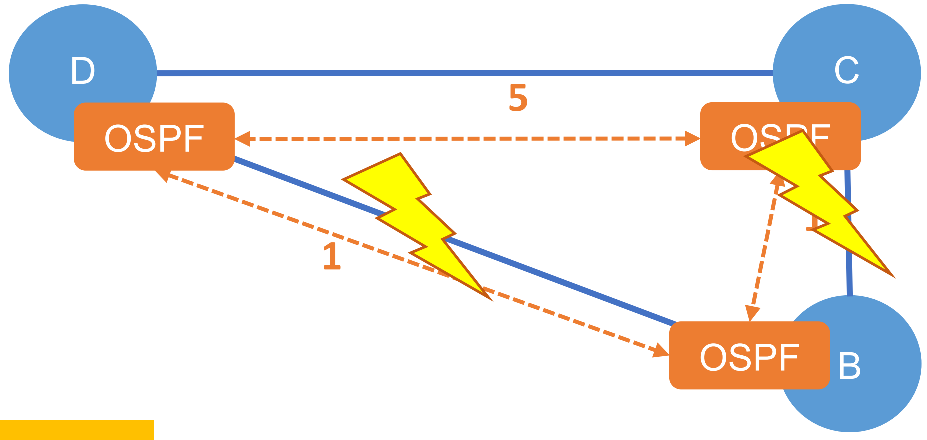
PATH ENUMERATION

VERIFICATION ALGORITHM-2

....

VERIFICATION ALGORITHM-N

Policy 2: Can C reach B with 1 link failure?



Min-cut = 2

Insights

Configurations

Multilayer graph with vector of edge weights

NETWORK MODEL

Categories policies based on fidelity

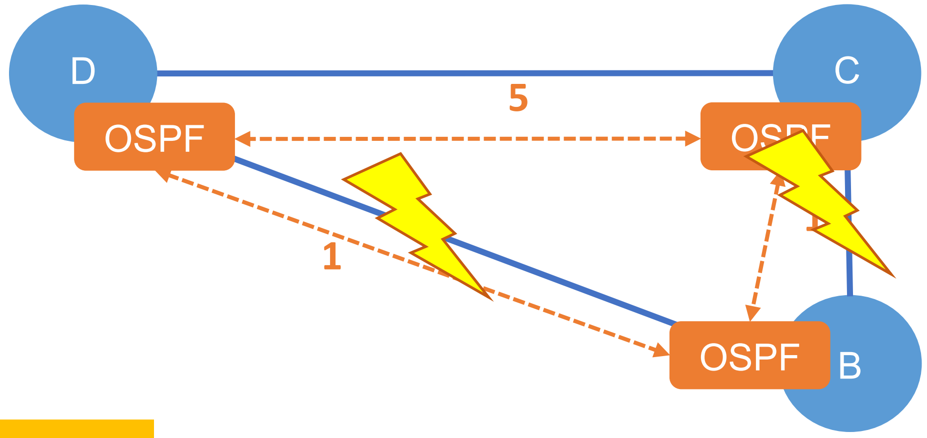
PATH ENUMERATION

QUANTITATIVE GRAPH PROPERTY

....

VERIFICATION ALGORITHM-N

Policy 2: Can C reach B with 1 link failure?



Min-cut = 2

Insights

Configurations

NETWORK MODEL

Multilayer graph with vector of edge weights

Categories policies based on fidelity

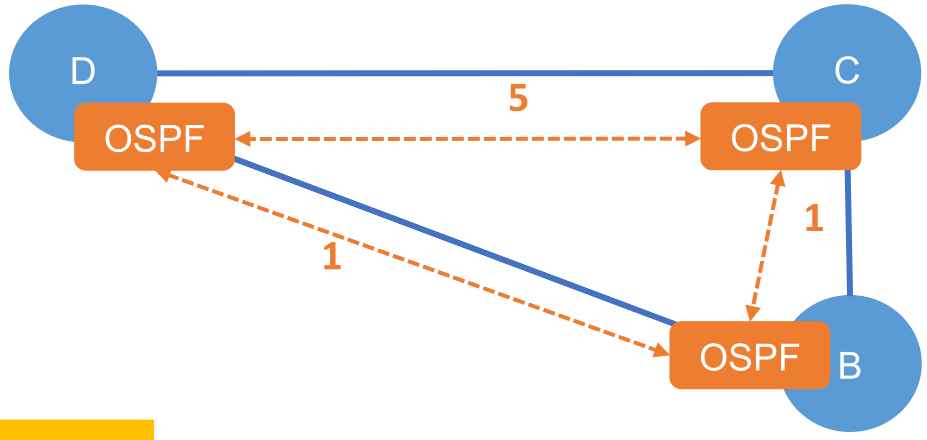
PATH ENUMERATION

QUANTITATIVE GRAPH PROPERTY

....

VERIFICATION ALGORITHM-N

Policy 3: Is C always unreachable/blocked from B?



Insights

Configurations

Multilayer graph with vector of edge weights

NETWORK MODEL

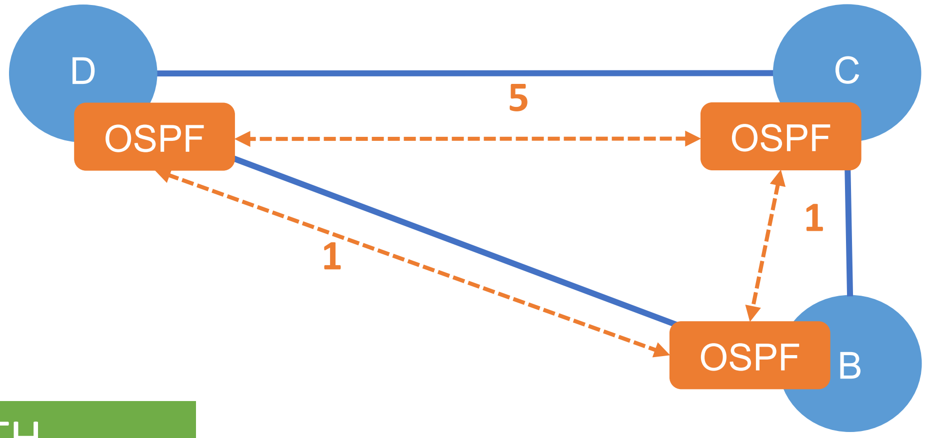
Categories policies based on fidelity

PATH ENUMERATION

QUANTITATIVE GRAPH PROPERTY

PATH EXISTENCE (CONNECTIVITY)

Policy 3: Is C always unreachable/blocked from B?



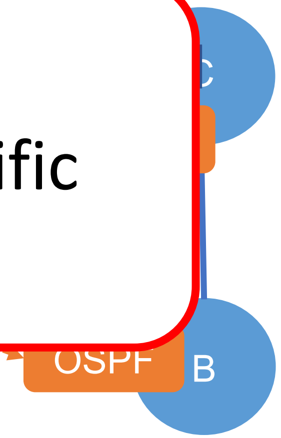
Insights

Configurations

Multi[…]

vecto[…]

Insight 2: Different properties require different levels of fidelity modeling of the control plane. Use property-specific algorithm for performance benefits



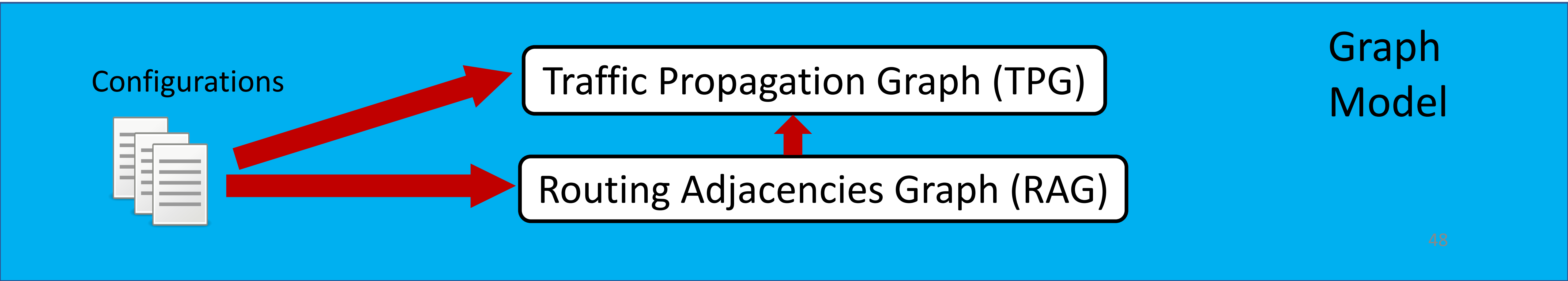
Categories policies based on fidelity

PATH ENUMERATION	QUANTITATIVE GRAPH PROPERTY	PATH EXISTENCE (CONNECTIVITY)

