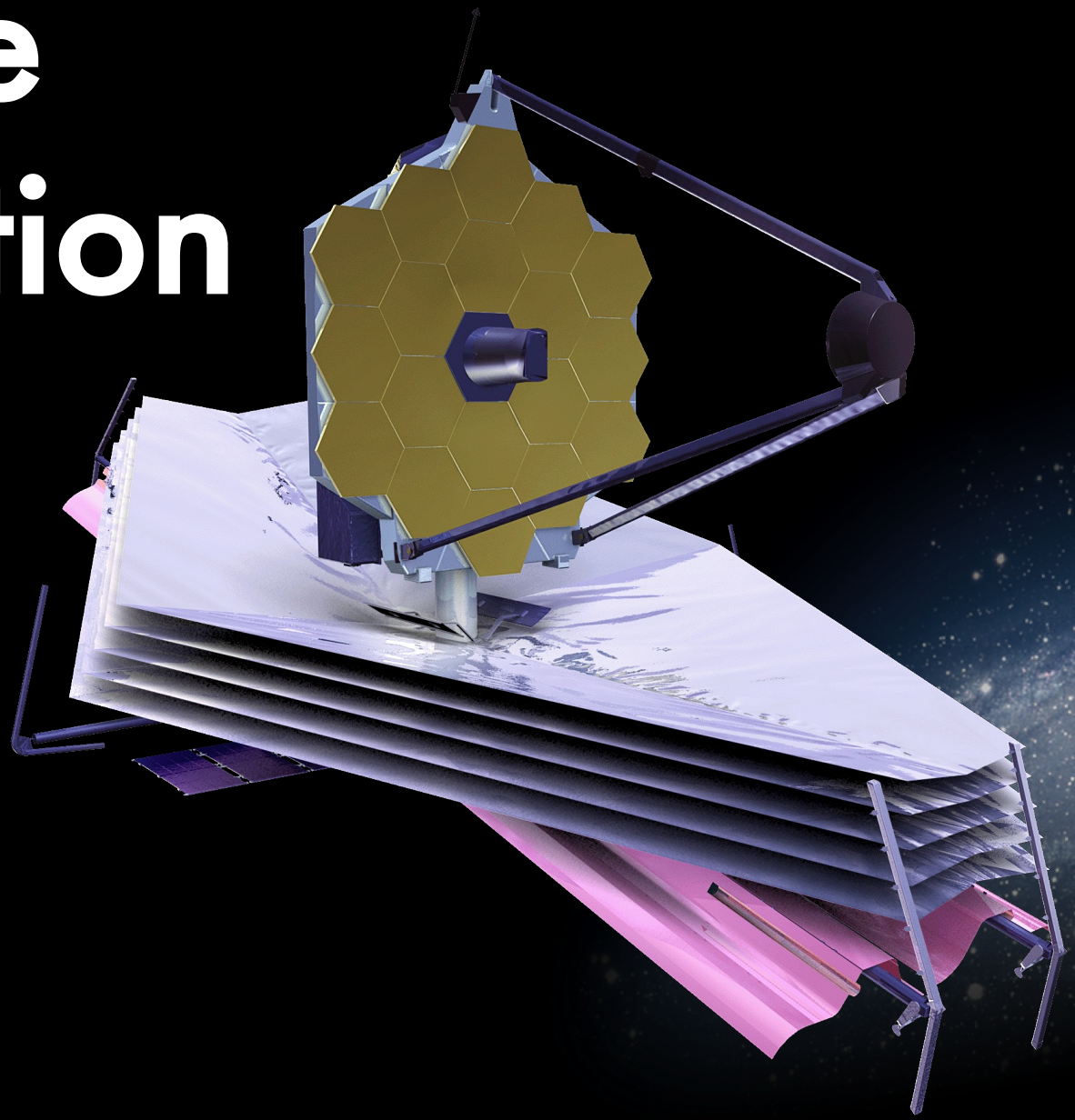
Symbolic Router Execution

Peng Zhang, Dan Wang

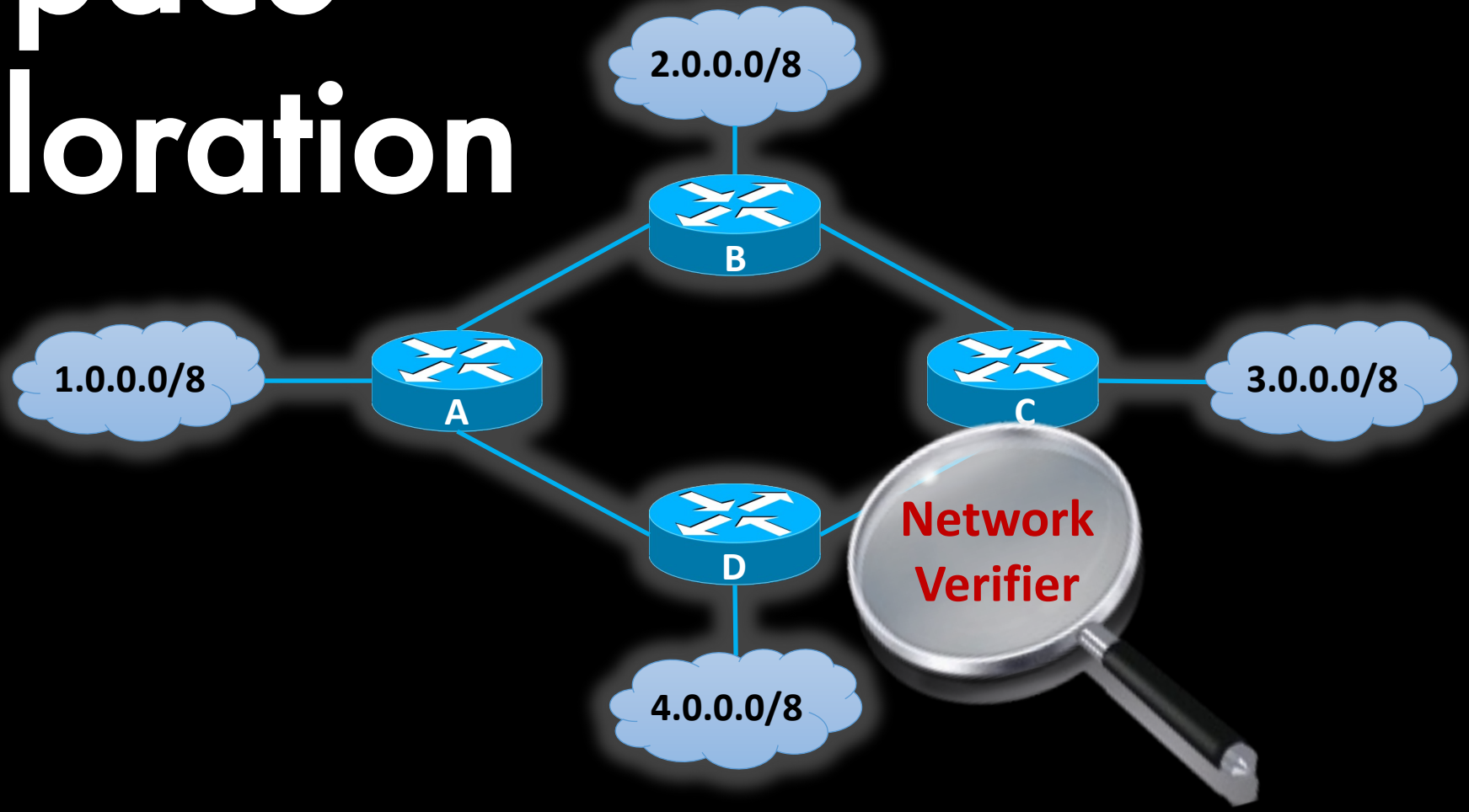
Aaron Gember-Jacobson

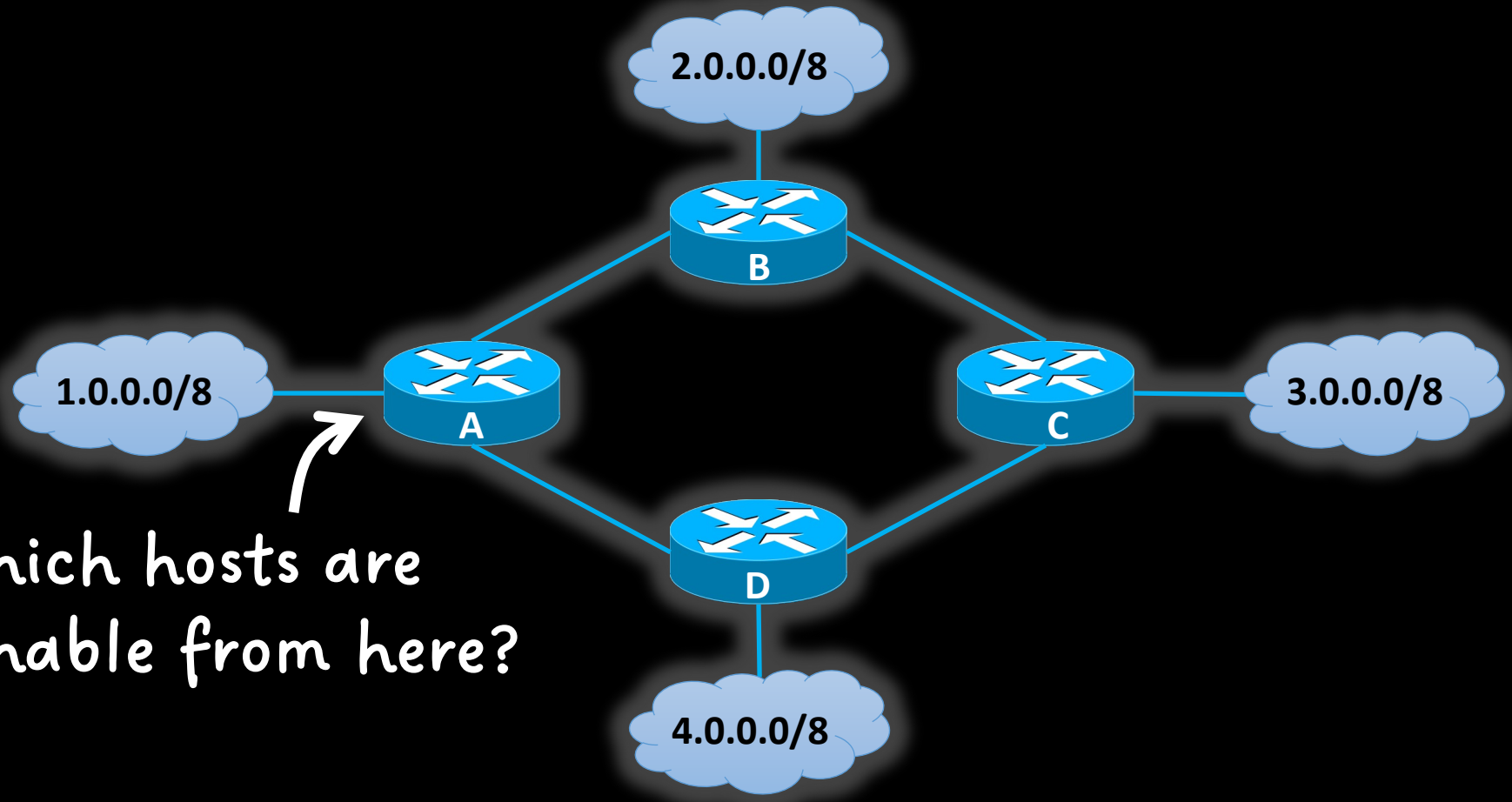
Space Exploration

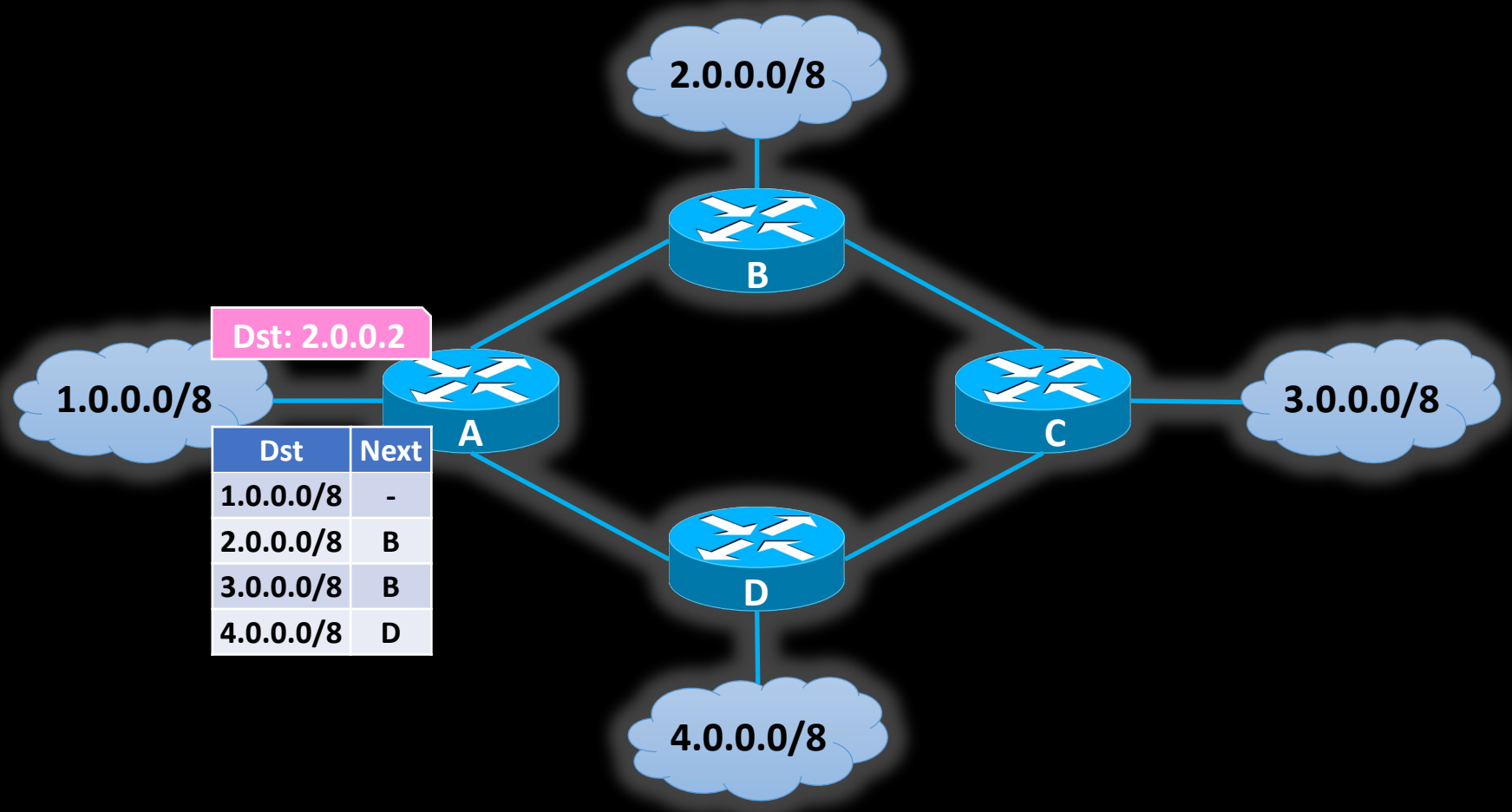


network header & failure

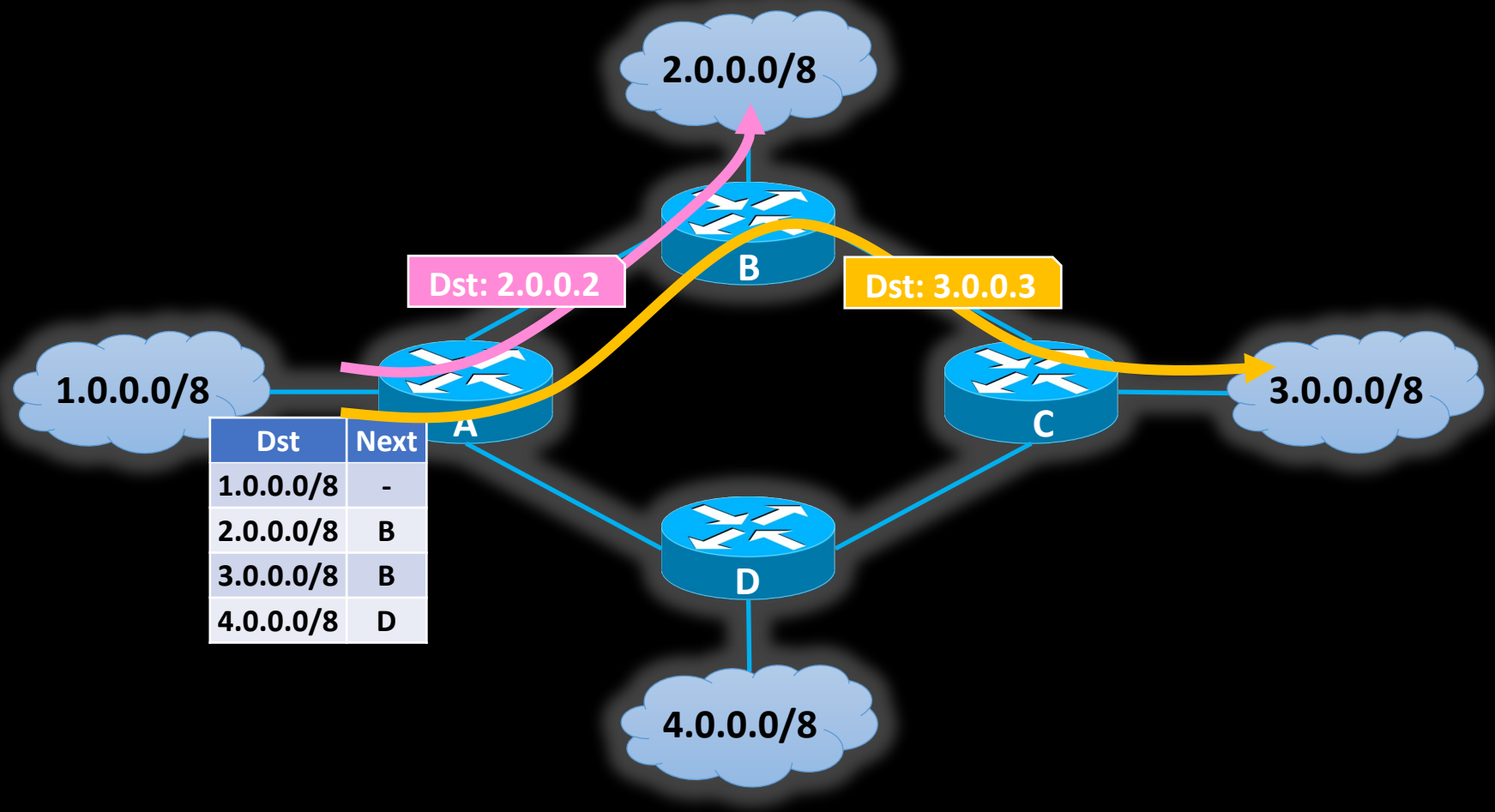