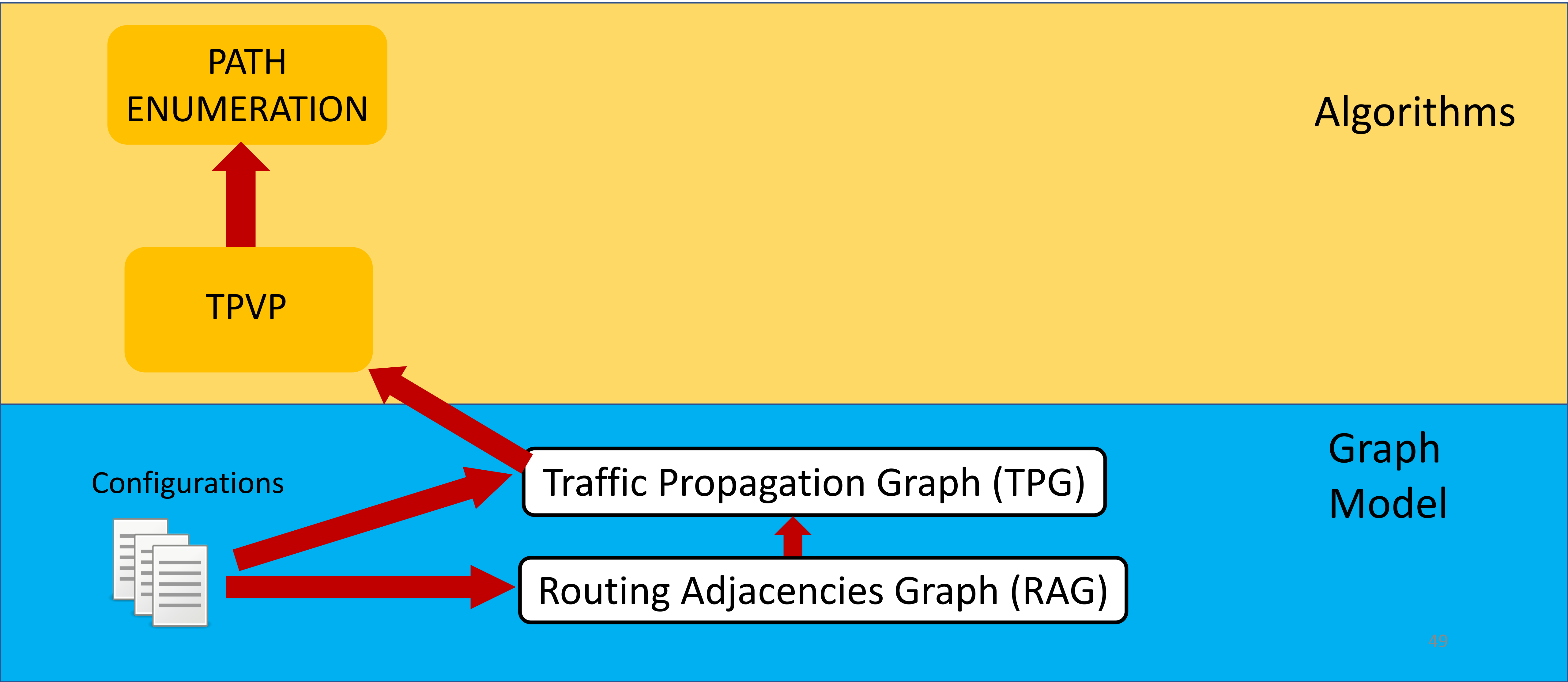
low performance
high fidelity

high performance
low fidelity

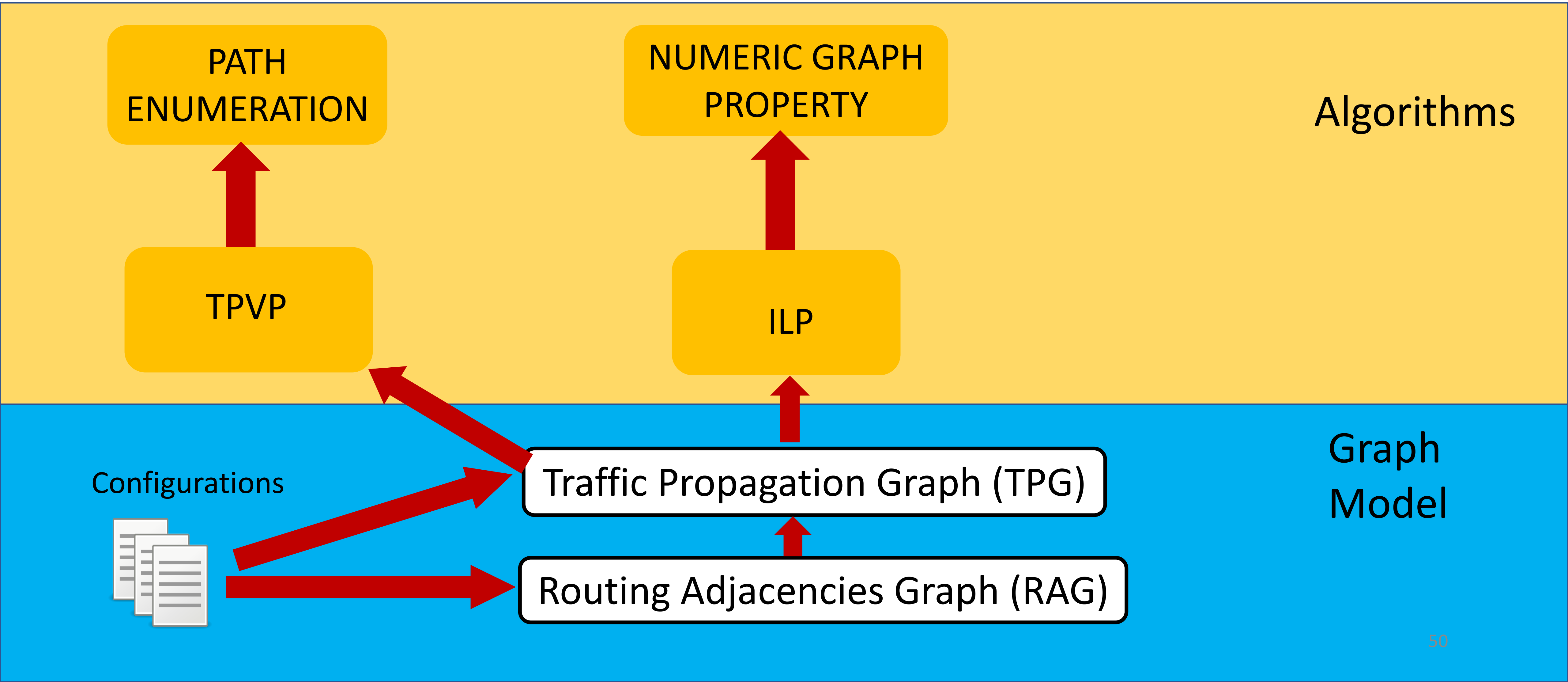
Tiramisu Overview



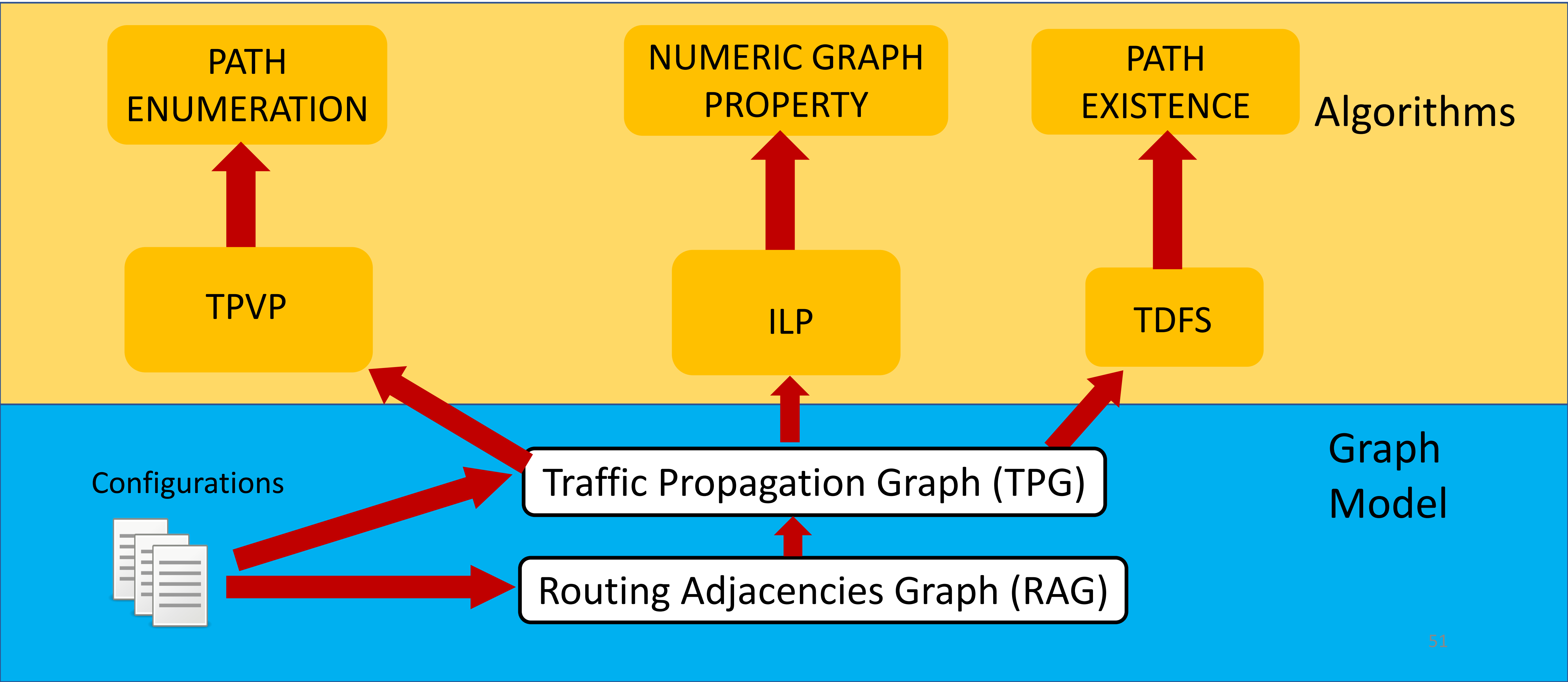
Tiramisu Overview



Tiramisu Overview



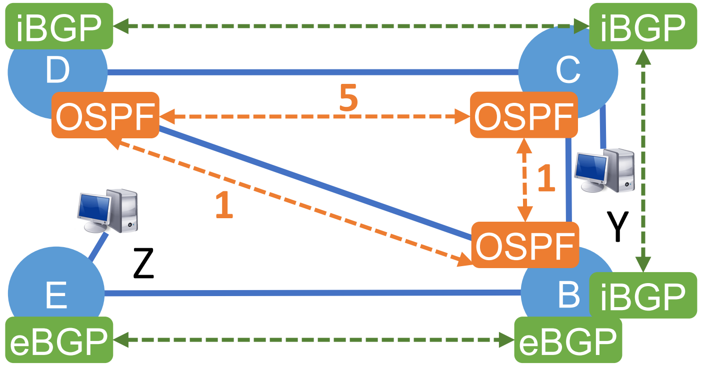
Tiramisu Overview



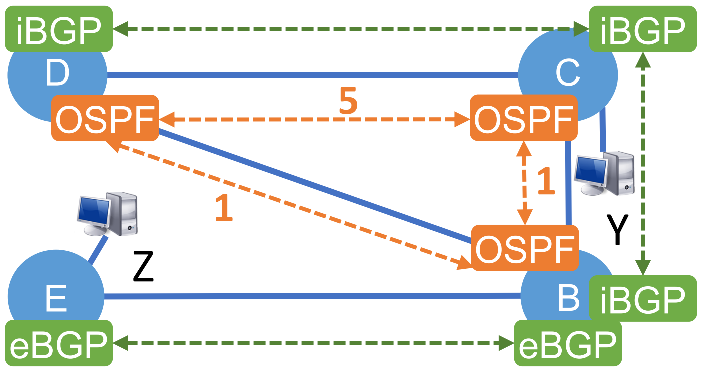
Tiramisu Overview

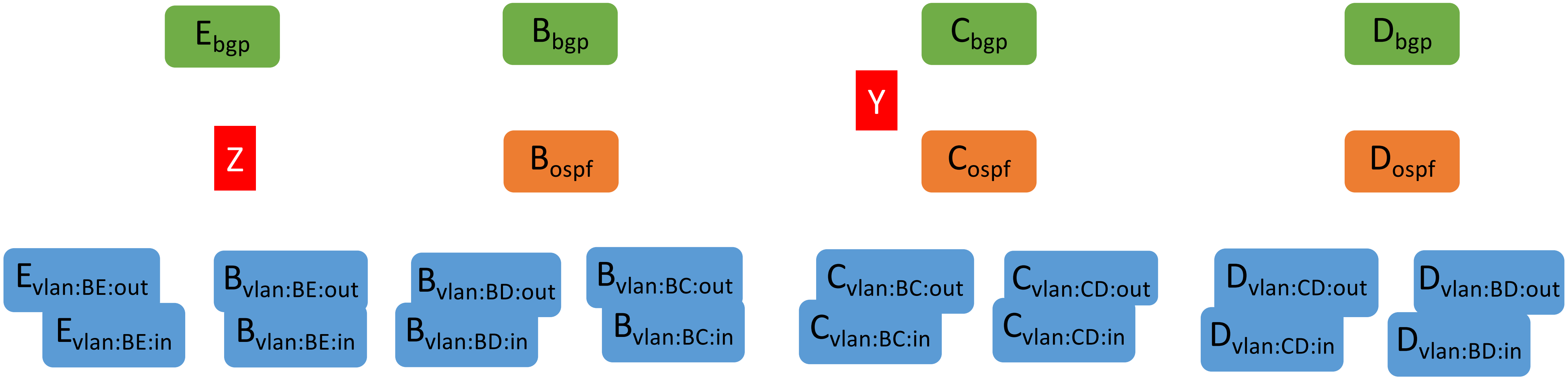
Graph
Model

Traffic Propagation Graph (TPG)



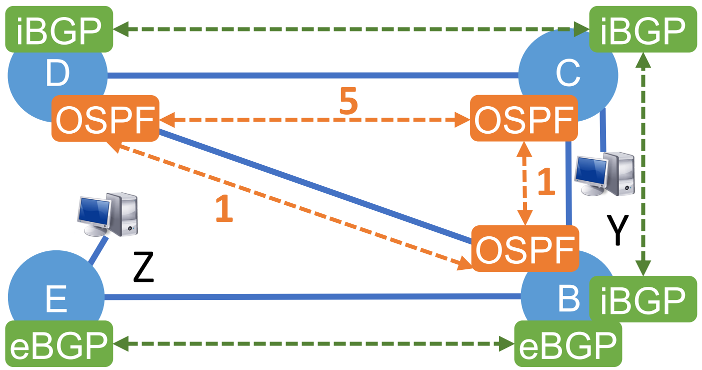
Traffic Propagation Graph (TPG)

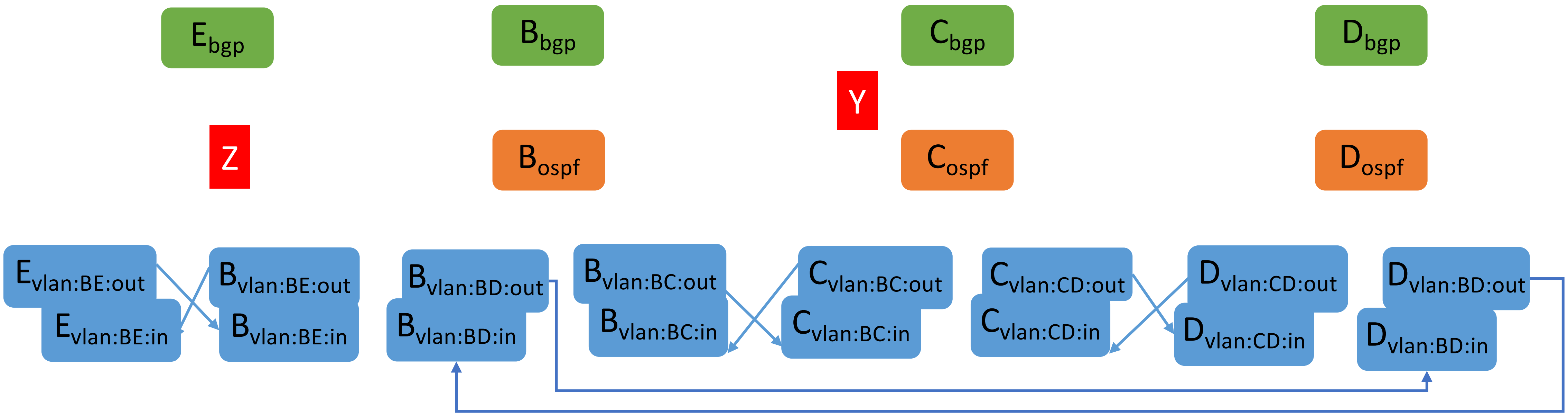




- **Vertices**: RIB of a routing processes and ingress/egress point of a switch/router

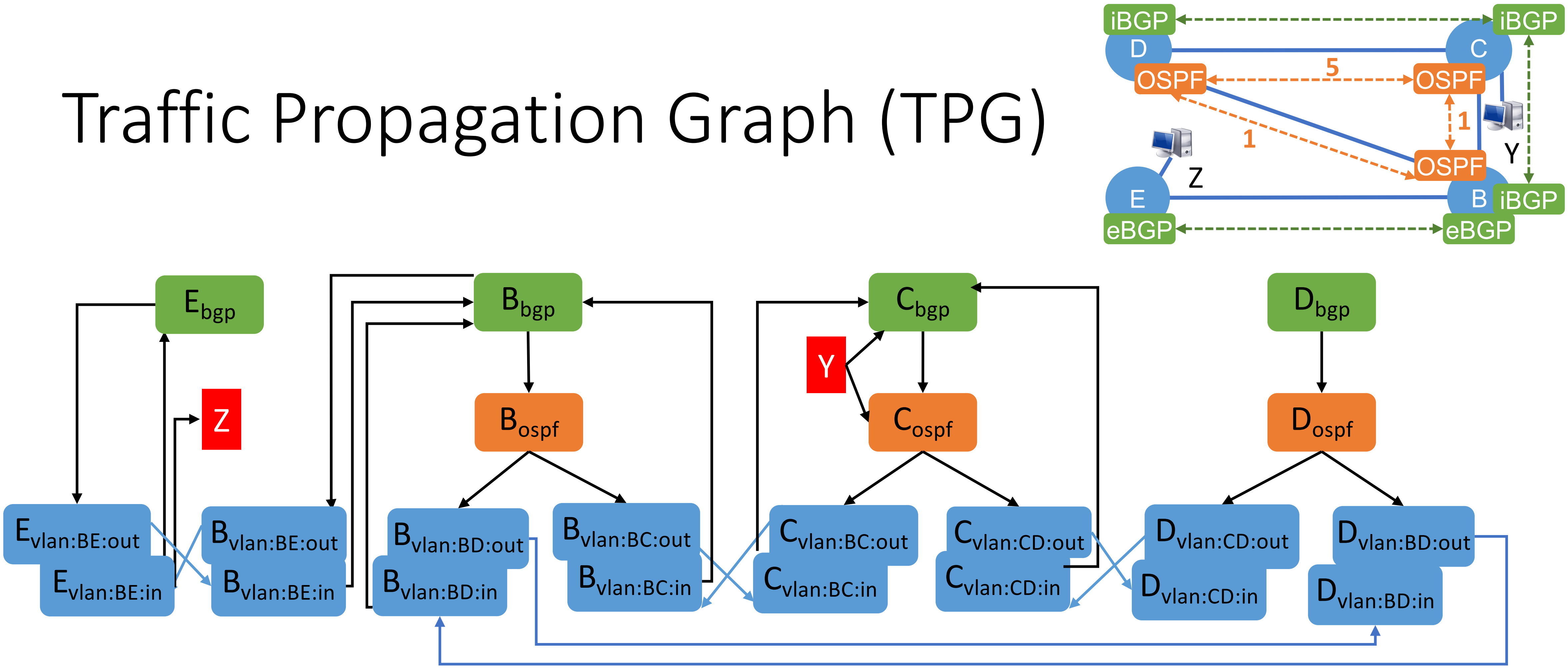
Traffic Propagation Graph (TPG)





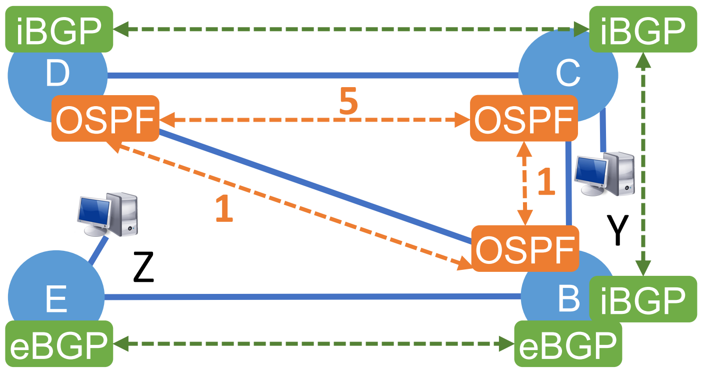
- **Vertices**: RIB of a routing processes and ingress/egress point of a switch/router
- **Edges**: establish route dependency and traffic flow

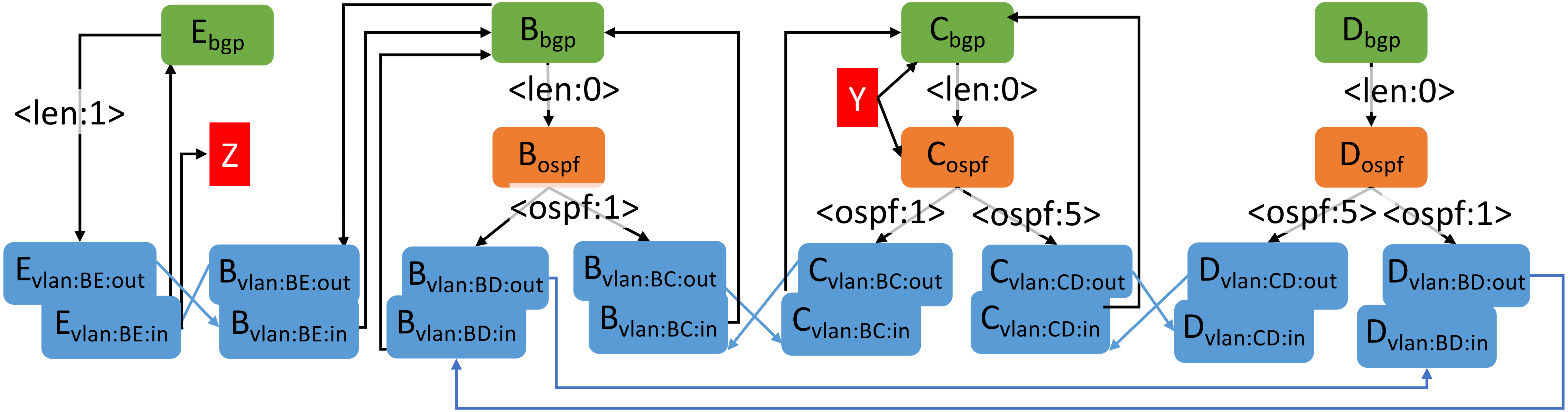
Traffic Propagation Graph (TPG)



- **Vertices**: RIB of a routing processes and ingress/egress point of a switch/router
- **Edges**: establish route dependency and traffic flow

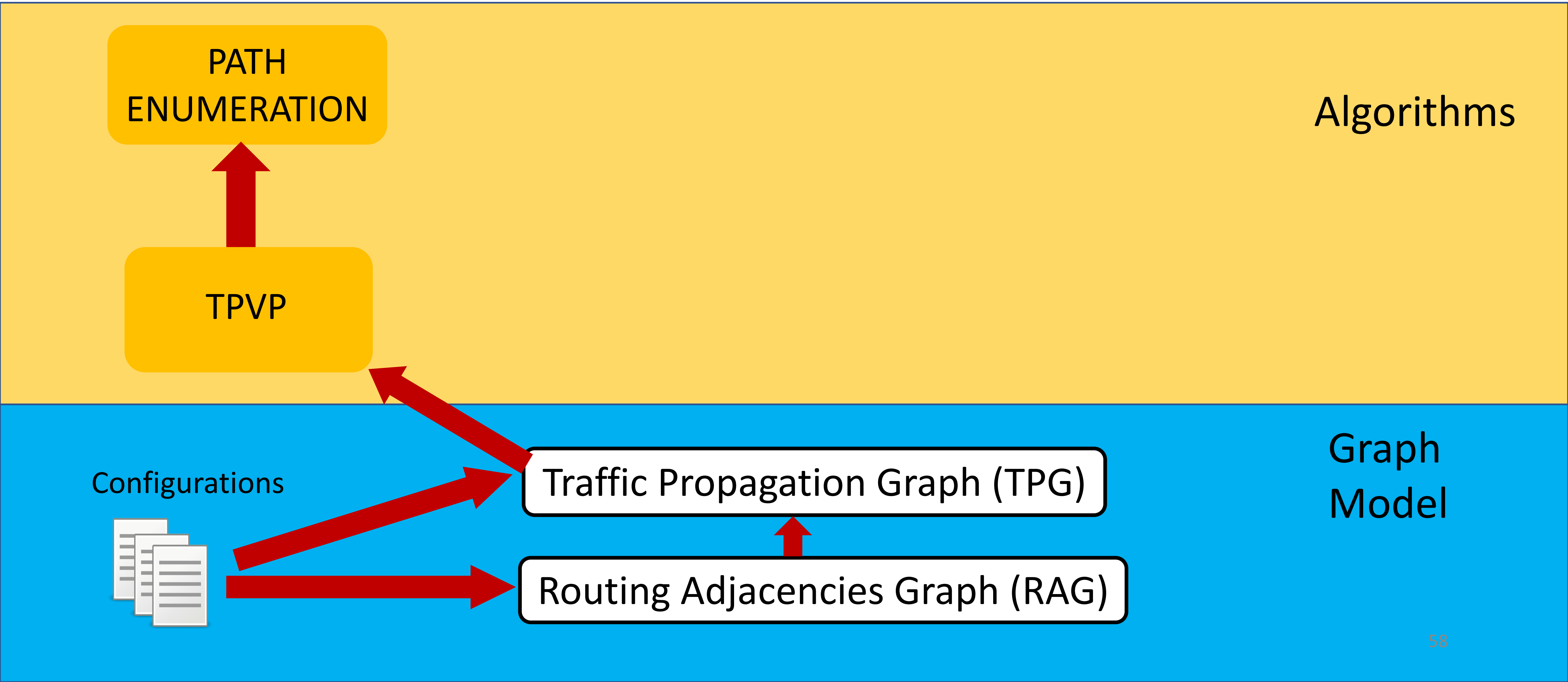
Traffic Propagation Graph (TPG)