Space Exploration



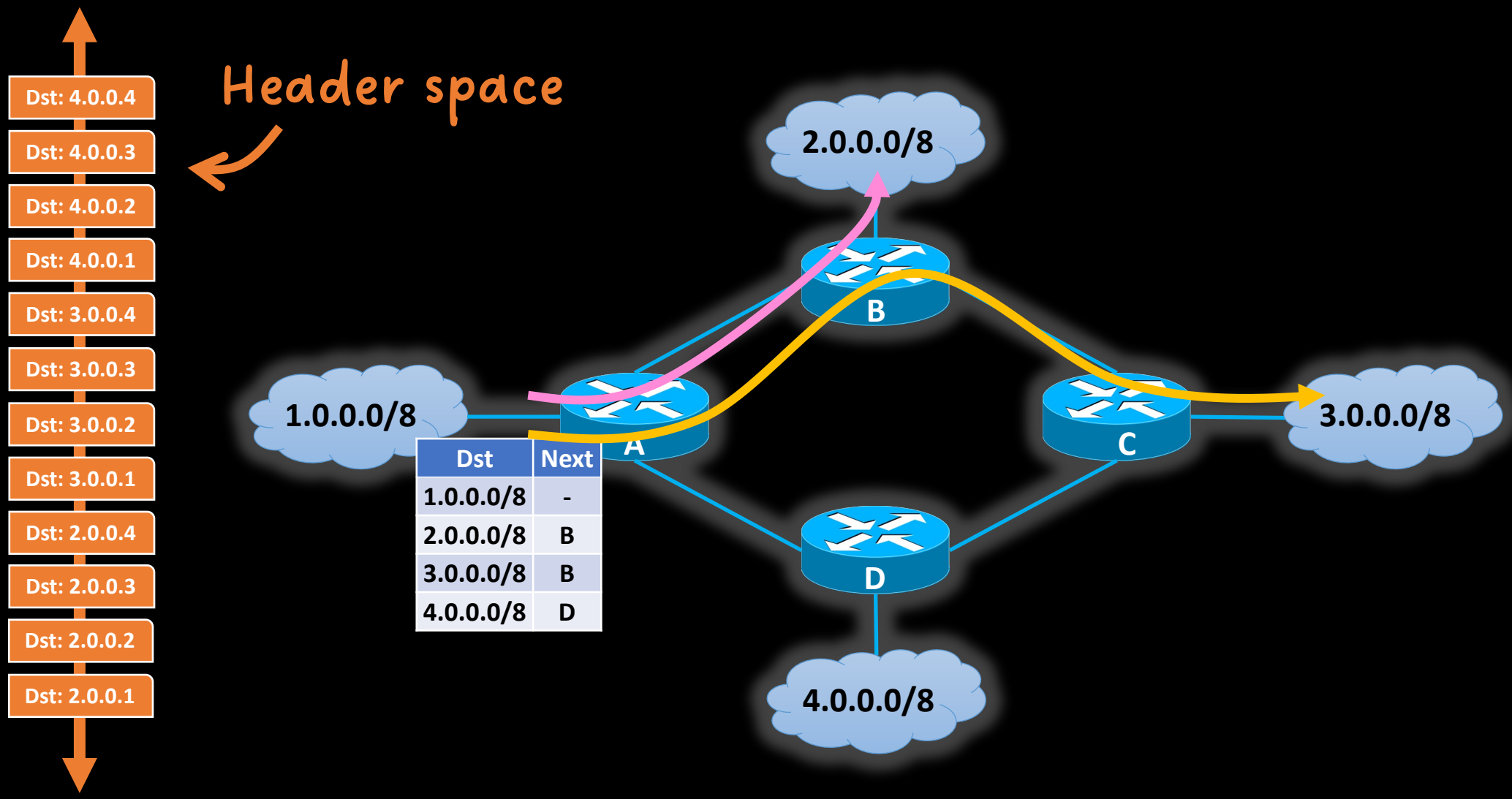

2.0.0.0/8
B
1.0.0.0/8
A
C
3.0.0.0/8
D
4.0.0.0/8
Which hosts are reachable from here?

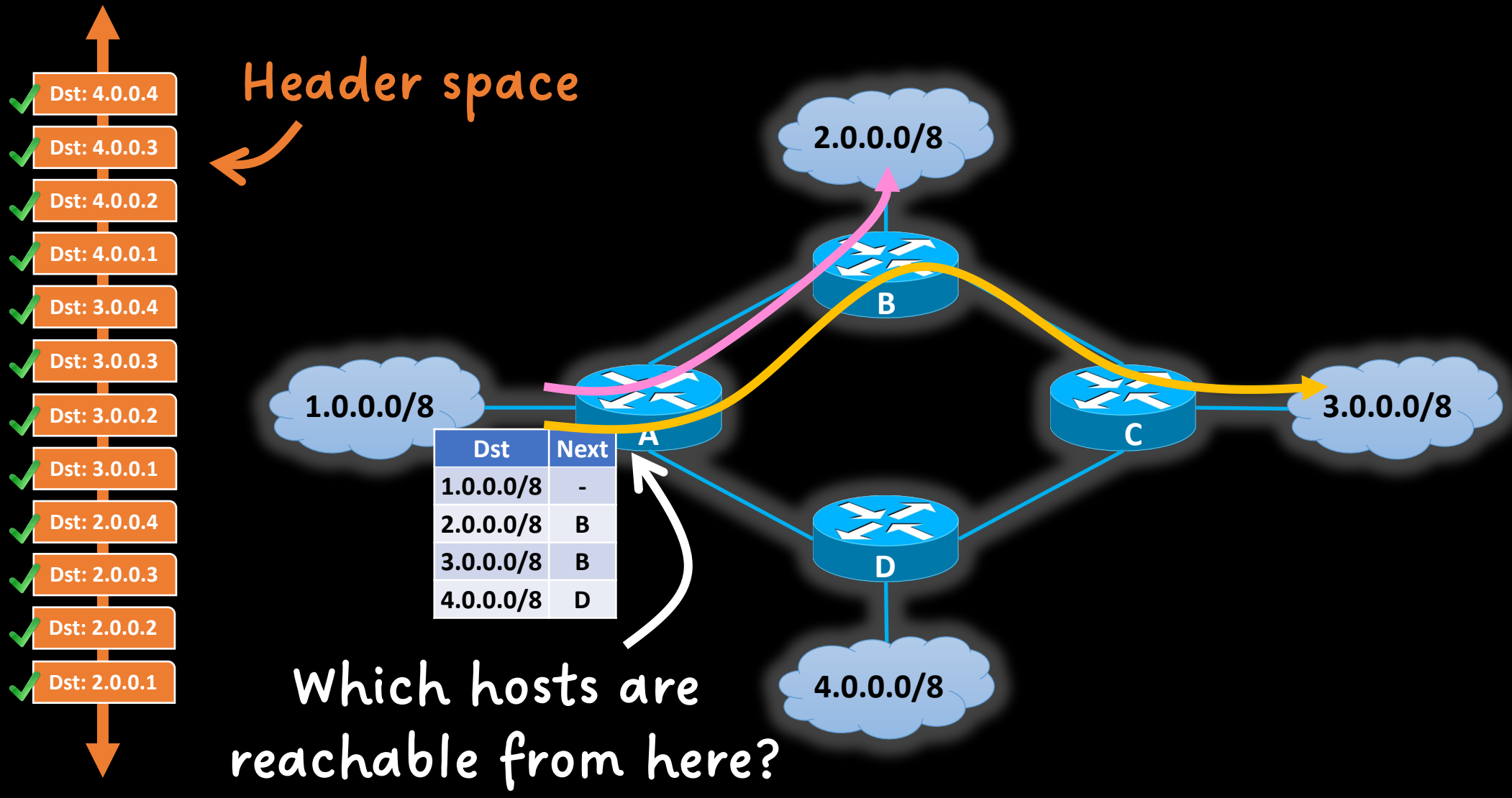


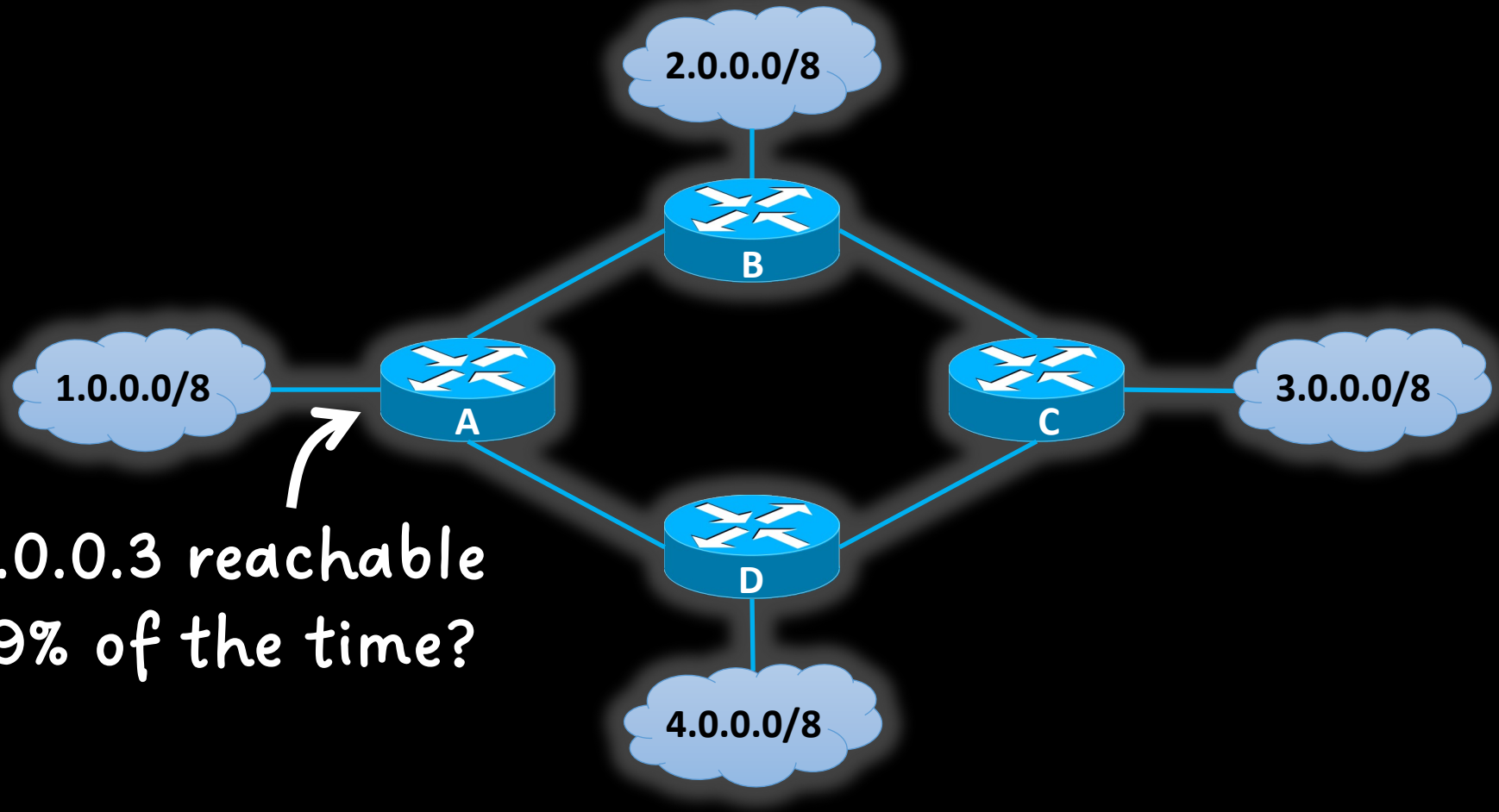
Dst	Next
1.0.0.0/8	-
2.0.0.0/8	B
3.0.0.0/8	B
4.0.0.0/8	D