- **Vertices**: RIB of a routing processes and ingress/egress point of a switch/router
- **Edges**: establish route dependency and traffic flow
- **Vector of edge weights:** multiple route metrics

Tiramisu Overview

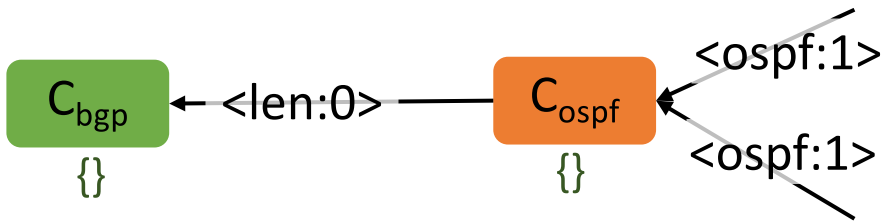


TPVP - Tiramisu Path Vector Protocol

- Griffin et al. (TON'2002) and Sobrinho (TON'2005) models stable paths problem as Simple Path Vector Protocol (SPVP) and routing algebra
- TPVP is derived from SPVP and is modeled on routing algebra
  - $\bigoplus$ operator to model path cost computation
  - $\preceq$ operator to model preference relation and path selection

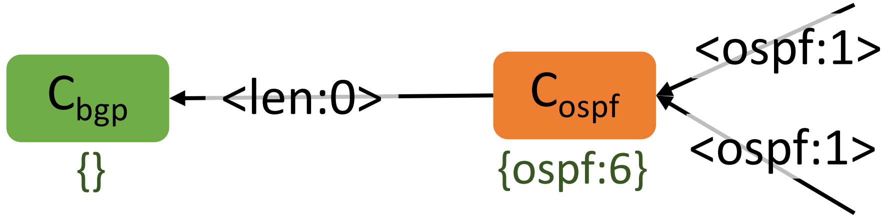
TPVP - Tiramisu Path Vector Protocol

- Griffin et al. (TON'2002) and Sobrinho (TON'2005) models stable paths problem as Simple Path Vector Protocol (SPVP) and routing algebra
- TPVP is derived from SPVP and is modeled on routing algebra
  - $\bigoplus$ operator to model path cost computation
  - $\preceq$ operator to model preference relation and path selection



TPVP - Tiramisu Path Vector Protocol

- Griffin et al. (TON’2002) and Sobrinho (TON’2005) models stable paths problem as Simple Path Vector Protocol (SPVP) and routing algebra
- TPVP is derived from SPVP and is modeled on routing algebra
  - $\oplus$ operator to model path cost computation
  - $\preceq$ operator to model preference relation and path selection

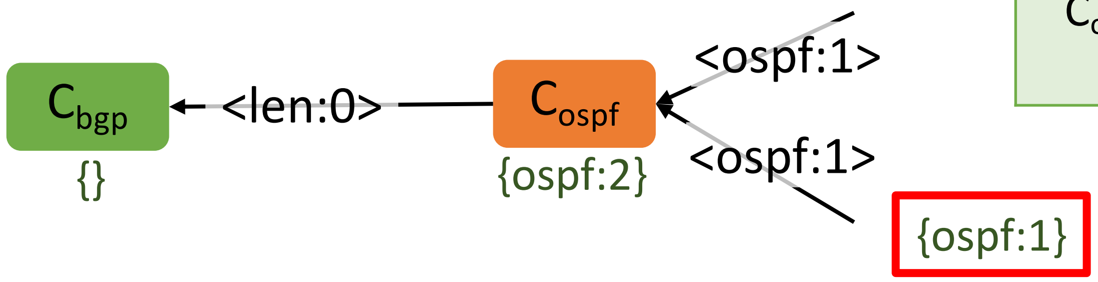


{ospf:5}

Node	$\oplus$	$\preceq$
C_{ospf}	{ospf:1} $\oplus_{ospf}$ {ospf:5} = {ospf:6}	{ospf:6} $\preceq_{ospf}$ {}

TPVP - Tiramisu Path Vector Protocol

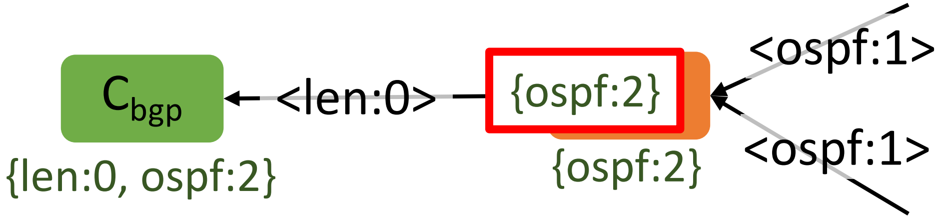
- Griffin et al. (TON’2002) and Sobrinho (TON’2005) models stable paths problem as Simple Path Vector Protocol (SPVP) and routing algebra
- TPVP is derived from SPVP and is modeled on routing algebra
  - $\oplus$ operator to model path cost computation
  - $\preceq$ operator to model preference relation and path selection



Node	$\oplus$	$\preceq$
C_{ospf}	$\{ospf:1\} \oplus_{ospf} \{ospf:1\} = \{ospf:2\}$	$\{ospf:2\} \preceq_{ospf} \{ospf:6\}$

TPVP - Tiramisu Path Vector Protocol

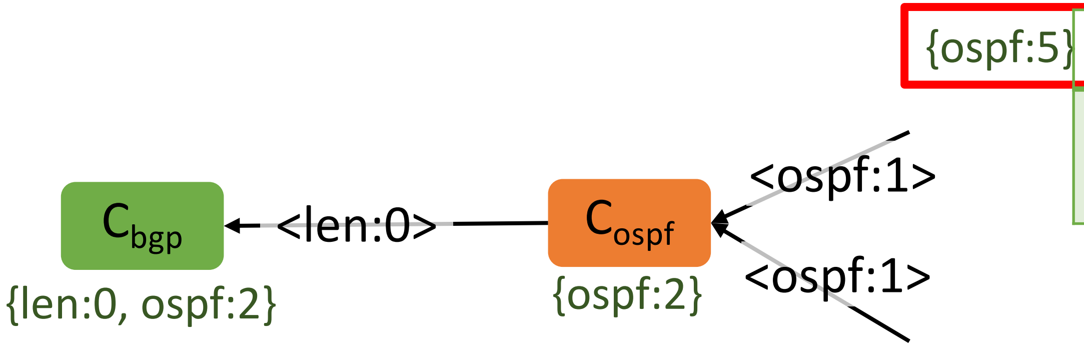
- Griffin et al. (TON’2002) and Sobrinho (TON’2005) models stable paths problem as Simple Path Vector Protocol (SPVP) and routing algebra
- TPVP is derived from SPVP and is modeled on routing algebra
  - $\bigoplus$ operator to model path cost computation
  - $\preceq$ operator to model preference relation and path selection



Node	$\bigoplus$	$\preceq$
C_{bgp}	{len:0} $\bigoplus_{bgp}$ {ospf:1} = {len:0, ospf: 2}	{len:0, ospf:2} $\preceq_{bgp}$ {}

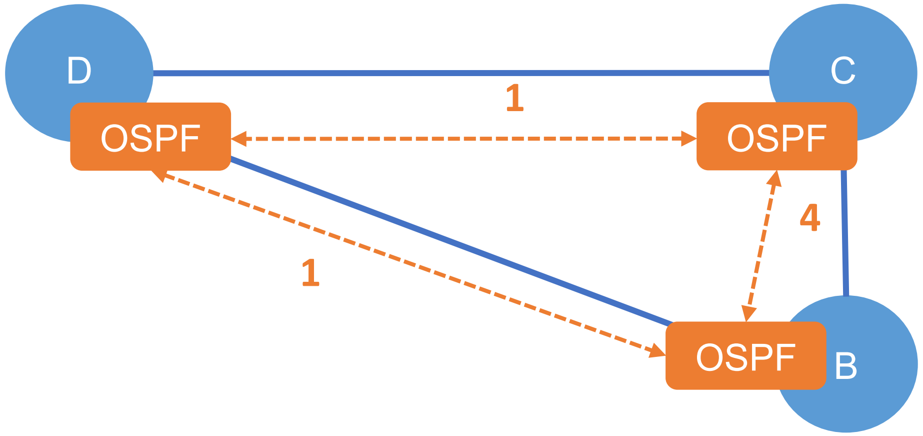
TPVP - Tiramisu Path Vector Protocol

- Griffin et al. (TON’2002) and Sobrinho (TON’2005) models stable paths problem as Simple Path Vector Protocol (SPVP) and routing algebra
- TPVP is derived from SPVP and is modeled on routing algebra
  - $\oplus$ operator to model path cost computation
  - $\preceq$ operator to model preference relation and path selection

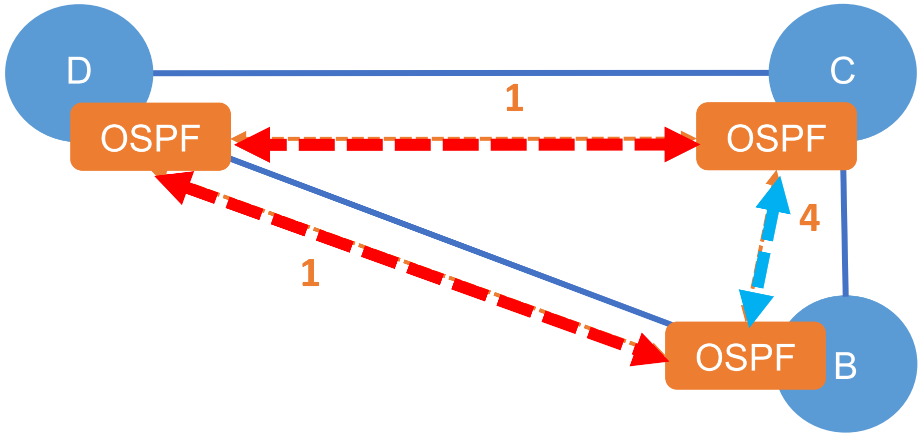


Node	$\oplus$	$\preceq$
C_{ospf}	{ospf:1} $\oplus_{ospf}$ {ospf:5} = {ospf: 6}	{ospf:6} $\not\preceq_{ospf}$ {ospf:2}