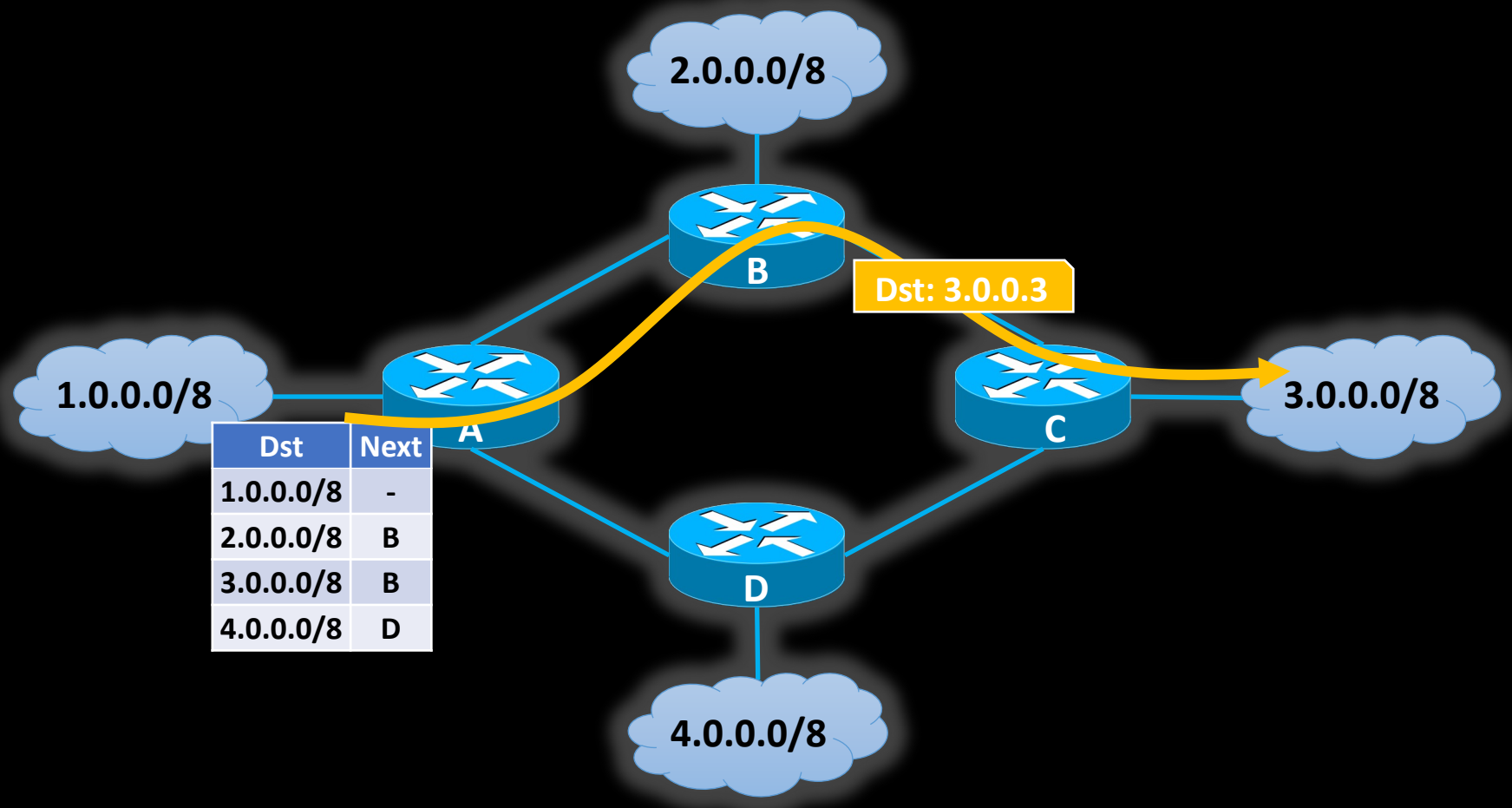


Dst	Next
1.0.0.0/8	-
2.0.0.0/8	B
3.0.0.0/8	B
4.0.0.0/8	D

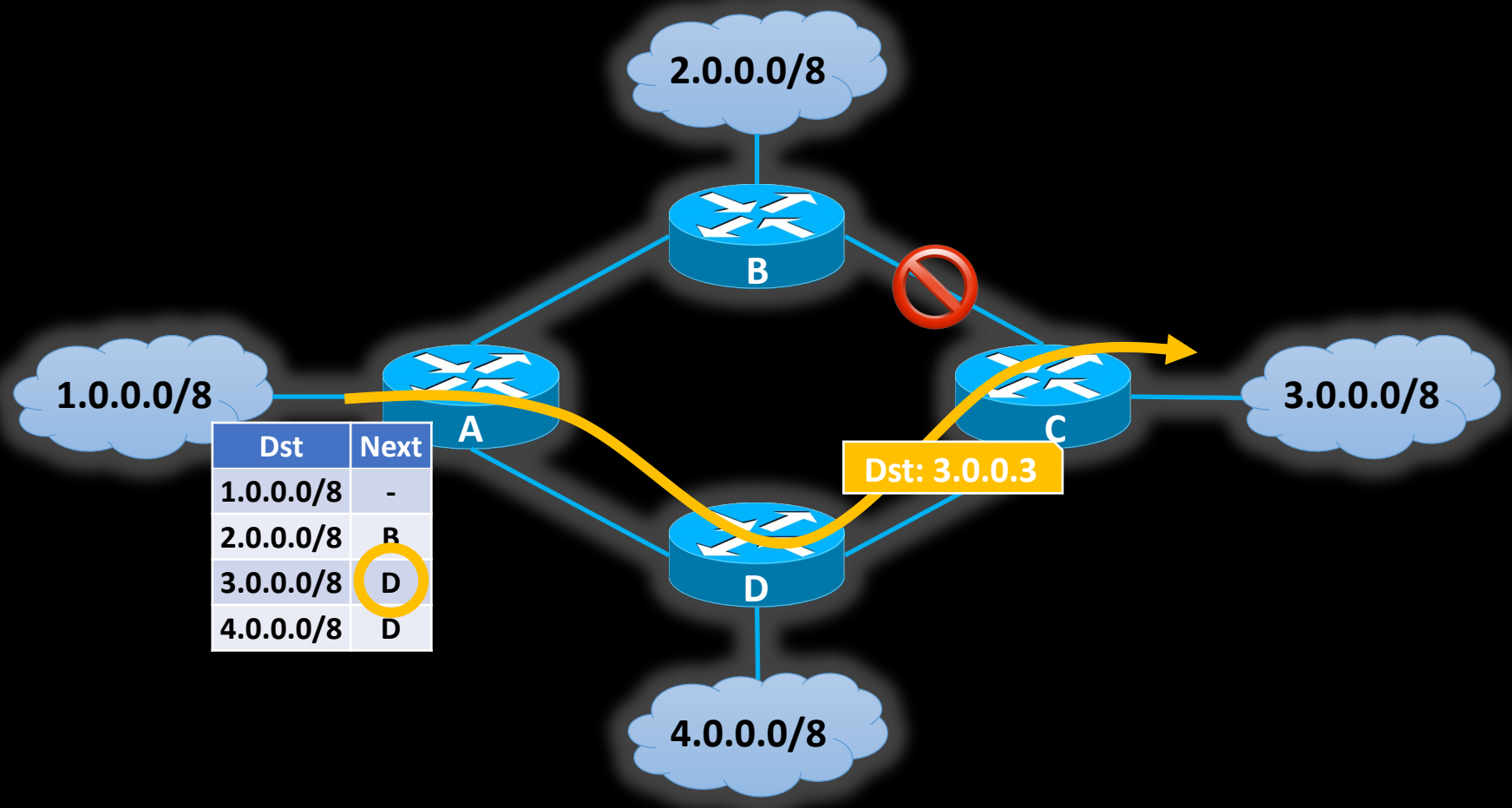

Header space
Dst: 4.0.0.4
Dst: 4.0.0.3
Dst: 4.0.0.2
Dst: 4.0.0.1
Dst: 3.0.0.4
Dst: 3.0.0.3
Dst: 3.0.0.2
Dst: 3.0.0.1
Dst: 2.0.0.4
Dst: 2.0.0.3
Dst: 2.0.0.2
Dst: 2.0.0.1
2.0.0.0/8
1.0.0.0/8
3.0.0.0/8
4.0.0.0/8
A
B
C
D
Dst
Next
1.0.0.0/8
-
2.0.0.0/8
B
3.0.0.0/8
B
4.0.0.0/8
D

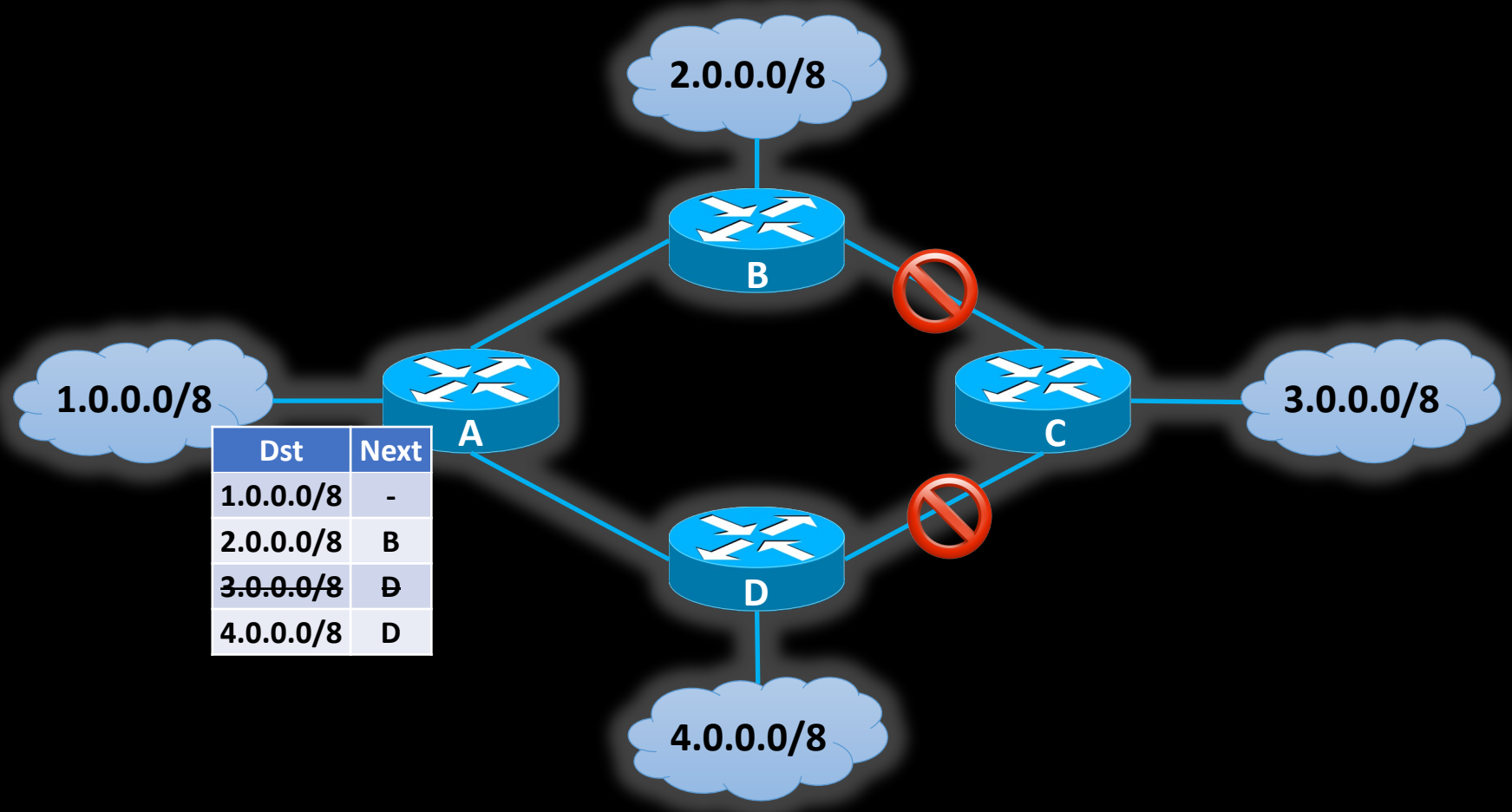

Header space
Dst: 4.0.0.4
Dst: 4.0.0.3
Dst: 4.0.0.2
Dst: 4.0.0.1
Dst: 3.0.0.4
Dst: 3.0.0.3
Dst: 3.0.0.2
Dst: 3.0.0.1
Dst: 2.0.0.4
Dst: 2.0.0.3
Dst: 2.0.0.2
Dst: 2.0.0.1
2.0.0.0/8
B
1.0.0.0/8
A
C
3.0.0.0/8
D
4.0.0.0/8
Dst
Next
1.0.0.0/8
-
2.0.0.0/8
B
3.0.0.0/8
B
4.0.0.0/8
D
Which hosts are reachable from here?


2.0.0.0/8
B
1.0.0.0/8
A
C
3.0.0.0/8
D
4.0.0.0/8
Is 3.0.0.3 reachable
99.9% of the time?