TPVP - Tiramisu Path Vector Protocol

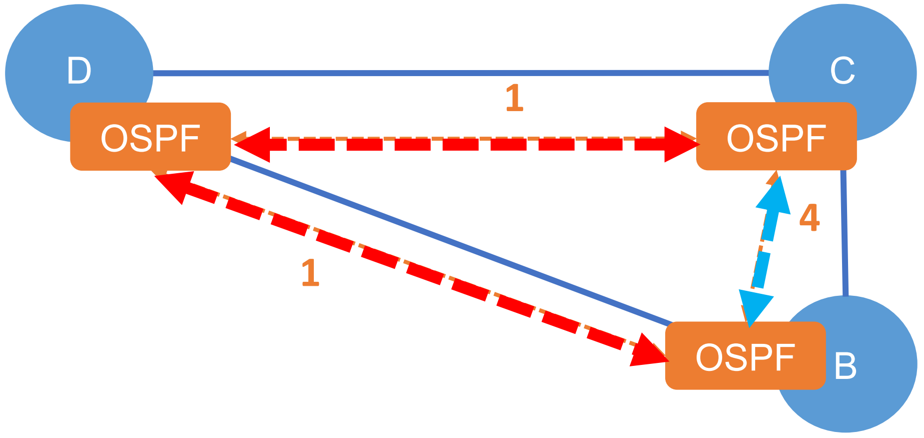


TPVP - Tiramisu Path Vector Protocol



P1: C→D→B : ospf cost = 2
P2: C→B : ospf cost = 4

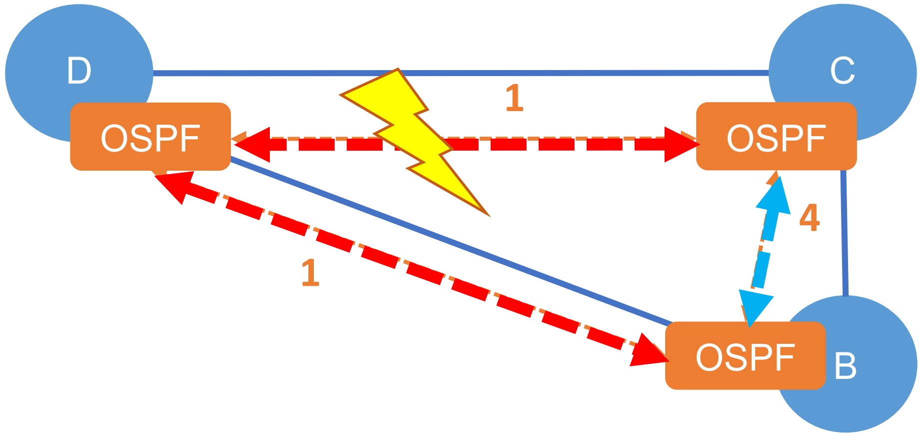
TPVP - Tiramisu Path Vector Protocol



P1: C→D→B : ospf cost = 2

P2: C→B : ospf cost = 4

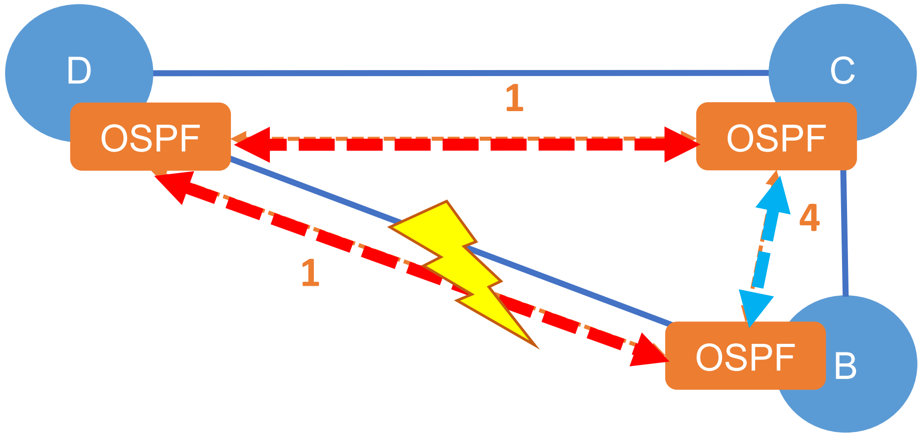
TPVP - Tiramisu Path Vector Protocol



P1: C→D→B : ospf cost = 2

P2: C→B : ospf cost = 4

TPVP - Tiramisu Path Vector Protocol

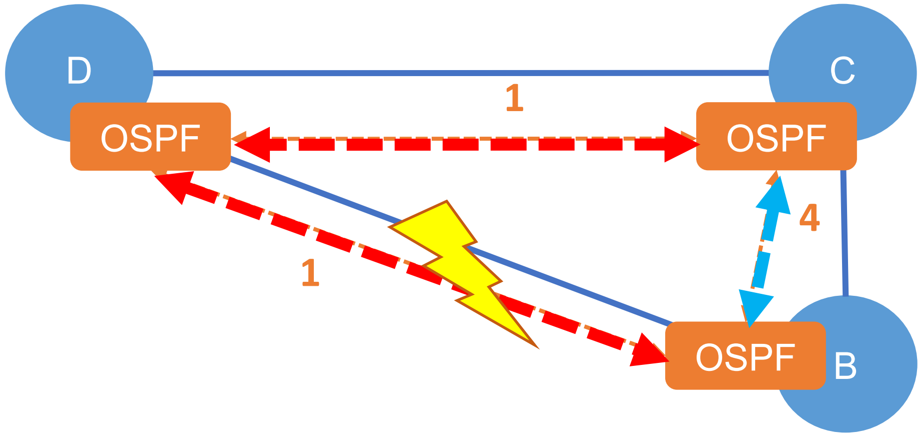


P1: C→D→B : ospf cost = 2

P2: C→B : ospf cost = 4

TPVP - Tiramisu Path Vector Protocol

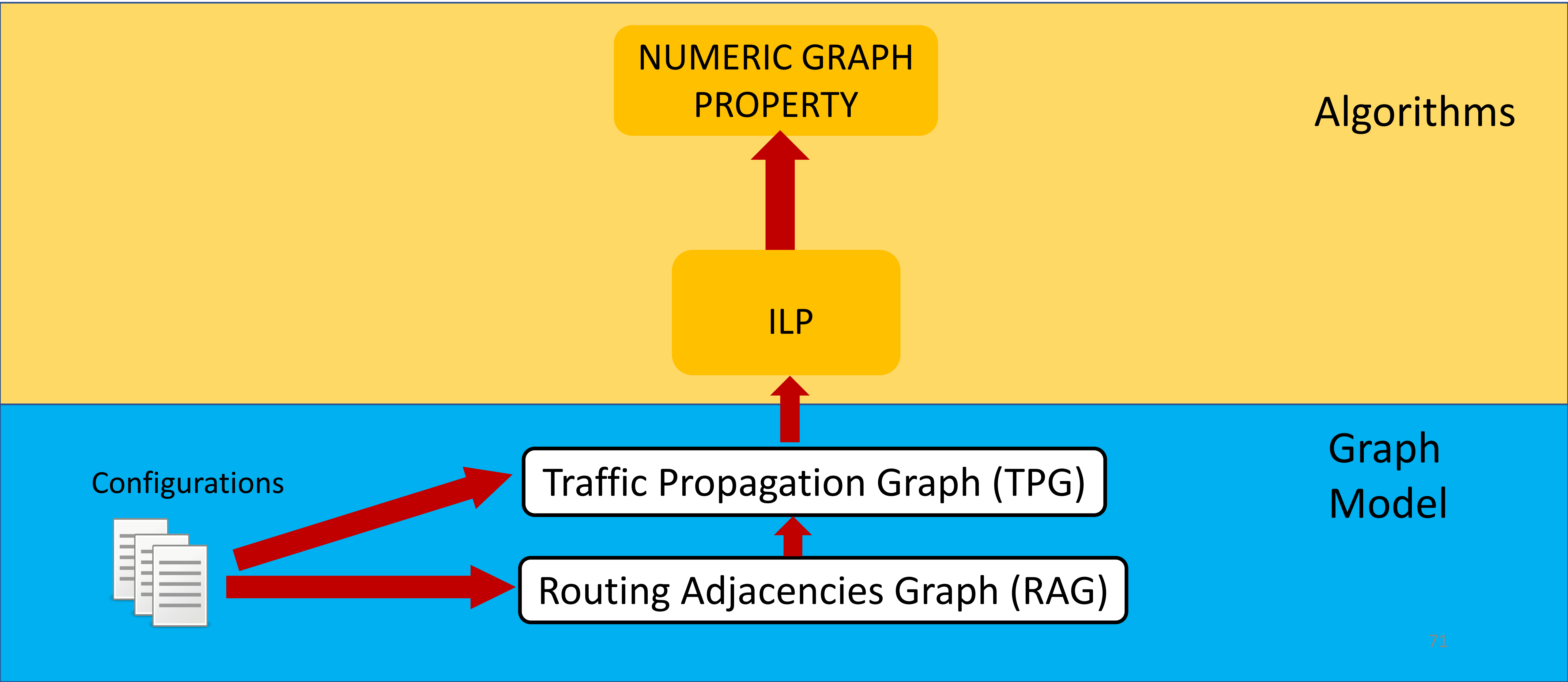
- To verify path preference (P1 << P2), Tiramisu multiple instances of TPVP for different failure scenarios
- Runs TPVP three times



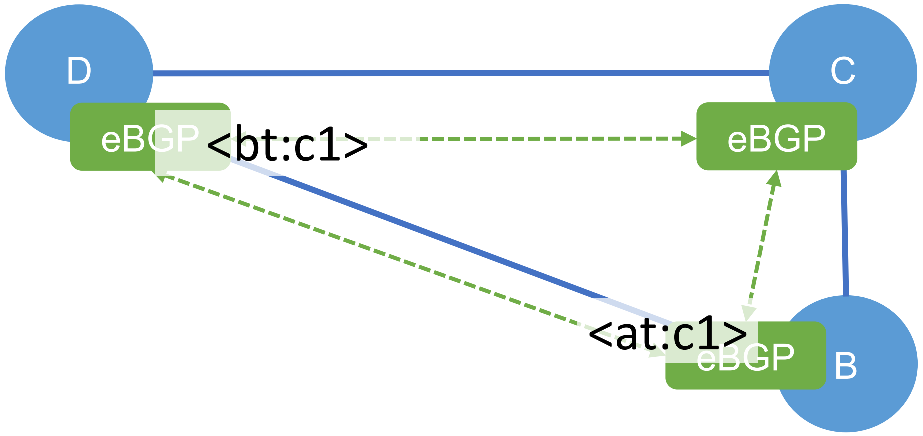
P1: C→D→B : ospf cost = 2

P2: C→B : ospf cost = 4

Tiramisu Overview

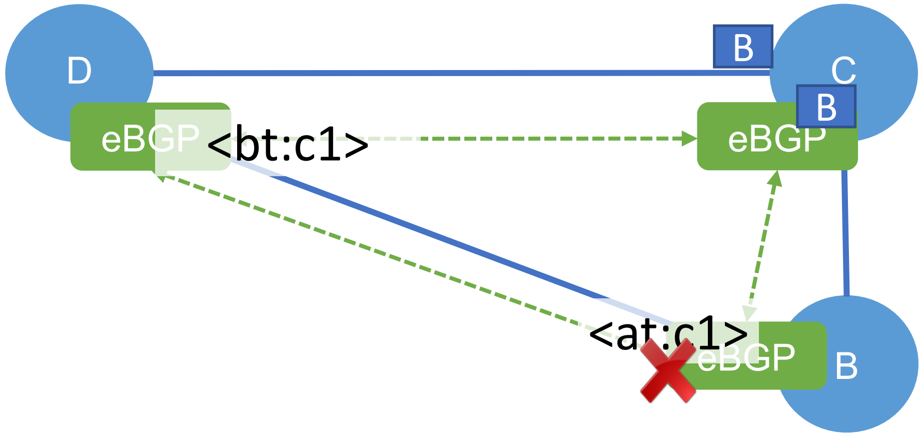


ILPs - Integer Linear Programs



- Traffic flows in the direction opposite to route advertisement

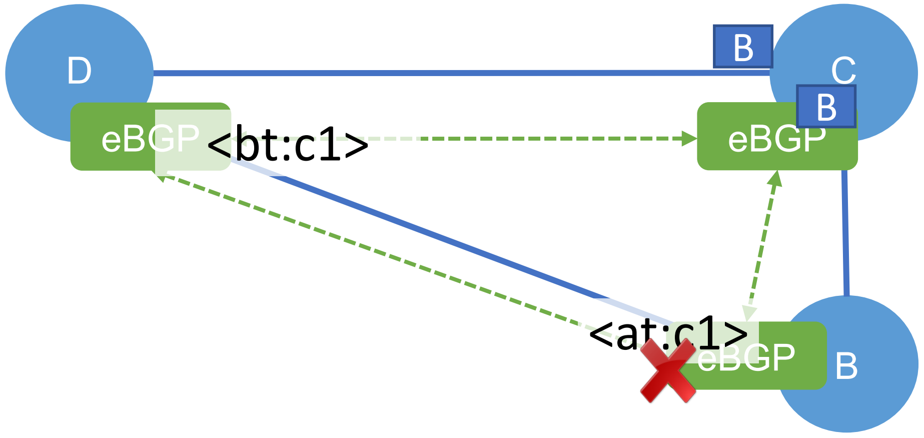
ILPs - Integer Linear Programs



min cut is 1

- Traffic flows in the direction opposite to route advertisement
- Only depends on quantitative path property and not exact path