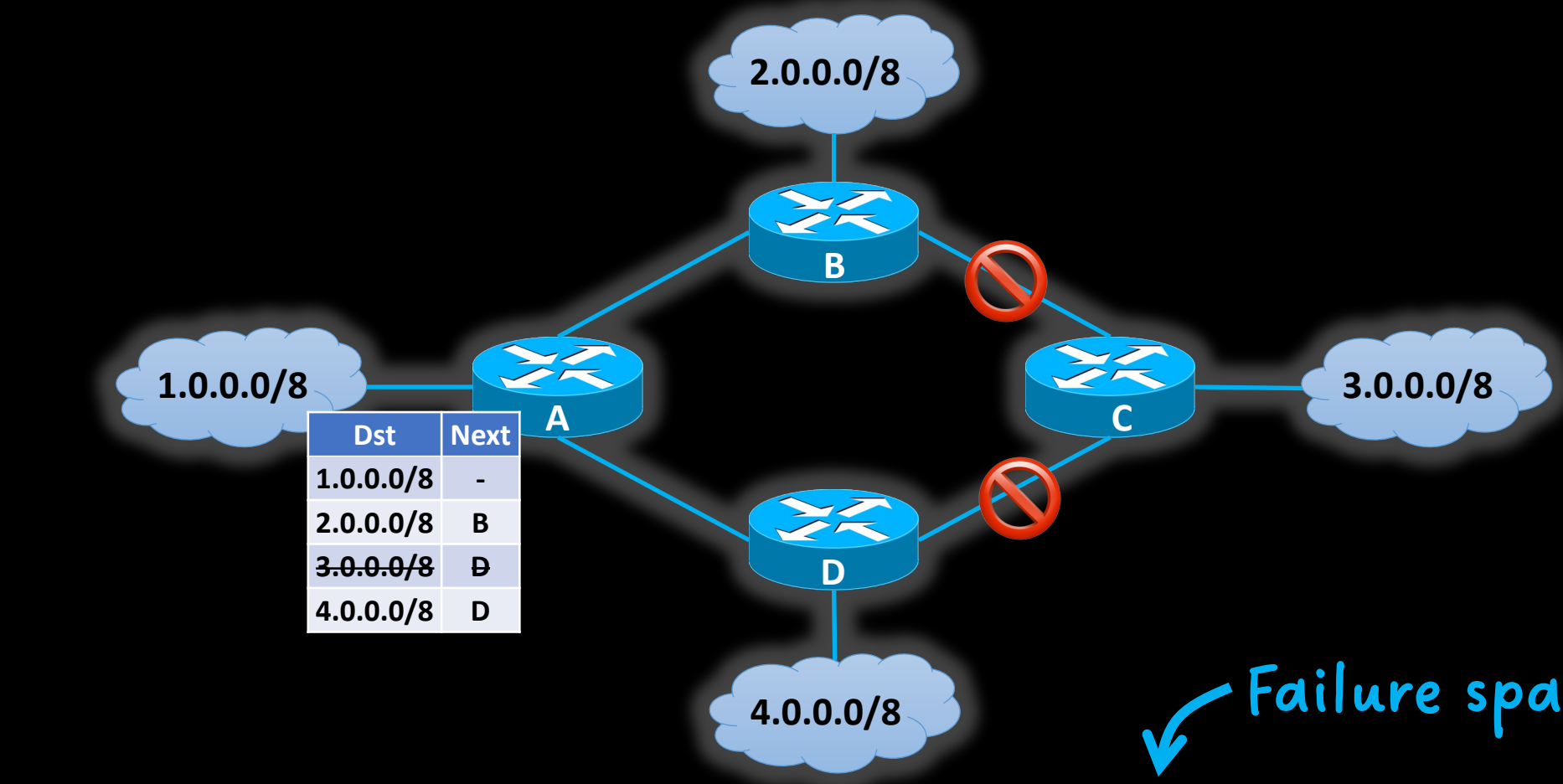


Dst	Next
1.0.0.0/8	-
2.0.0.0/8	B
3.0.0.0/8	B
4.0.0.0/8	D

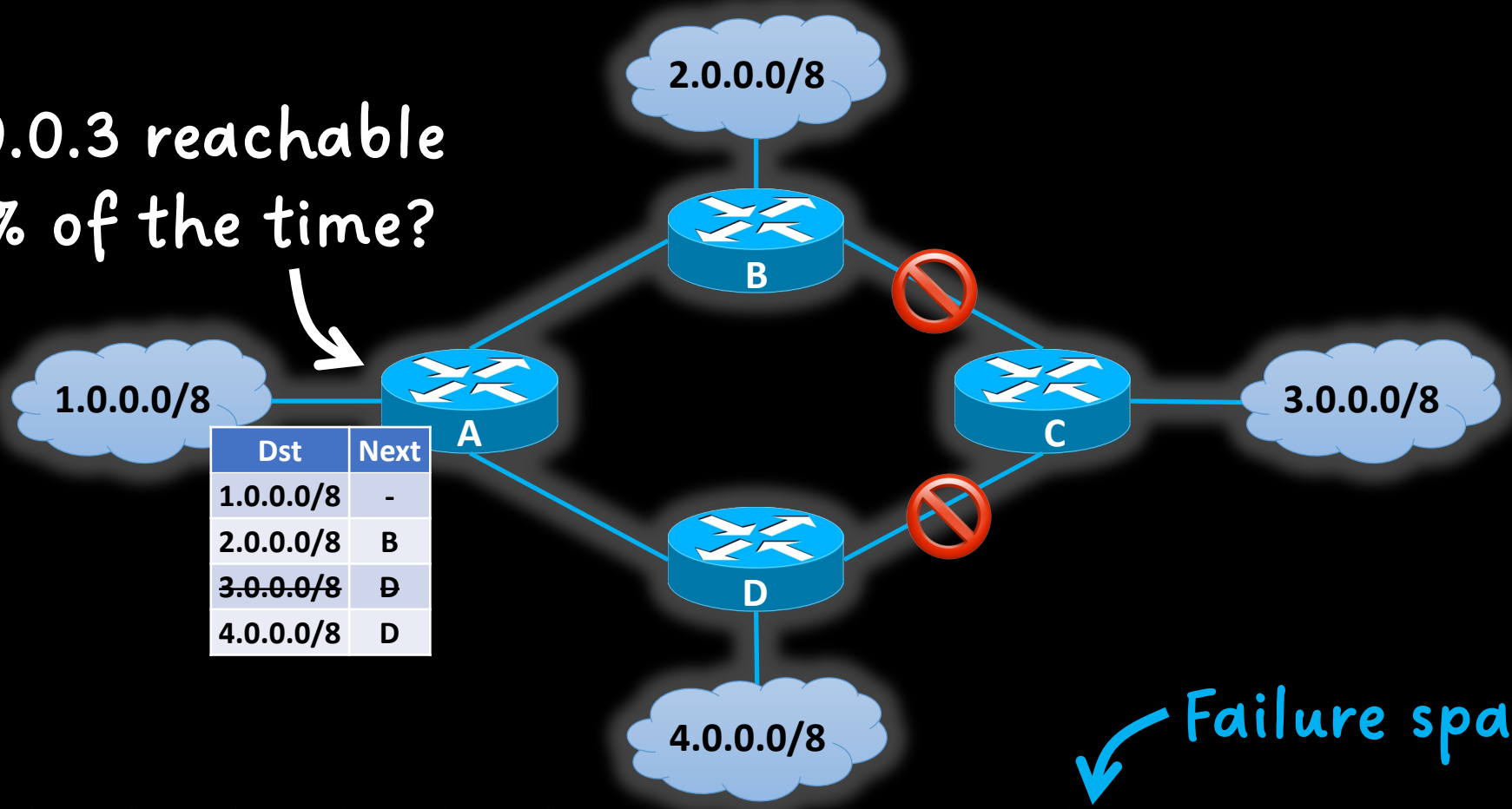

2.0.0.0/8
B
1.0.0.0/8
A
C
3.0.0.0/8
Dst: 3.0.0.3
D
4.0.0.0/8
Dst
Next
1.0.0.0/8
-
2.0.0.0/8
B
3.0.0.0/8
D
4.0.0.0/8
D



Dst	Next
1.0.0.0/8	-
2.0.0.0/8	B
~~3.0.0.0/8~~	~~D~~
4.0.0.0/8	D

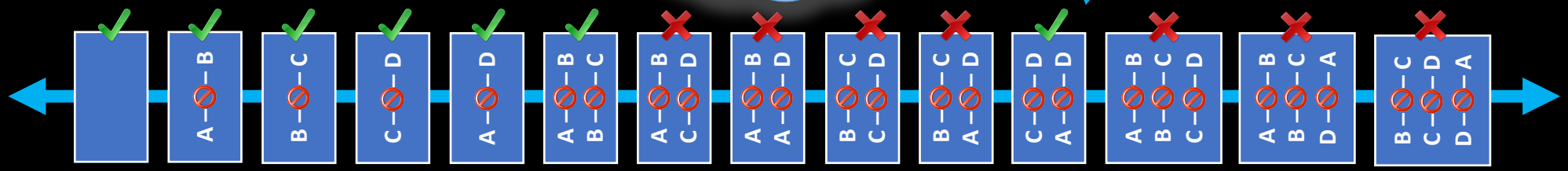

2.0.0.0/8
B
1.0.0.0/8
A
C
3.0.0.0/8
D
4.0.0.0/8
Dst
Next
1.0.0.0/8
-
2.0.0.0/8
B
~~3.0.0.0/8~~
~~D~~
4.0.0.0/8
D
Failure space
A—B
B—C
C—D
A—D
A—B
B—C
A—B
C—D
A—B
A—D
B—C
C—D
B—C
A—D
C—D
A—D
A—B
B—C
C—D
A—B
B—C
D—A
B—C
C—D
D—A

Is 3.0.0.3 reachable 99.9% of the time?

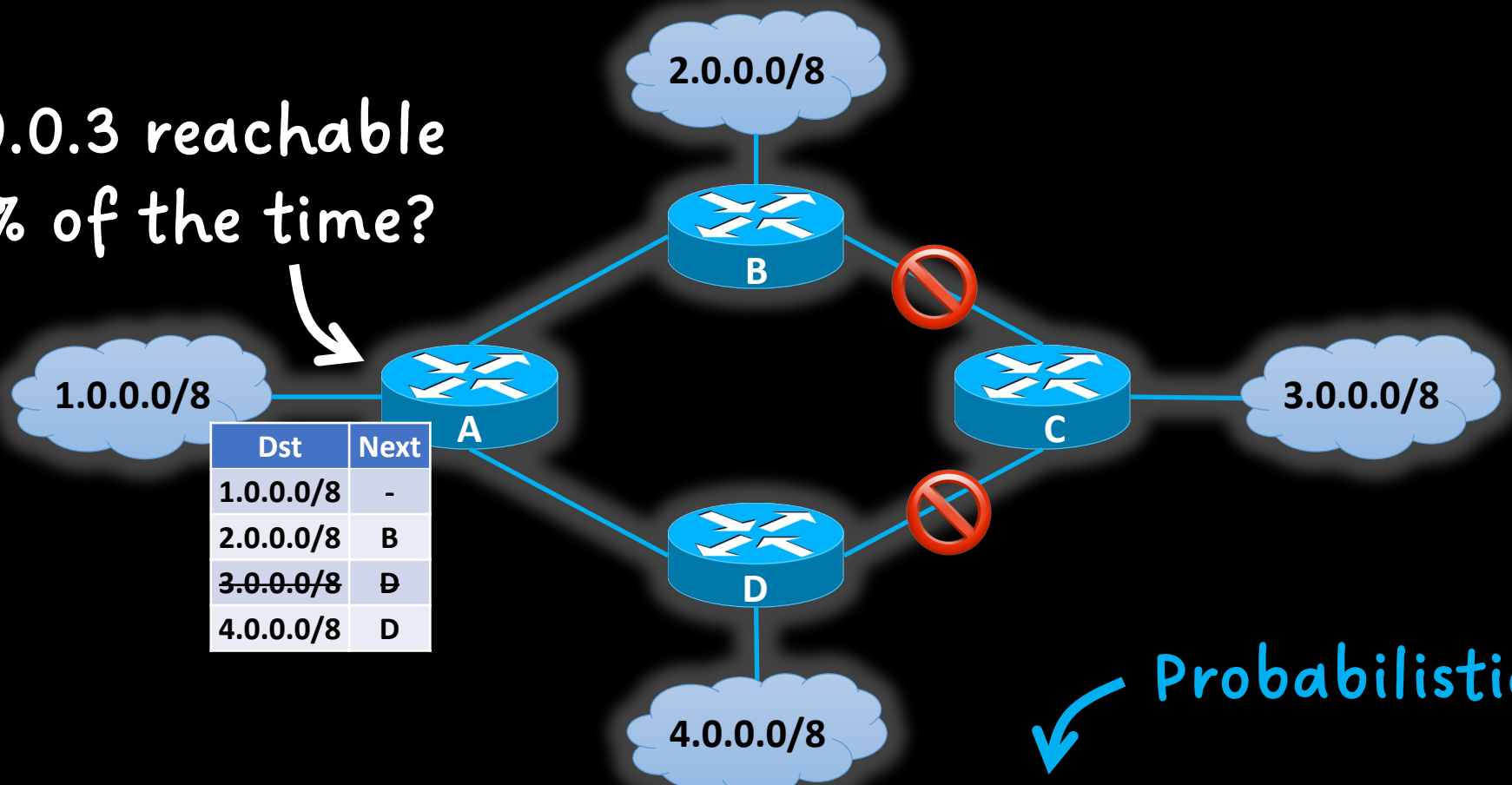


Dst	Next
1.0.0.0/8	-
2.0.0.0/8	B
~~3.0.0.0/8~~	~~D~~
4.0.0.0/8	D

Failure space

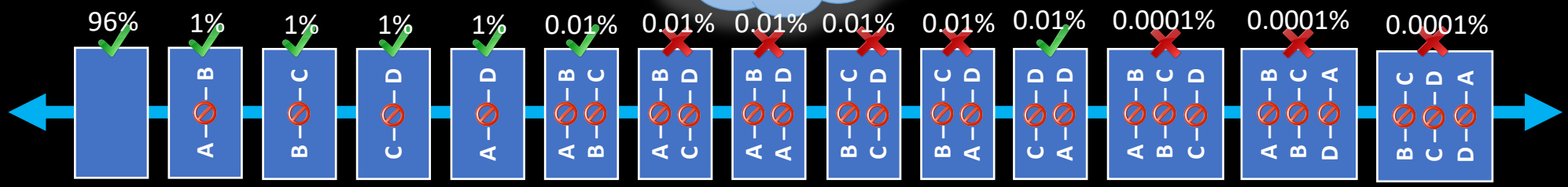


Is 3.0.0.3 reachable 99.9% of the time?

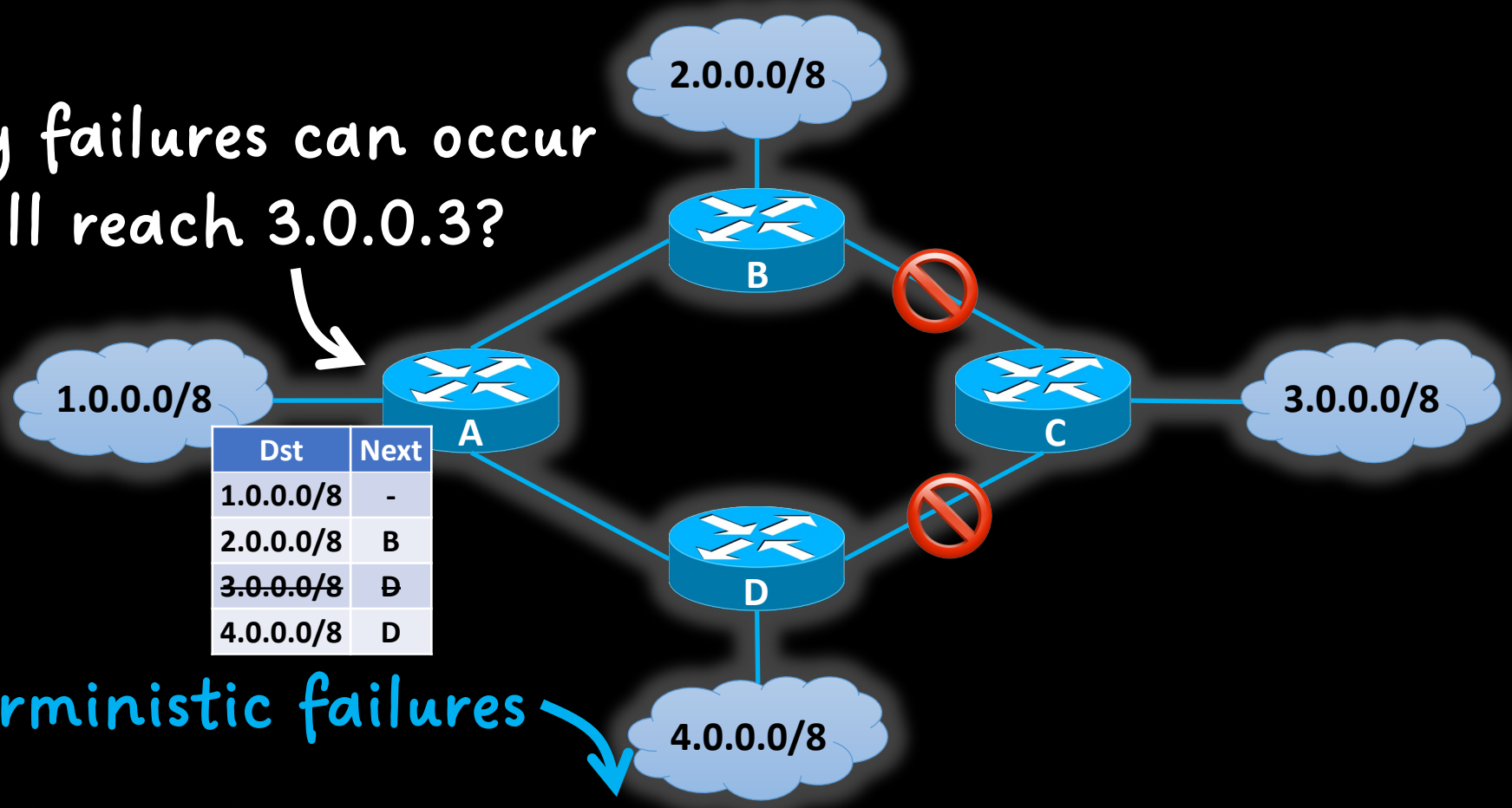


Dst	Next
1.0.0.0/8	-
2.0.0.0/8	B
~~3.0.0.0/8~~	~~D~~
4.0.0.0/8	D

Probabilistic failures

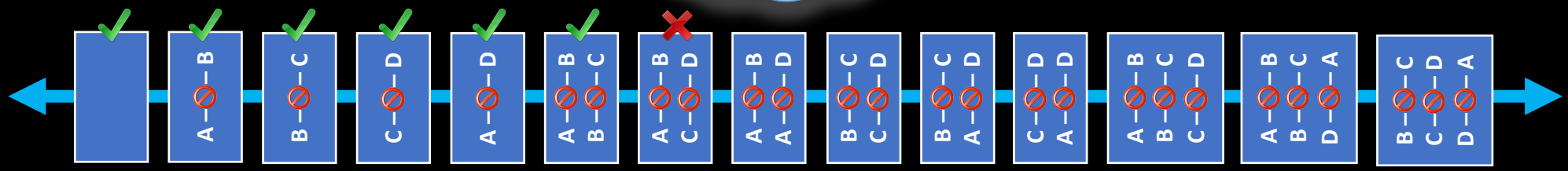


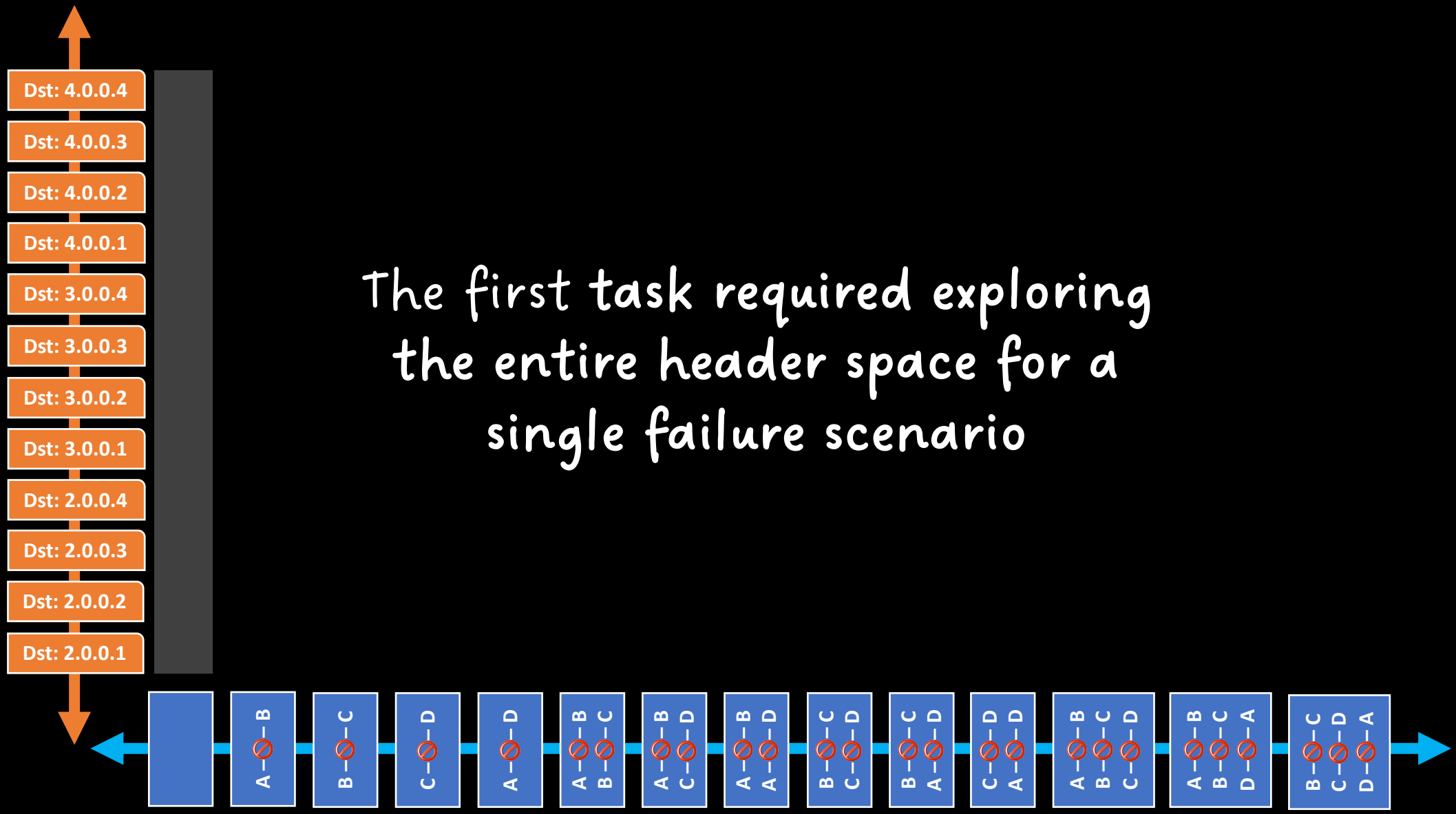
How many failures can occur and still reach 3.0.0.3?



Dst	Next
1.0.0.0/8	-
2.0.0.0/8	B
~~3.0.0.0/8~~	~~D~~
4.0.0.0/8	D

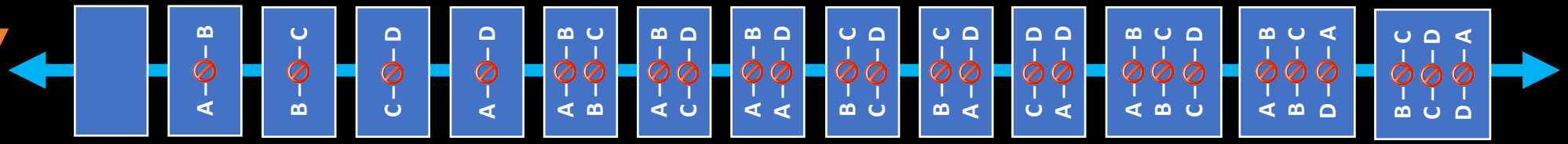
Deterministic failures





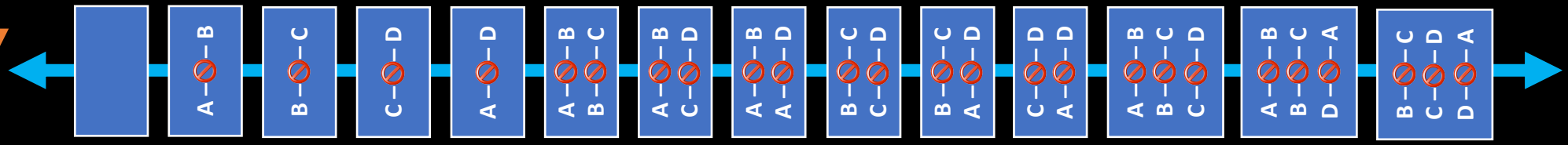
The first task required exploring the entire header space for a single failure scenario

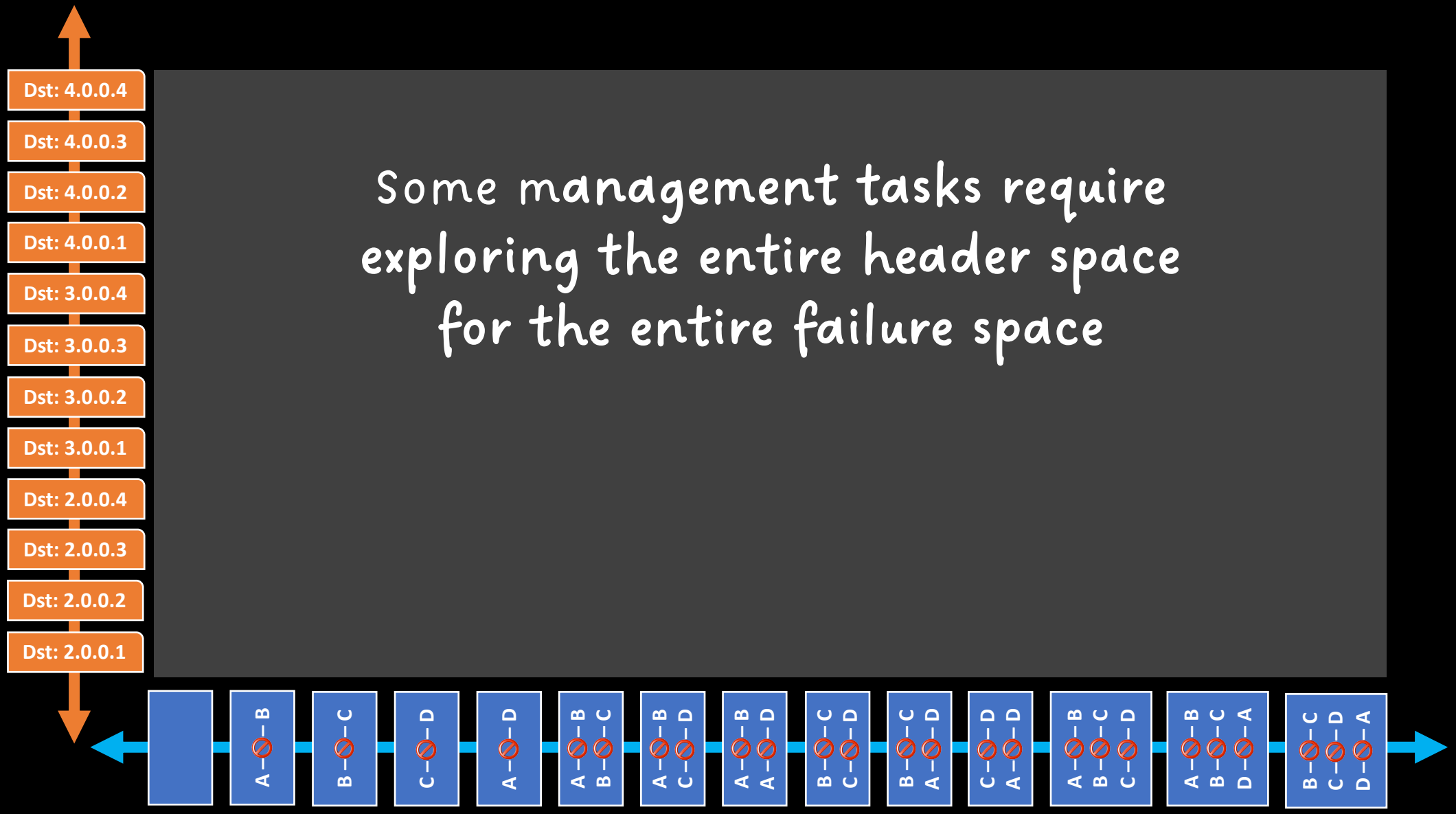
The second task required exploring
a single header
for the entire failure space

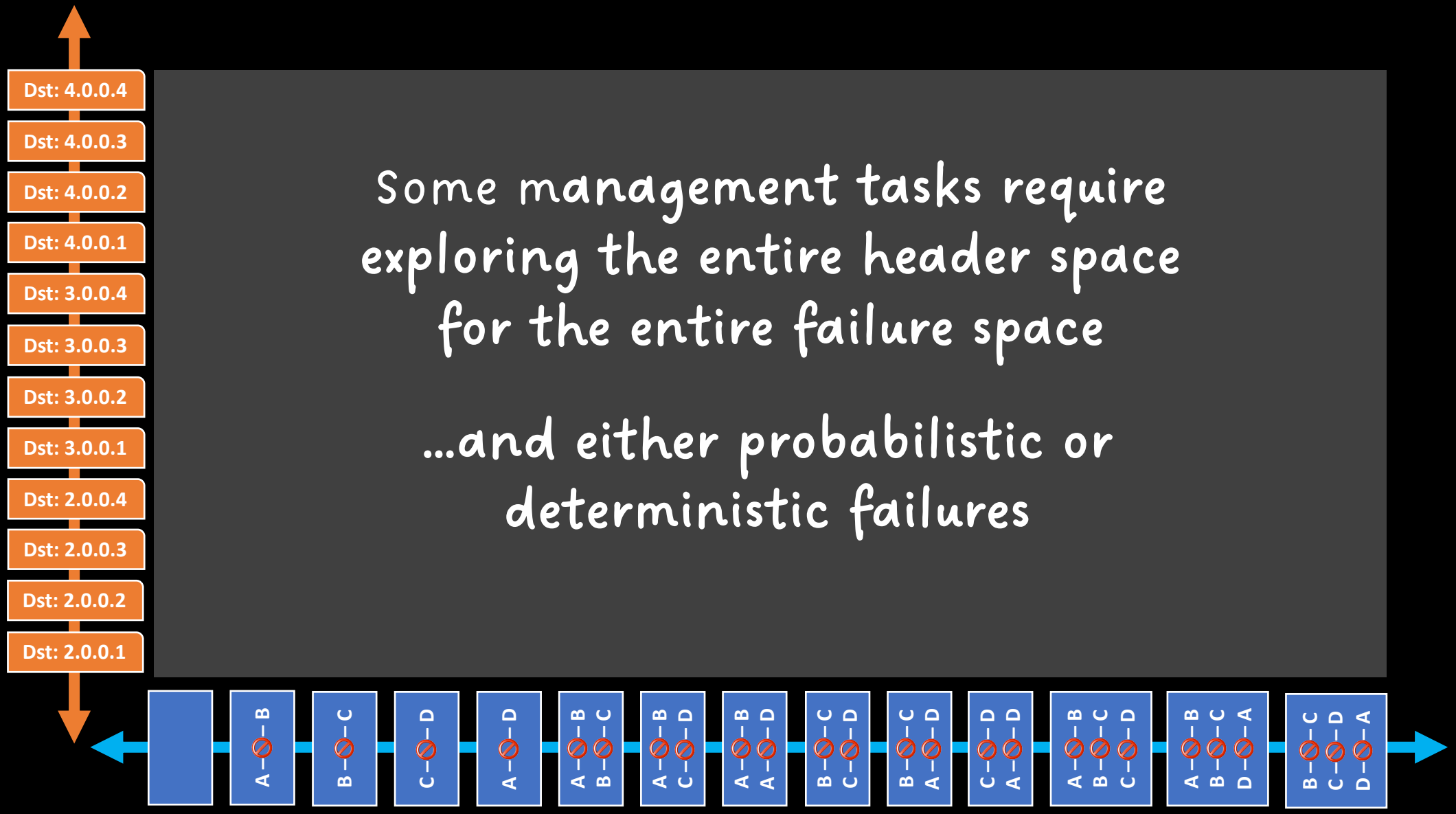


The second task required exploring
a single header
for the entire failure space

...and either probabilistic or
deterministic failures

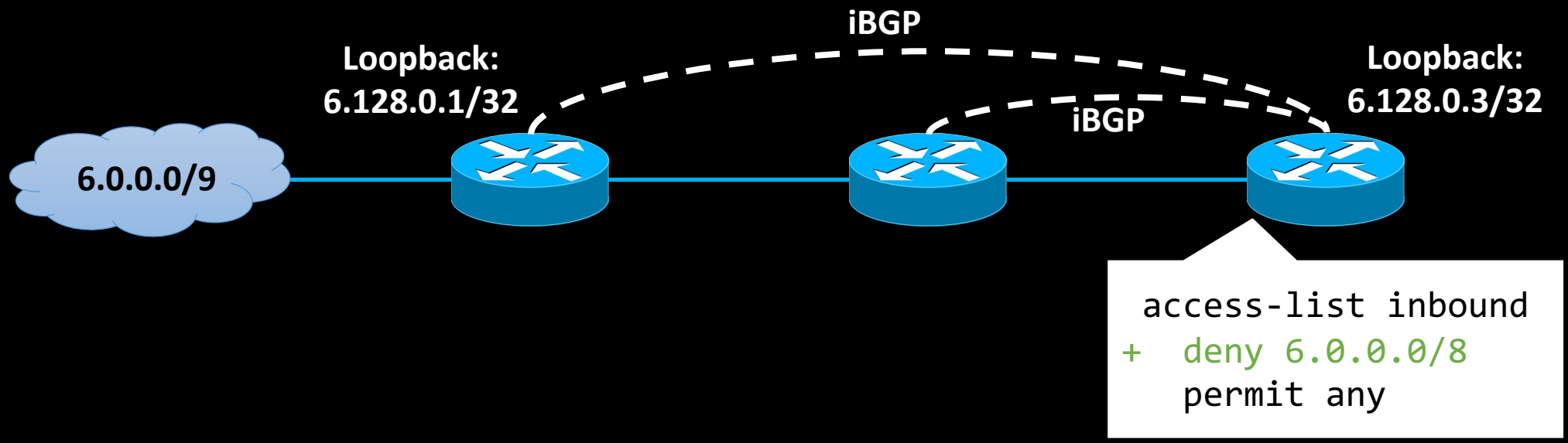



Dst: 4.0.0.4
Dst: 4.0.0.3
Dst: 4.0.0.2
Dst: 4.0.0.1
Dst: 3.0.0.4
Dst: 3.0.0.3
Dst: 3.0.0.2
Dst: 3.0.0.1
Dst: 2.0.0.4
Dst: 2.0.0.3
Dst: 2.0.0.2
Dst: 2.0.0.1
Some management tasks require exploring the entire header space for the entire failure space
A—B
B—C
C—D
A—D
A—B B—C
A—B C—D
A—B A—D
B—C C—D
B—C A—D
C—D A—D
A—B B—C C—D
A—B B—C D—A
B—C C—D D—A


Dst: 4.0.0.4
Dst: 4.0.0.3
Dst: 4.0.0.2
Dst: 4.0.0.1
Dst: 3.0.0.4
Dst: 3.0.0.3
Dst: 3.0.0.2
Dst: 3.0.0.1
Dst: 2.0.0.4
Dst: 2.0.0.3
Dst: 2.0.0.2
Dst: 2.0.0.1
Some management tasks require exploring the entire header space for the entire failure space
...and either probabilistic or deterministic failures
A—B
B—C
C—D
A—D
A—B B—C
A—B C—D
A—B A—D
B—C C—D
B—C A—D
C—D A—D
A—B B—C C—D
A—B B—C D—A
B—C C—D D—A