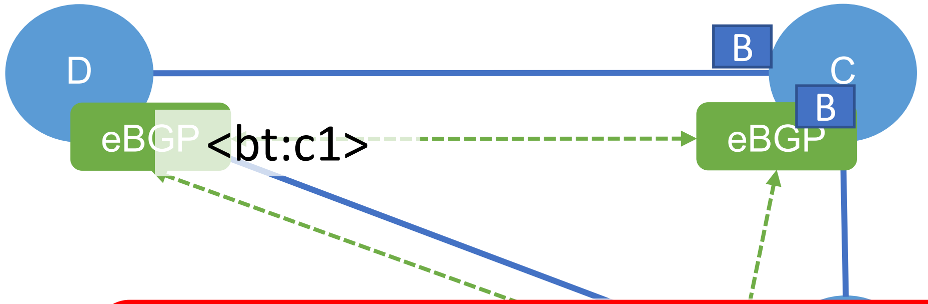
ILPs - Integer Linear Programs



min cut is 1

- Traffic flows in the direction opposite to route advertisement
- Only depends on quantitative path property and not exact path
- ILP constraints model reachability
  - Tag/Community

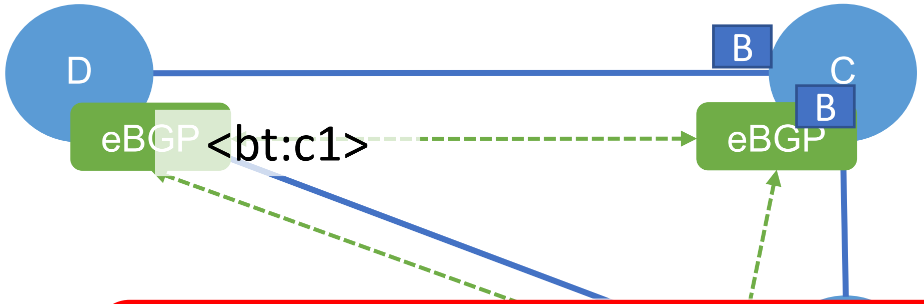
ILPs - Integer Linear Programs



Reachability < K failures = Minimize link failures
Longest path = Maximize path length

- Traffic flows in the direction opposite to route advertisement
- Only depends on quantitative path property and not exact path
- ILP constraints model reachability
  - Tag/Community
- Objective models the graph property

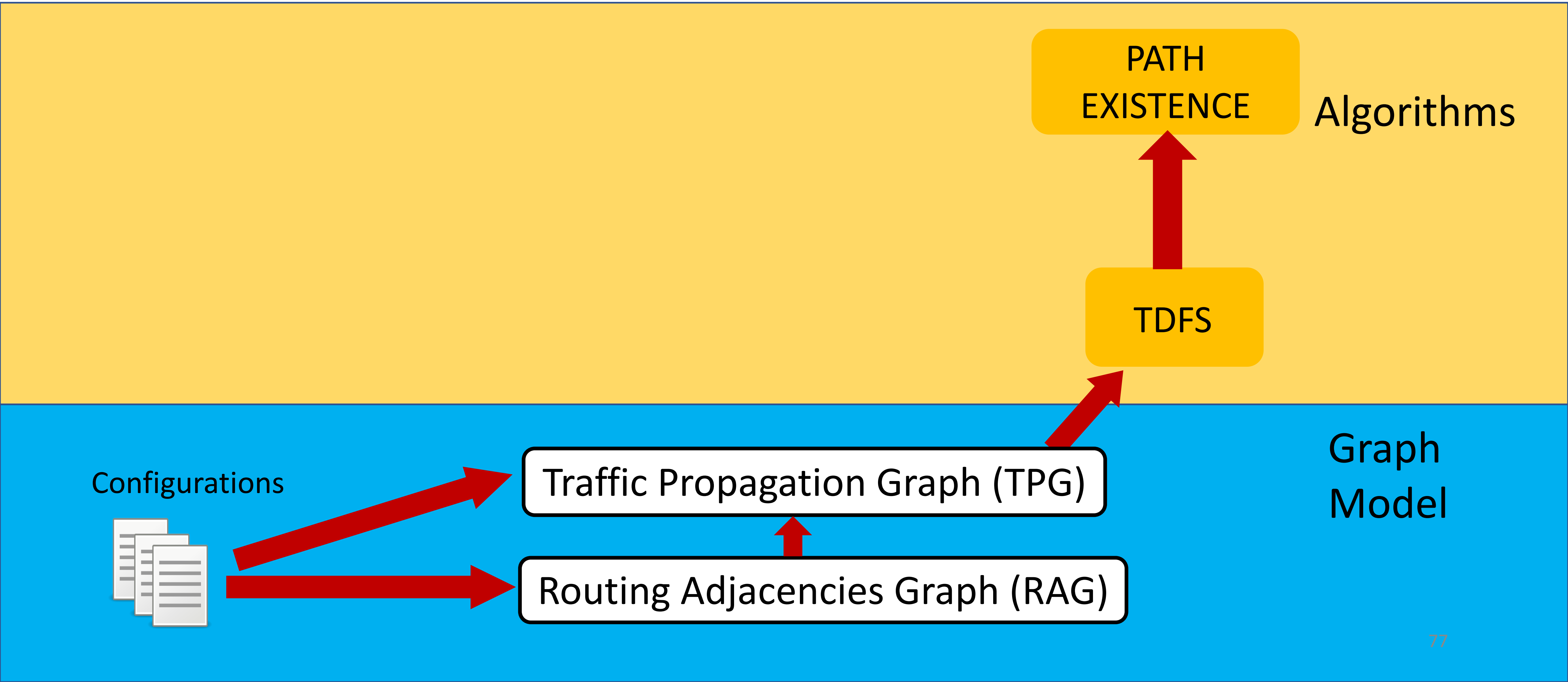
ILPs - Integer Linear Programs



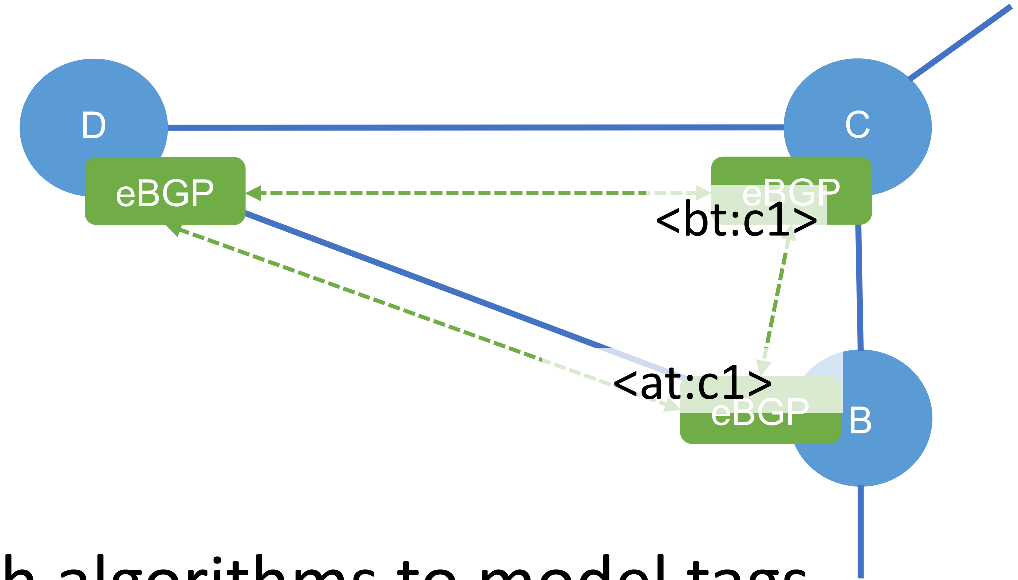
Reachability < K failures = Minimize link failures
Longest path = Maximize path length

- Traffic flows in the direction opposite to route advertisement
- Only depends on quantitative path property and not exact path
- ILP constraints model reachability
  - Tag/Community
- Objective models the graph property
- More details about the ILP and its corner cases are in the paper

Tiramisu Overview

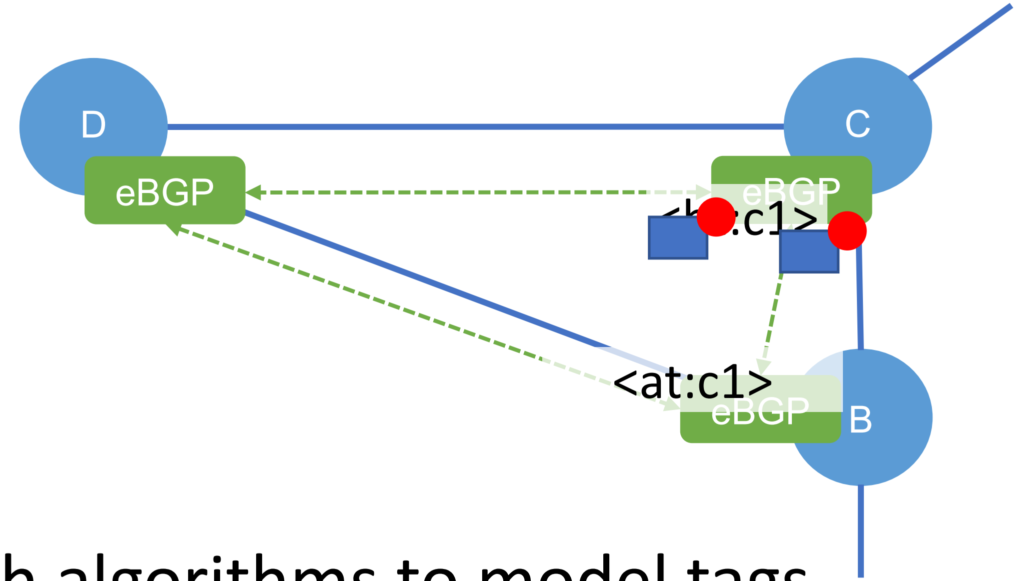


TDFS – Tiramisu Depth First Search



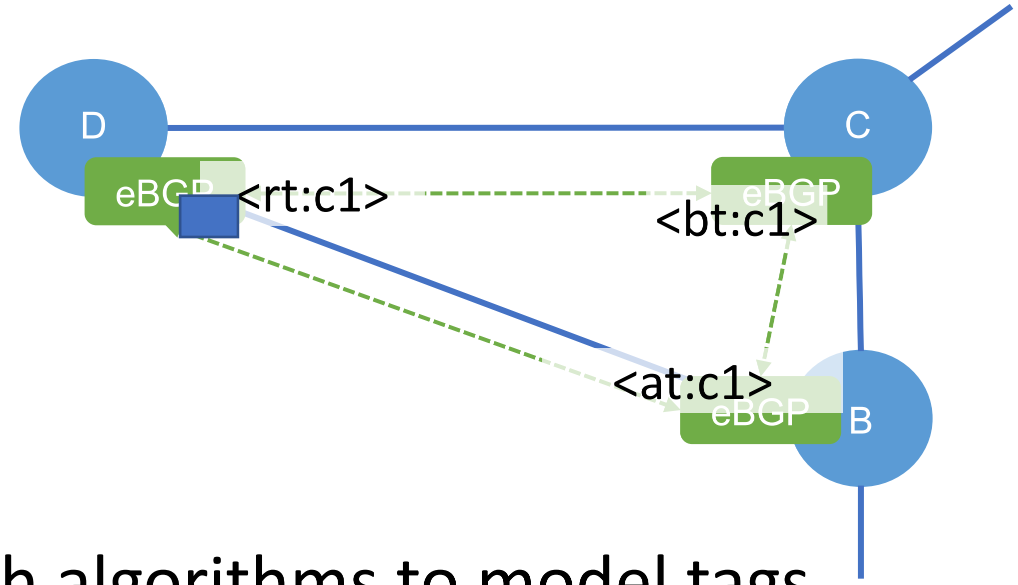
- Can’t use vanilla graph algorithms to model tags