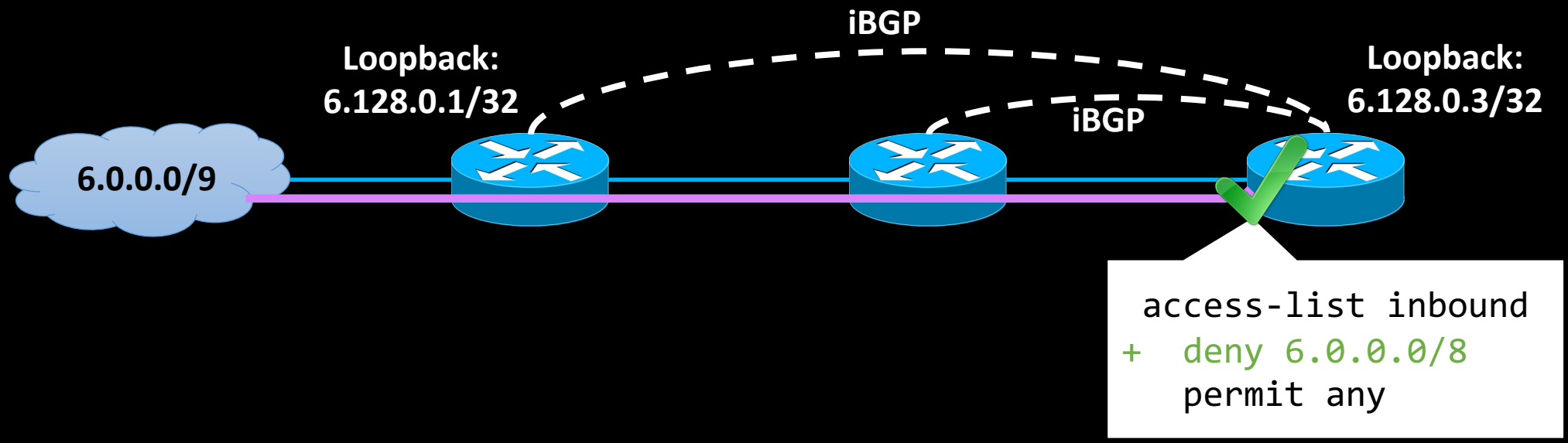
Some management tasks require exploring the product space

① Verify a proposed configuration change has no side-effects



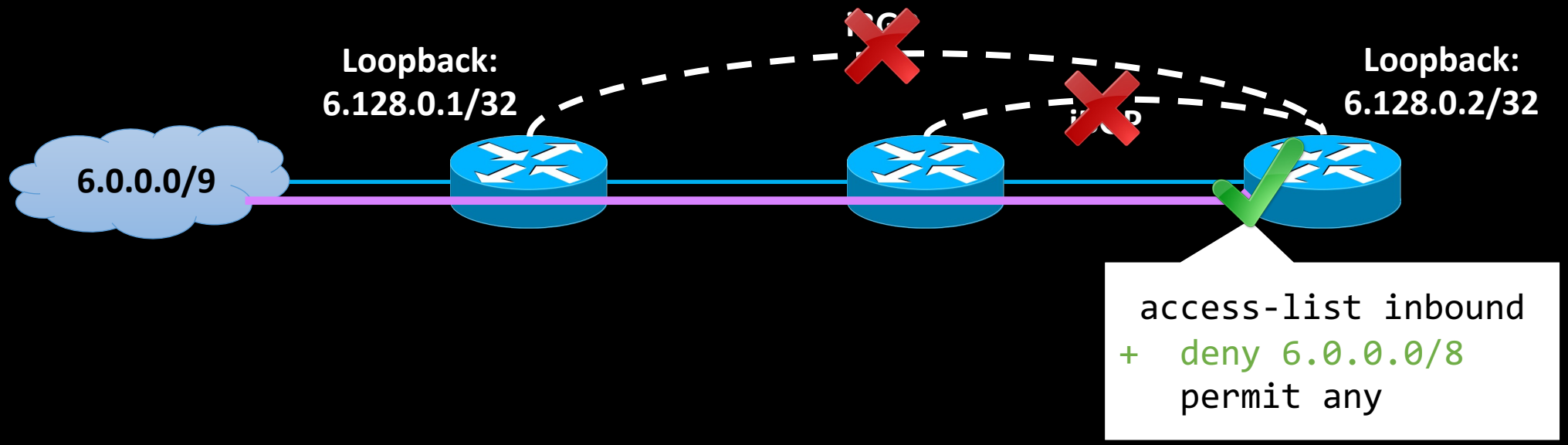
Some management tasks require exploring the product space

① Verify a proposed configuration change has no side-effects



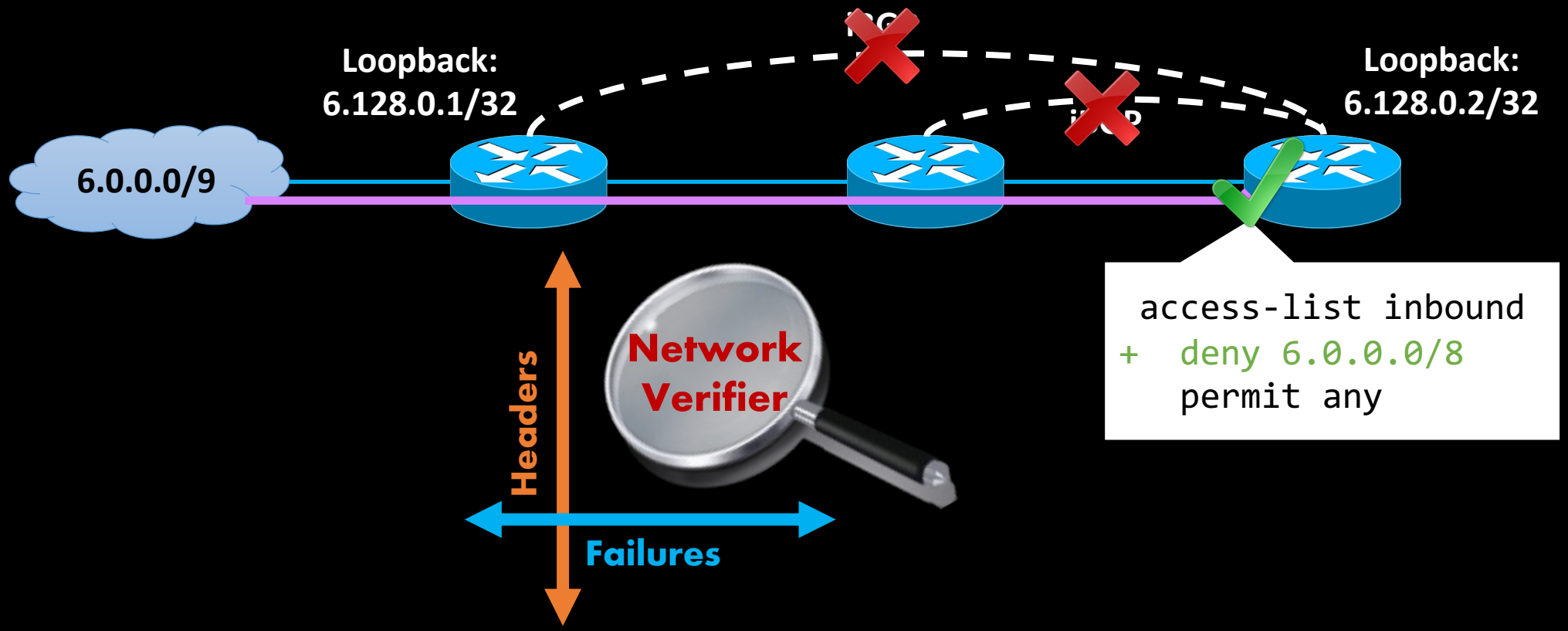
Some management tasks require exploring the product space

① Verify a proposed configuration change has no side-effects



Some management tasks require exploring the product space

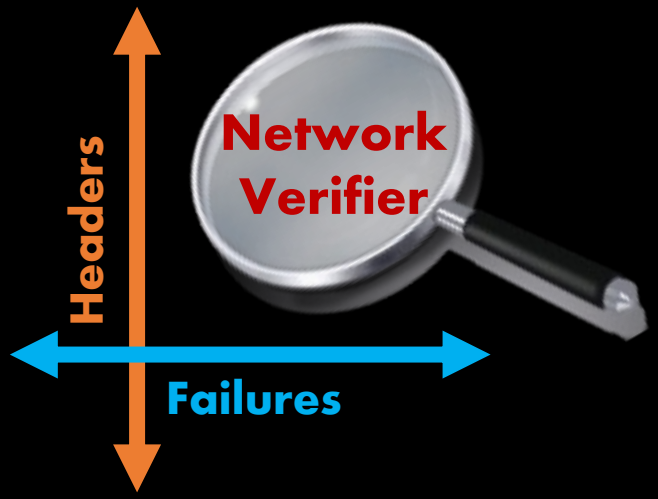
① Verify a proposed configuration change has no side-effects



Some management tasks require exploring the product space

① Verify a proposed configuration change has no side-effects

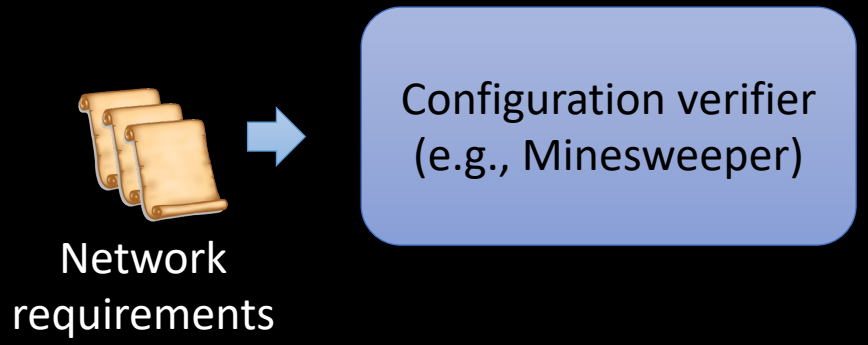
② Mine network requirements (e.g., Config2Spec)



Some management tasks require exploring the product space

① Verify a proposed configuration change has no side-effects

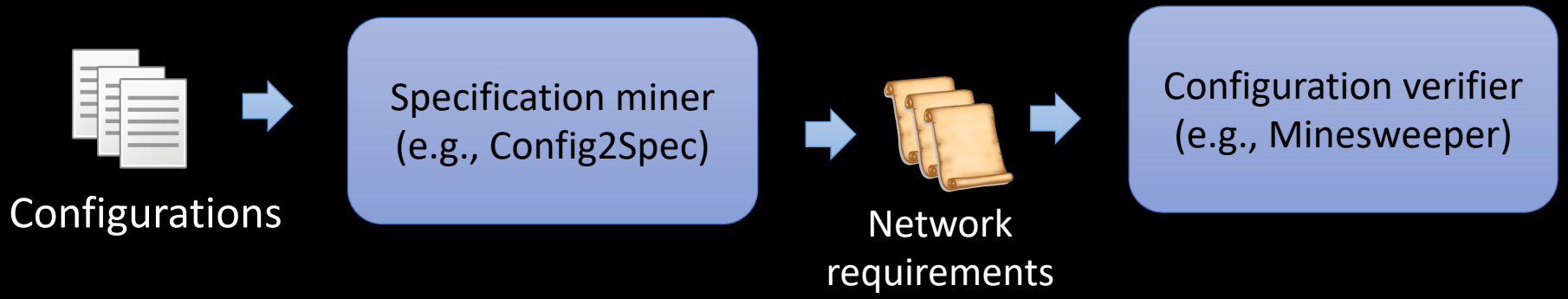
② Mine network requirements (e.g., Config2Spec)



Some management tasks require exploring the product space

① Verify a proposed configuration change has no side-effects

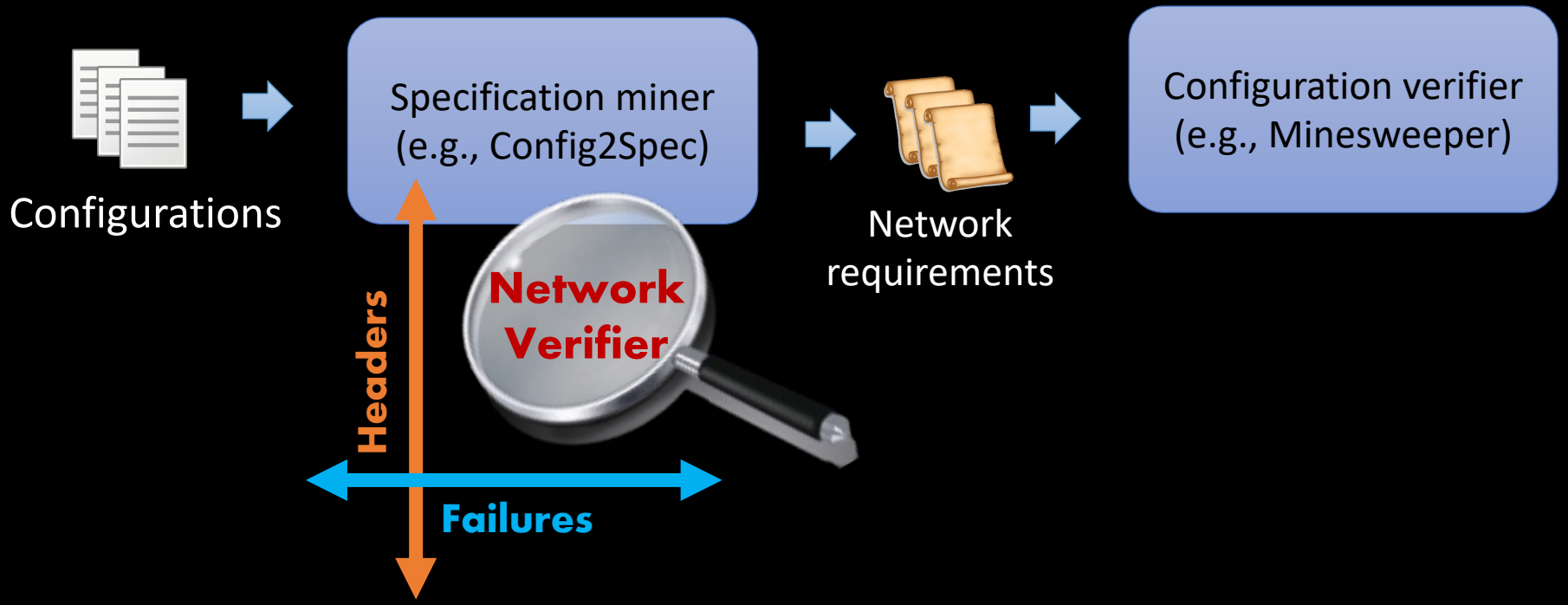
② Mine network requirements (e.g., Config2Spec)

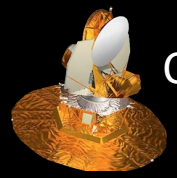


Some management tasks require exploring the product space

① Verify a proposed configuration change has no side-effects

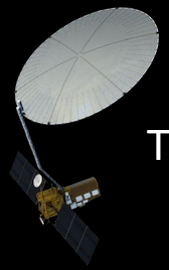
② Mine network requirements (e.g., Config2Spec)

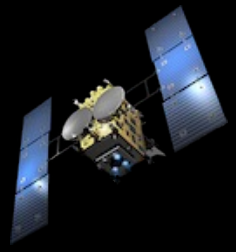


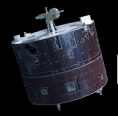

Config2Spec


ARC


Bagpipe


Tiramisu


Hoyan


Plankton

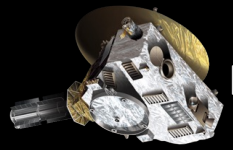

ERA


ShapeShifter

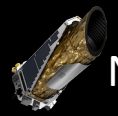

Minesweeper

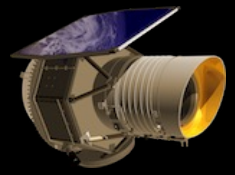

Origami


DNA

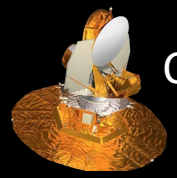

Bagpipe


NV

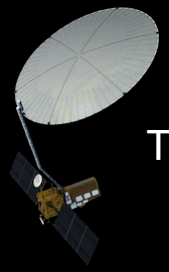

NetDice

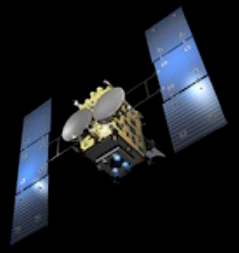

ProbNV

Can we use the verifiers we already have?


Config2Spec

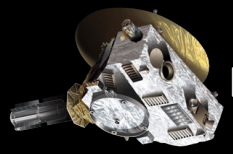

ARC


Tiramisu

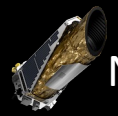

Hoyan


Minesweeper


Origami

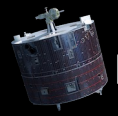

Bagpipe


NV


NetDice


ProbNV


Bagpipe


Plankton


ERA

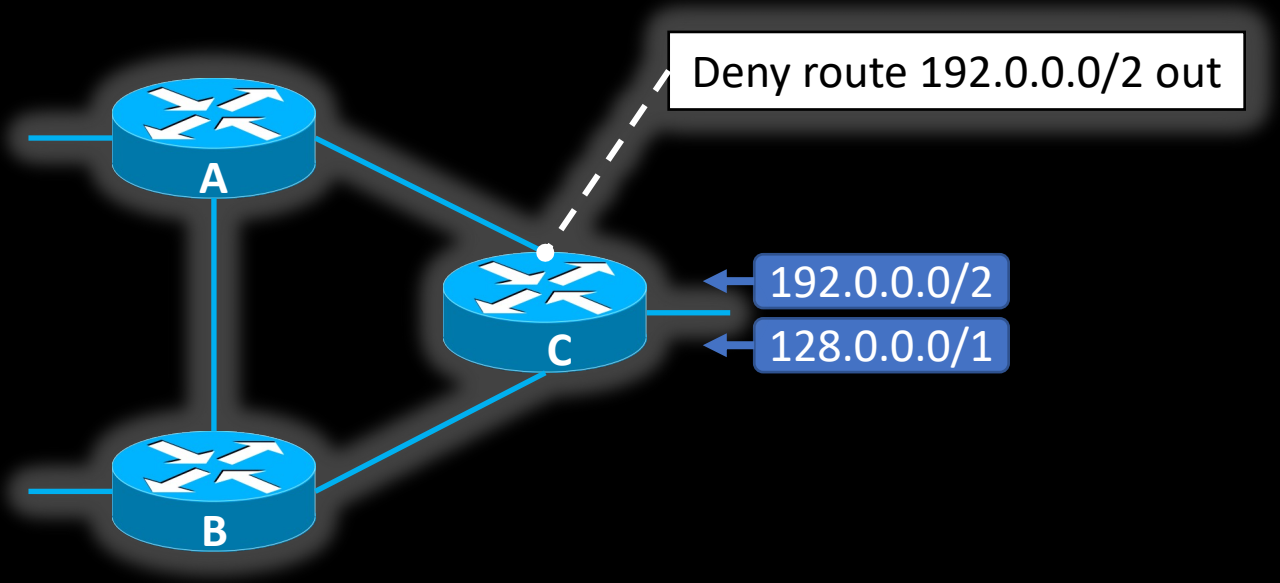

ShapeShifter


DNA

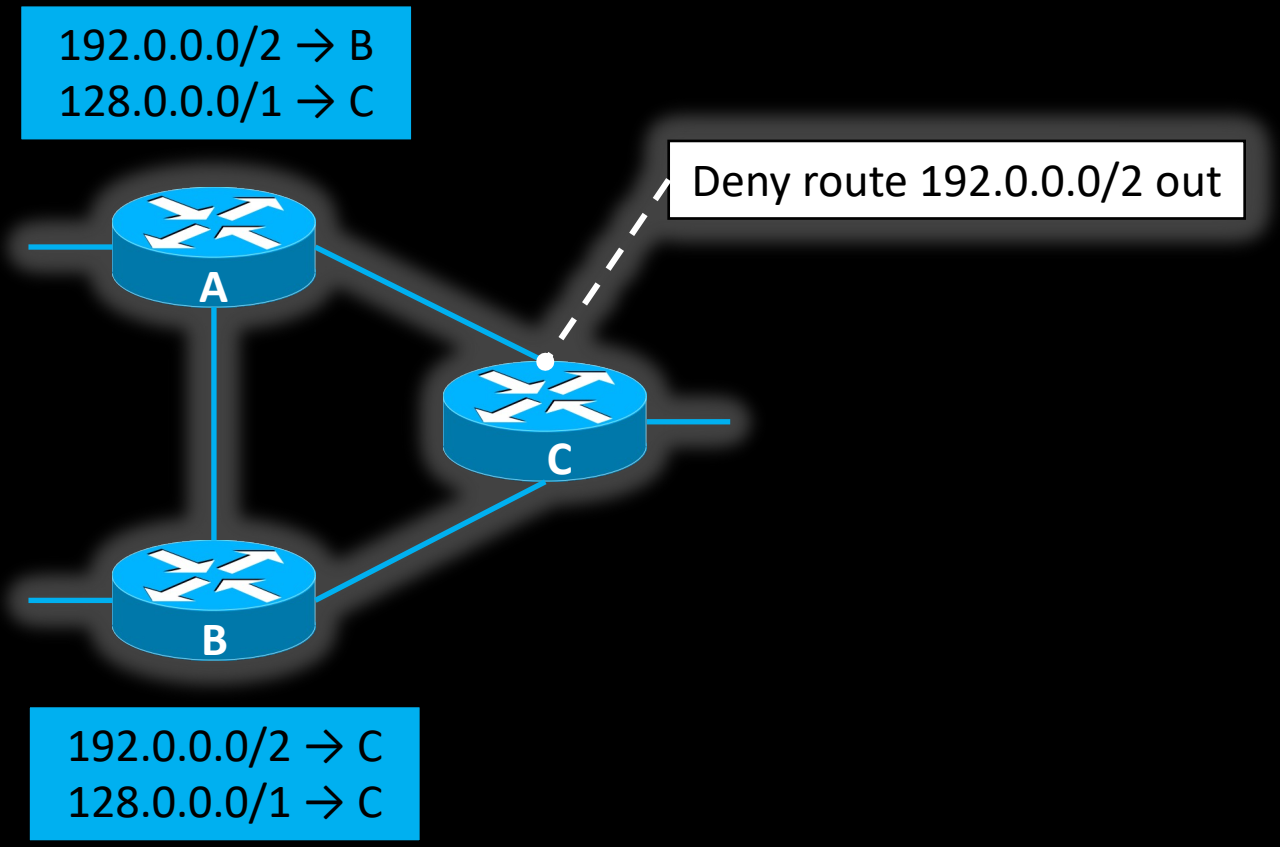
Divide headers into PECs

Efficient header space exploration

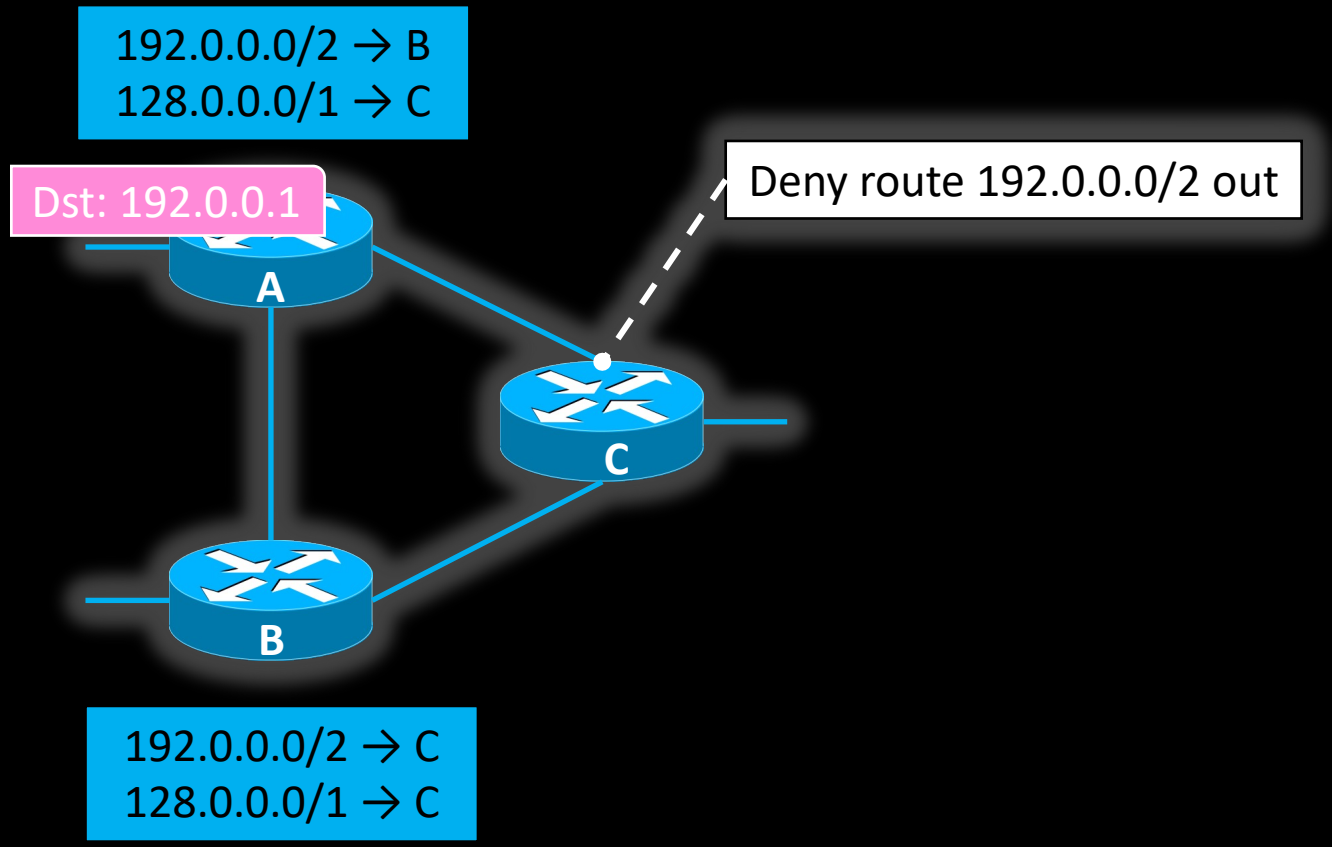
Packet Equivalence Classes

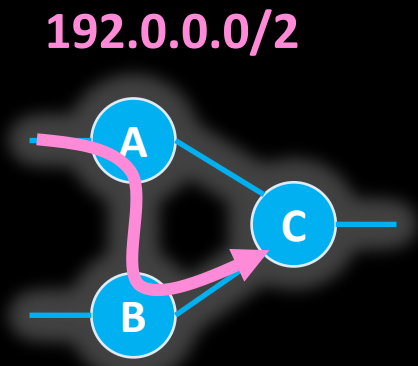


Packet Equivalence Classes

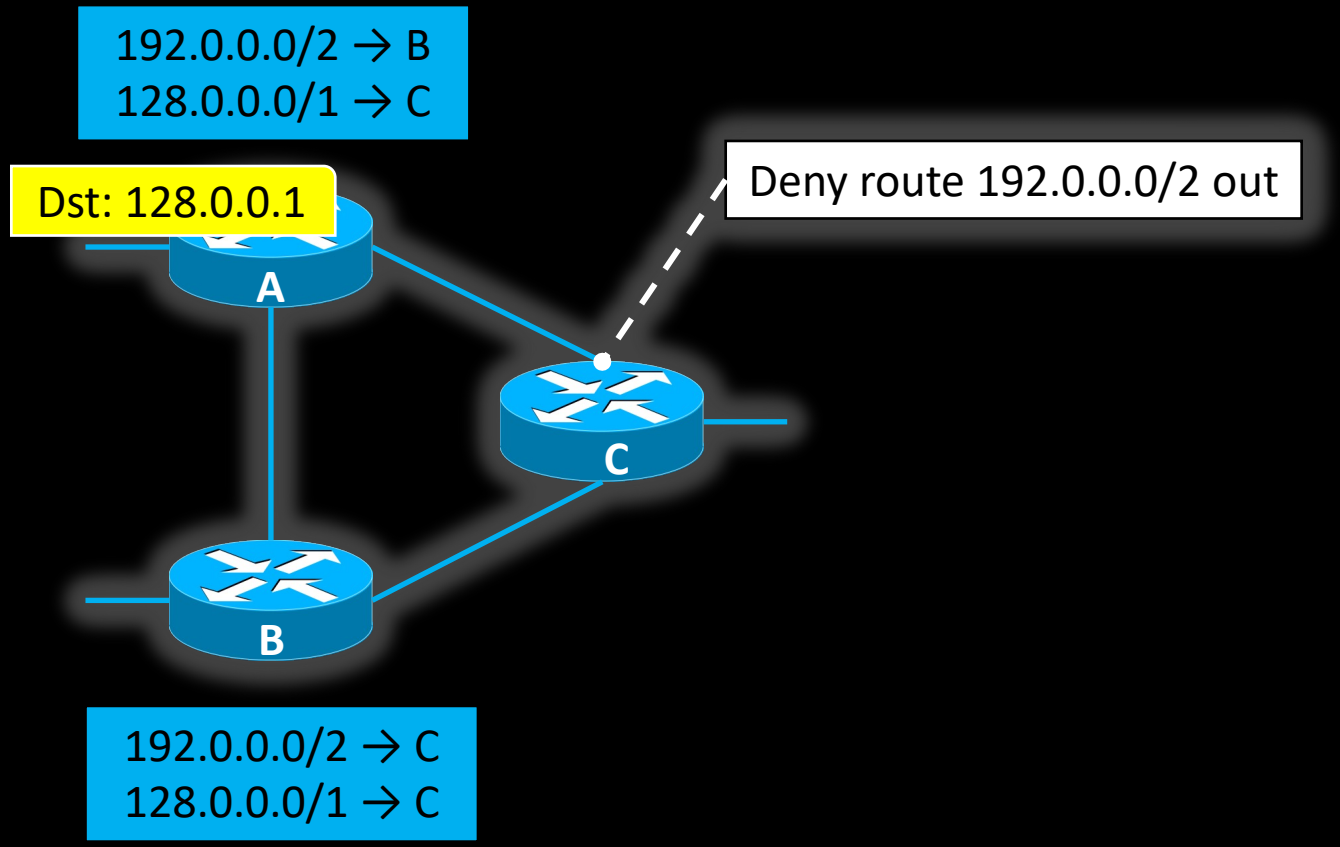


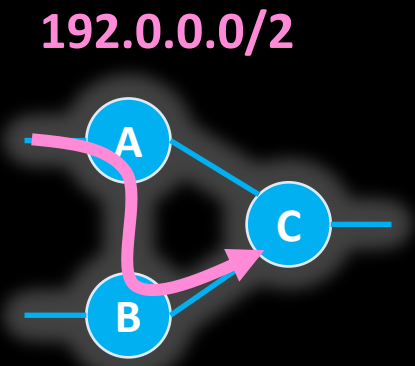
Packet Equivalence Classes

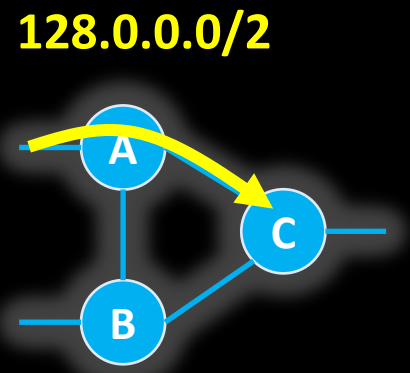




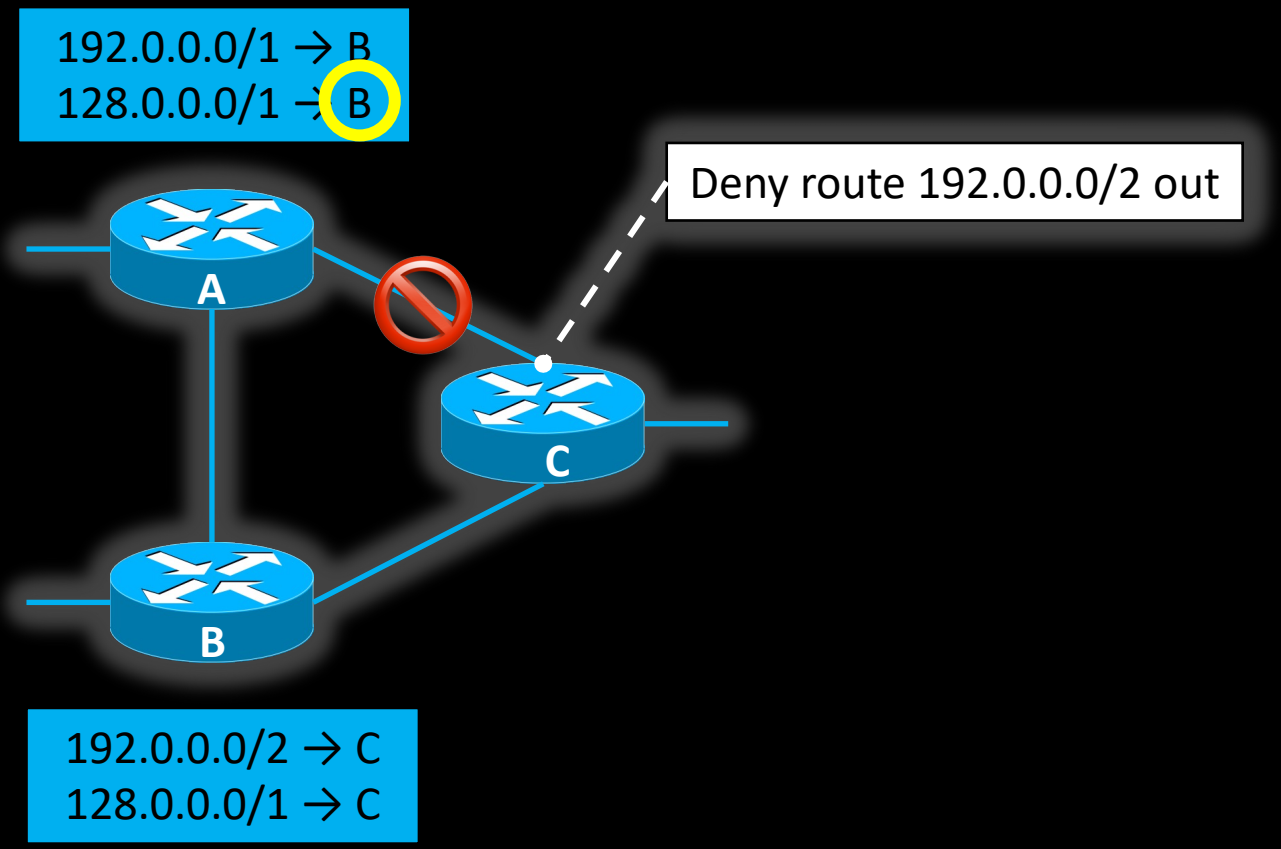
Packet Equivalence Classes

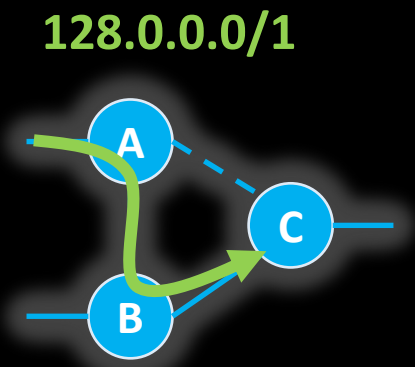


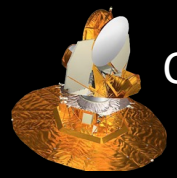




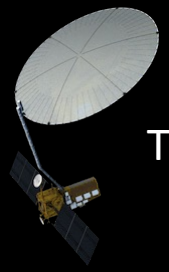
Packet Equivalence Classes

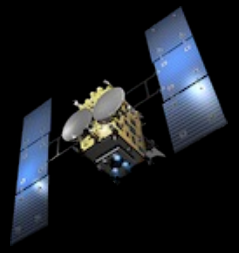





Config2Spec

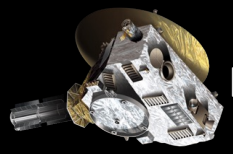

ARC


Tiramisu

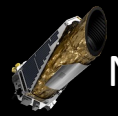

Hoyan