TDFS – Tiramisu Depth First Search



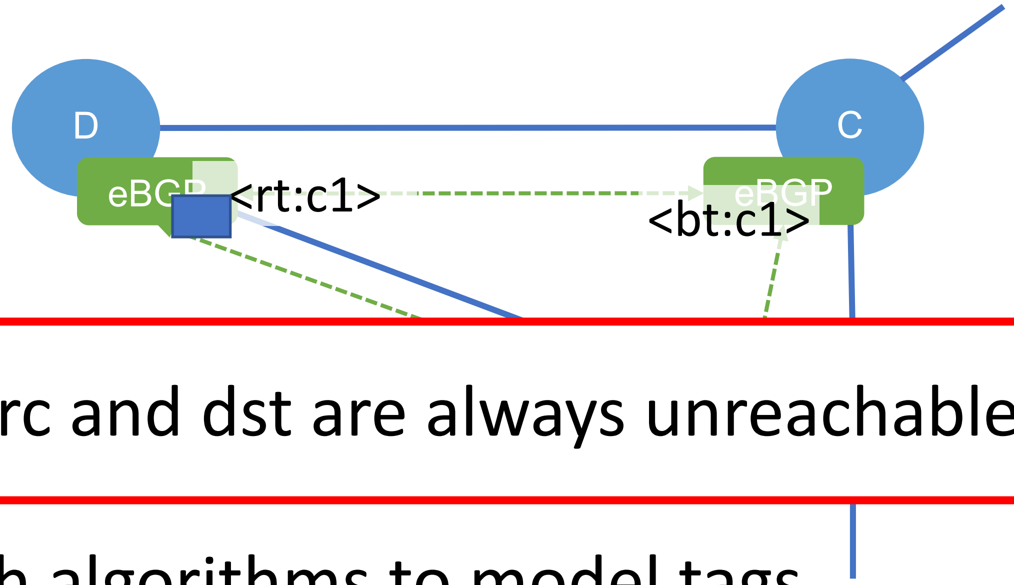
- Can't use vanilla graph algorithms to model tags
- Conditions
  - Block path: tag-blocking node → tag-adding node

TDFS – Tiramisu Depth First Search



- Can't use vanilla graph algorithms to model tags
- Conditions
  - Block path: tag-blocking node → tag-adding node
  - Allow path: tag-blocking node → tag-removing node → tag-adding node

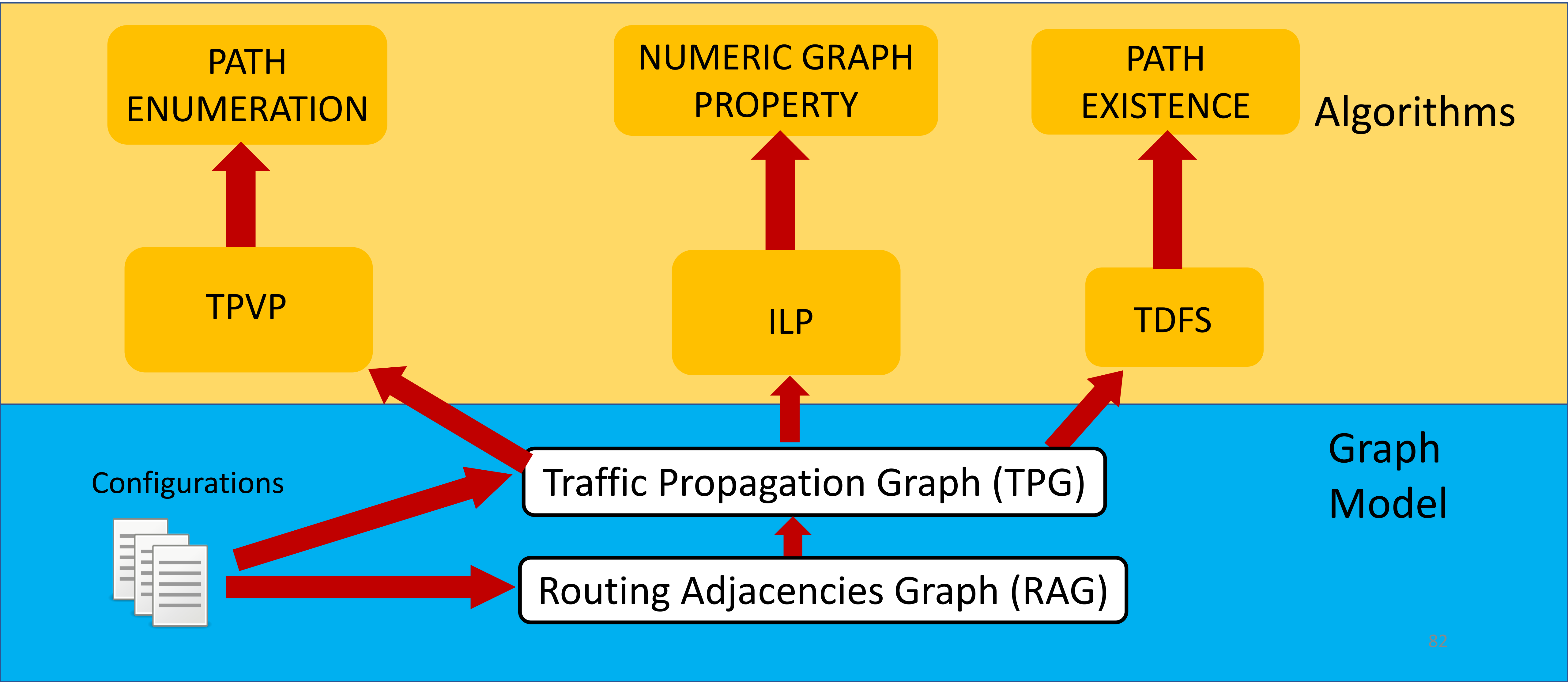
TDFS – Tiramisu Depth First Search



Verify if src and dst are always unreachable

- Can't use vanilla graph algorithms to model tags
- Conditions
  - Block path: tag-blocking node → tag-adding node
  - Allow path: tag-blocking node → tag-removing node → tag-adding node

Tiramisu Overview

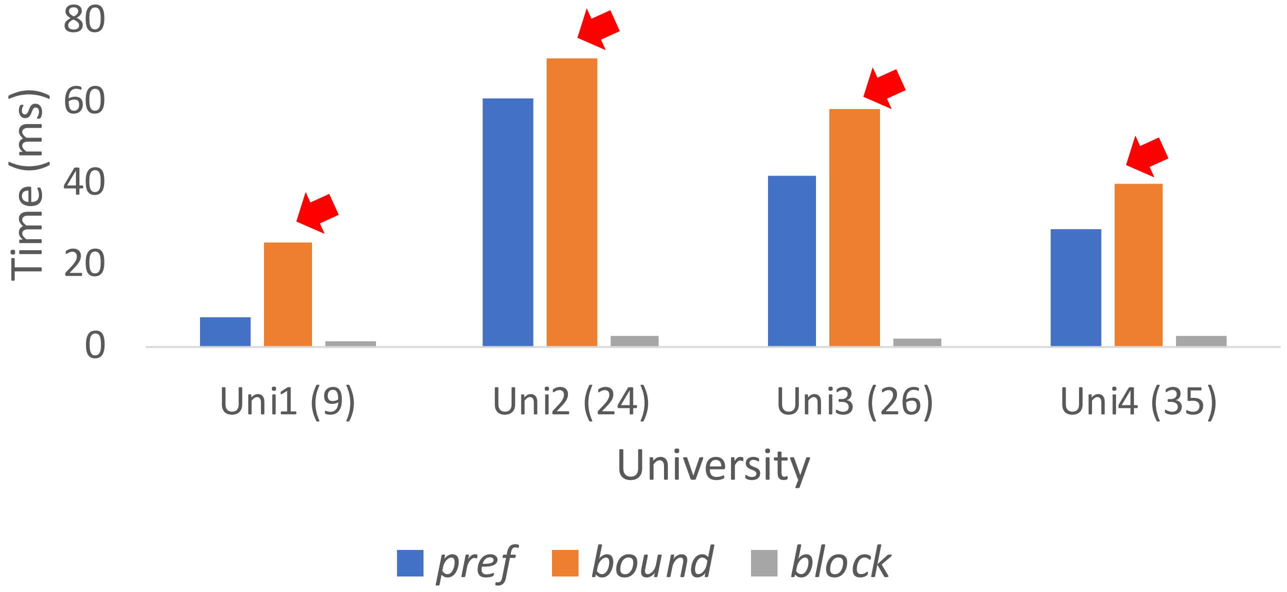


Evaluation

- Networks used
  - Real networks: 4 universities and 34 datacenters
- Evaluation
  - Tiramisu verification performance
  - Comparison with other state-of-the-art

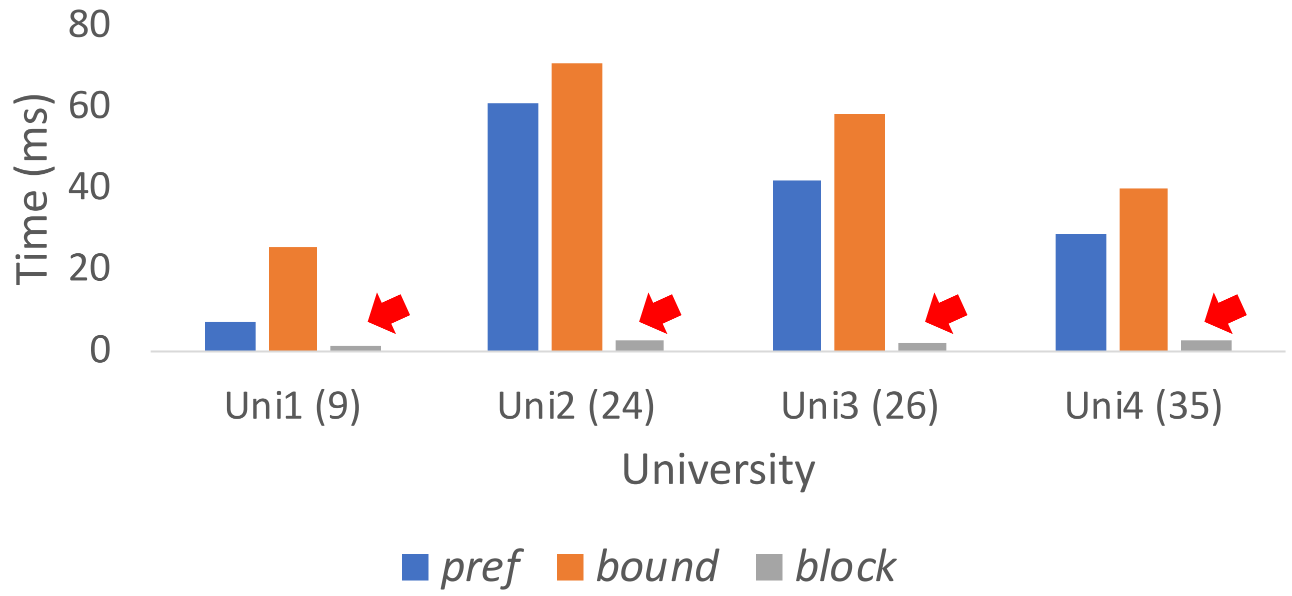
Policy	Name	Algorithm
block	Always blocked	TDFS
bound	Always bounded length	ILP
pref	Path preference	TPVP

Evaluation – Tiramisu’s Performance



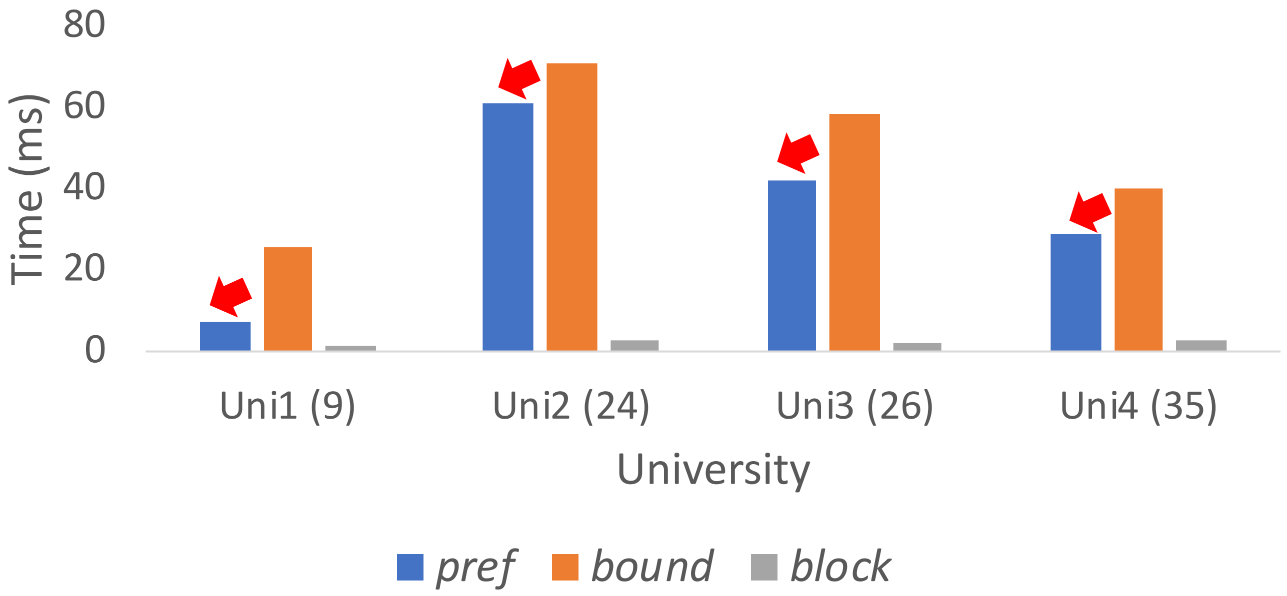
- *bound* is slowest because it uses ILP

Evaluation – Tiramisu’s Performance



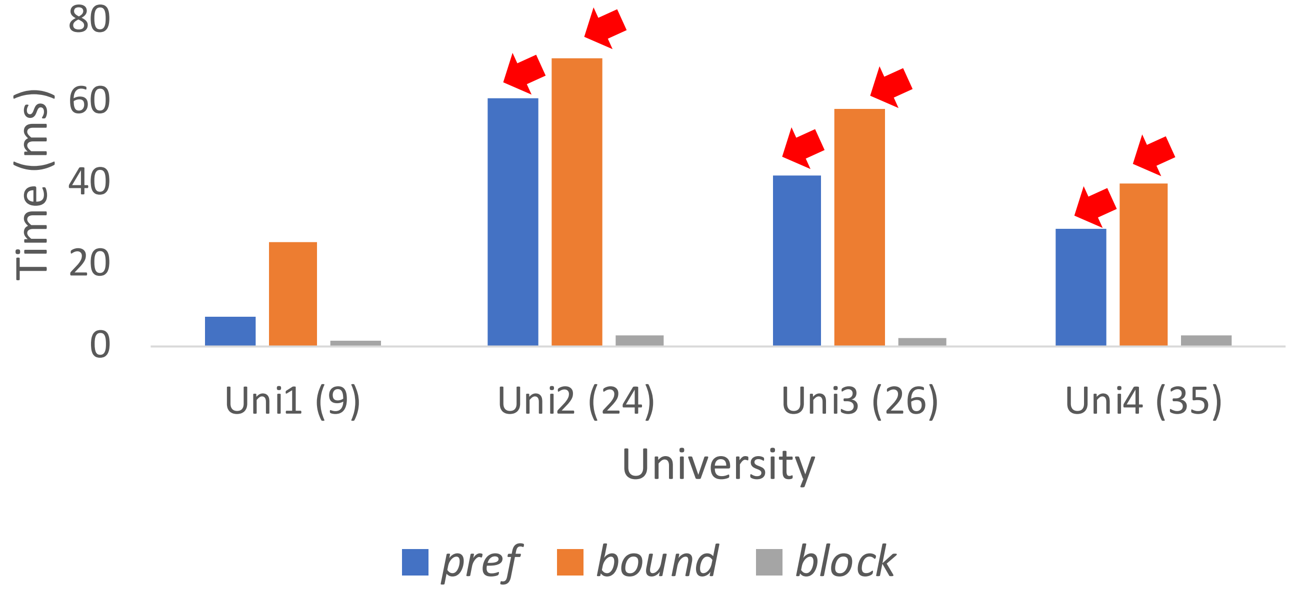
- *bound* is slowest because it uses ILP
- *block* is fastest because it uses TDFS

Evaluation – Tiramisu's Performance



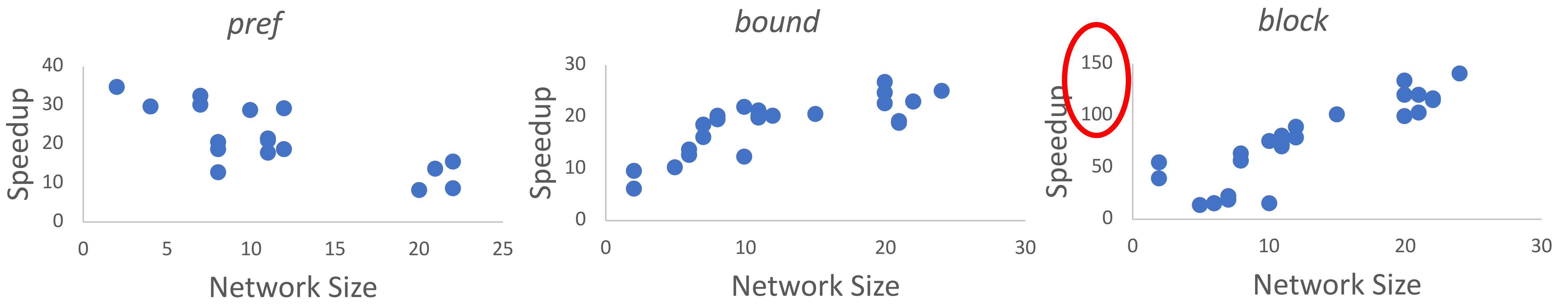
- *bound* is slowest because it uses ILP
- *block* is fastest because it uses TDFS
- *pref* is slower than *block* as TPVP is more complex

Evaluation – Tiramisu's Performance



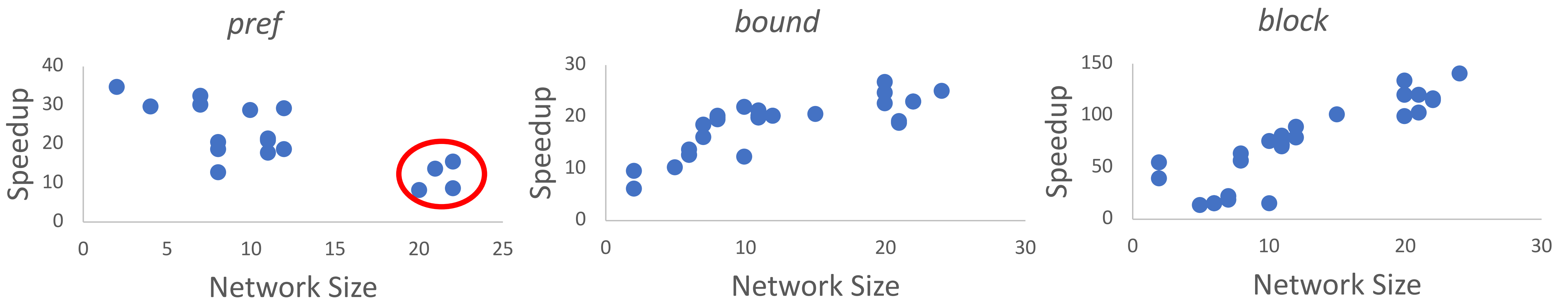
- *bound* is slowest because it uses ILP
- *block* is fastest because it uses TDFS
- *pref* is slower than *block* as TPVP is more complex
- For large networks, *pref* is as long as *bound*
  - Large networks → More candidate and longer paths → More calls to TPVP

Evaluation: Tiramisu vs. Minesweeper **(No failures)**



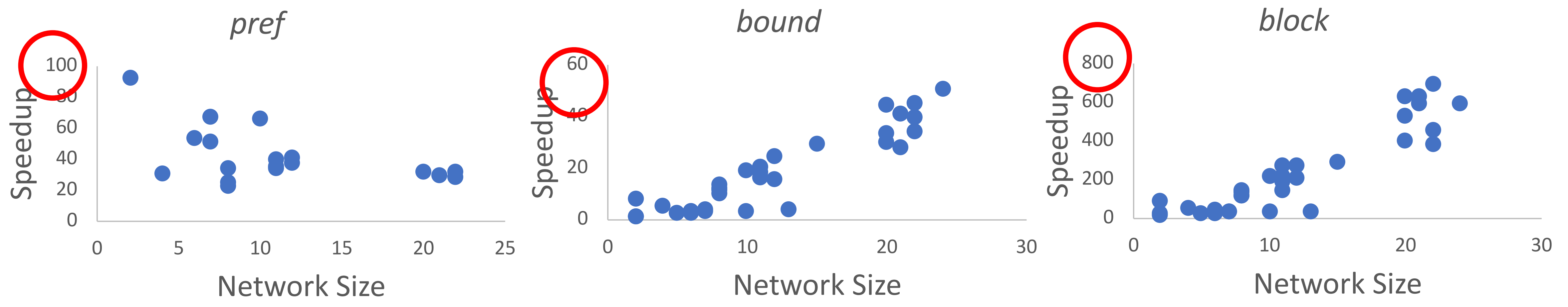
- *block* has the most speedup

Evaluation: Tiramisu vs. Minesweeper **(No failures)**



- *block* has the most speedup
- *pref* has the least speedup (especially for large networks)
  - Large networks → more and longer candidate paths → more calls to TPVP

Evaluation: Tiramisu vs. Minesweeper **(All failures)**



- Same trend but significantly better speed up
- Minesweeper uses same encoding for all policies
- Tiramisu uses property specific algorithms

Summary

- Tiramisu decouples encoding of the network from verification algorithms
- Tiramisu uses a multilayer graph control plane model
- Tiramisu uses
  - TPVP to enumerate paths
  - ILP to measure quantitative graph property
  - TDFS to check path existence
- Tiramisu achieves good performance without losing too much coverage
  - No external advertisements