Minesweeper


Origami

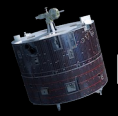

Bagpipe

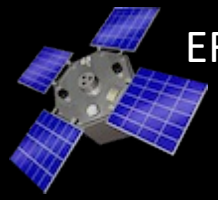

NV


NetDice


ProbNV


Bagpipe


Plankton


ERA

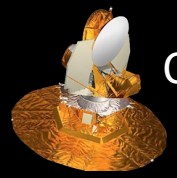

ShapeShifter


DNA

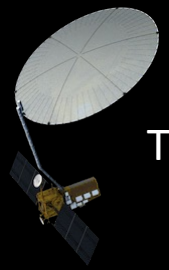
Divide headers into PECs

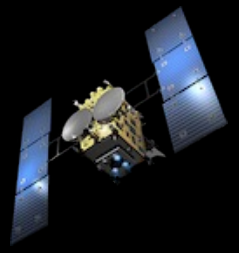
Efficient header space exploration

Inefficient failure space exploration


Config2Spec

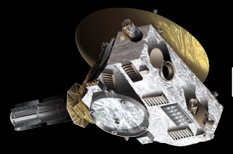

ARC


Tiramisu

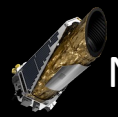

Hoyan


Minesweeper


Origami


Bagpipe


NV

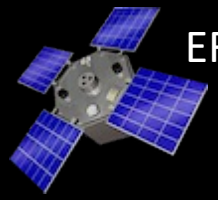

NetDice


ProbNV

Divide failures into FECs


Bagpipe


Plankton


ERA


ShapeShifter


DNA

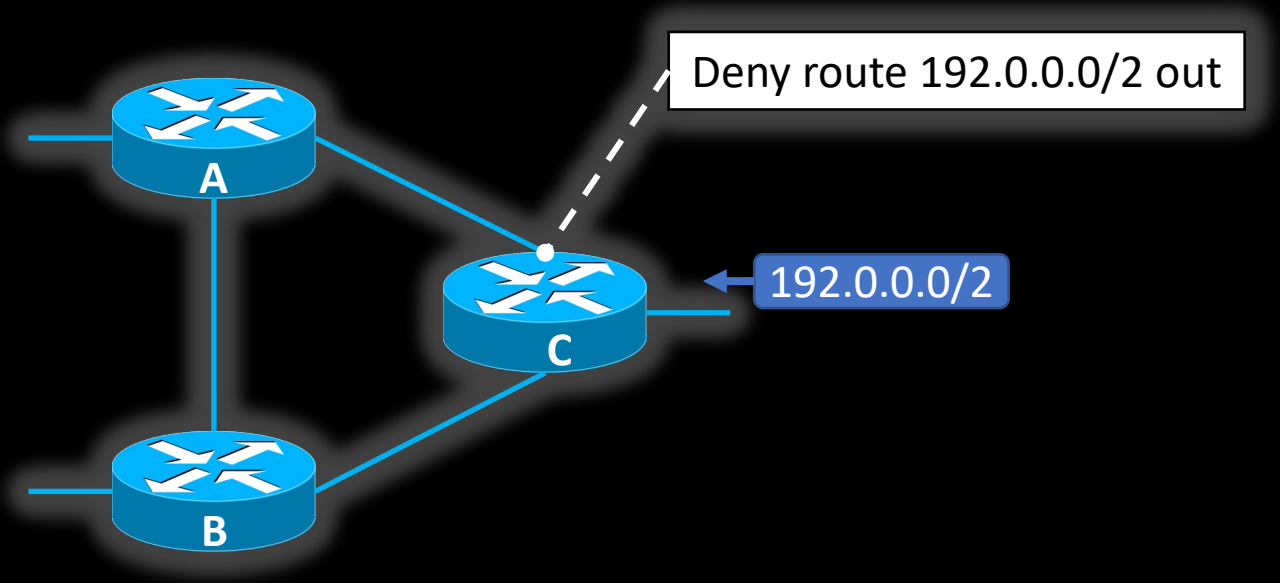
Divide headers into PECs

Efficient failure space exploration

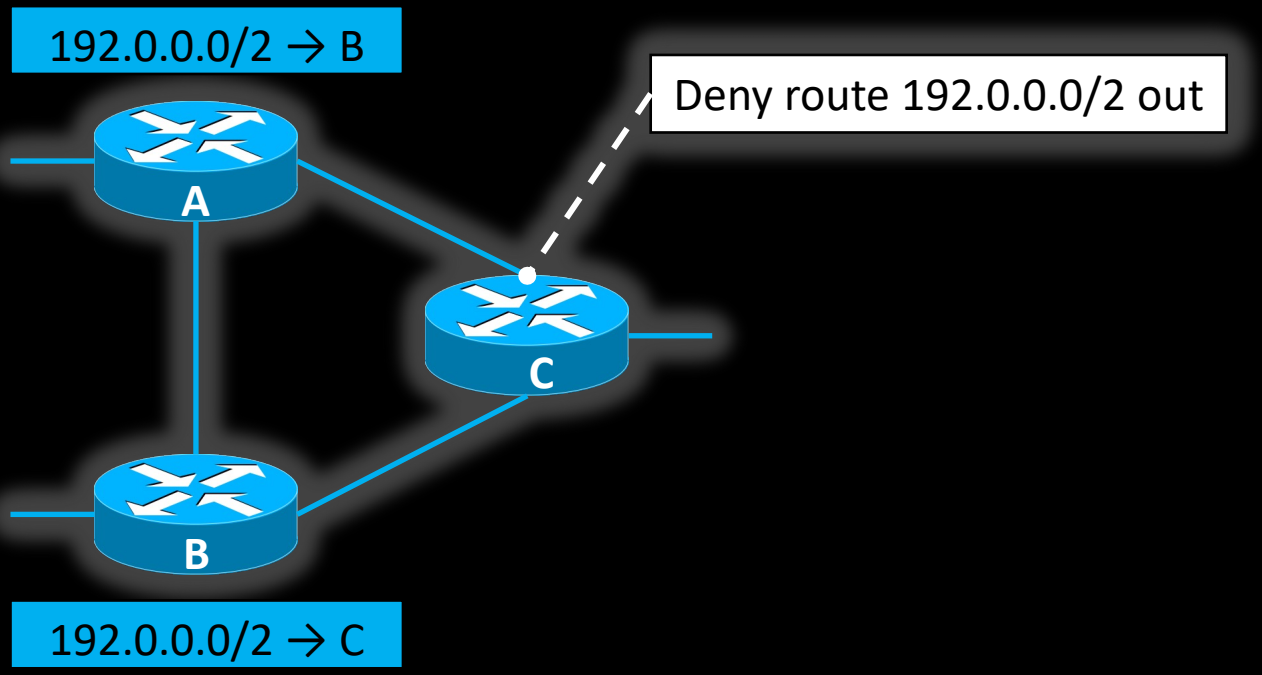
Efficient header space exploration

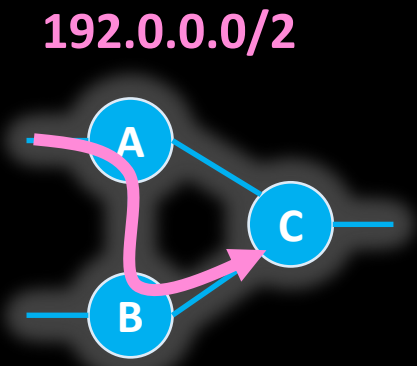
Inefficient failure space exploration

Failure Equivalence Classes

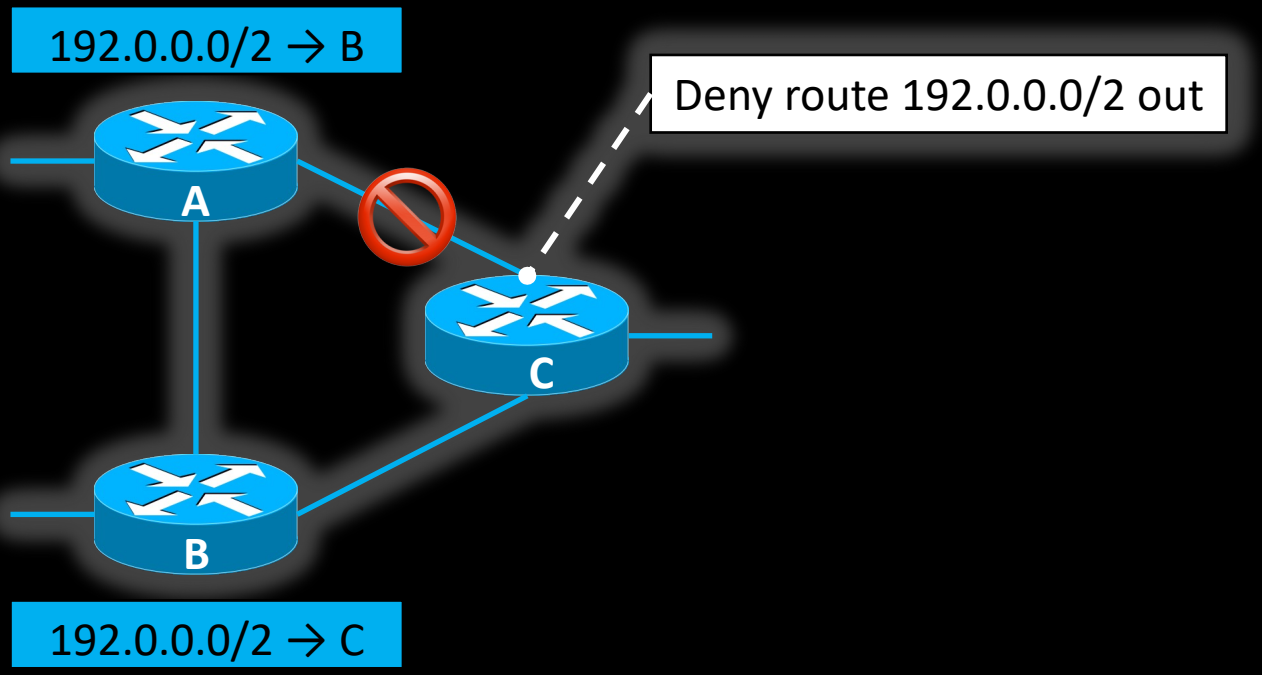


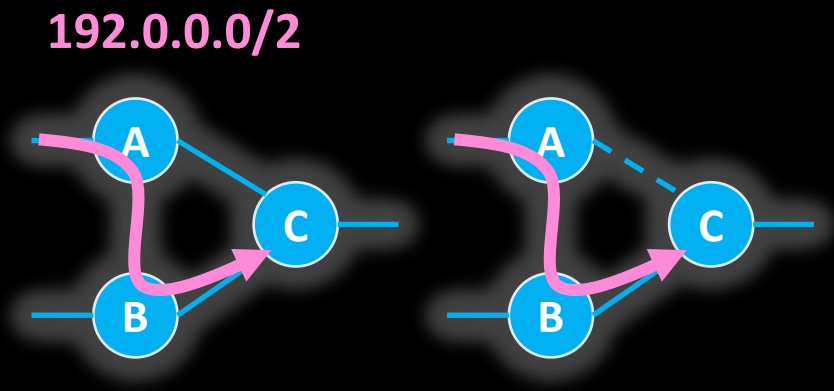
Failure Equivalence Classes



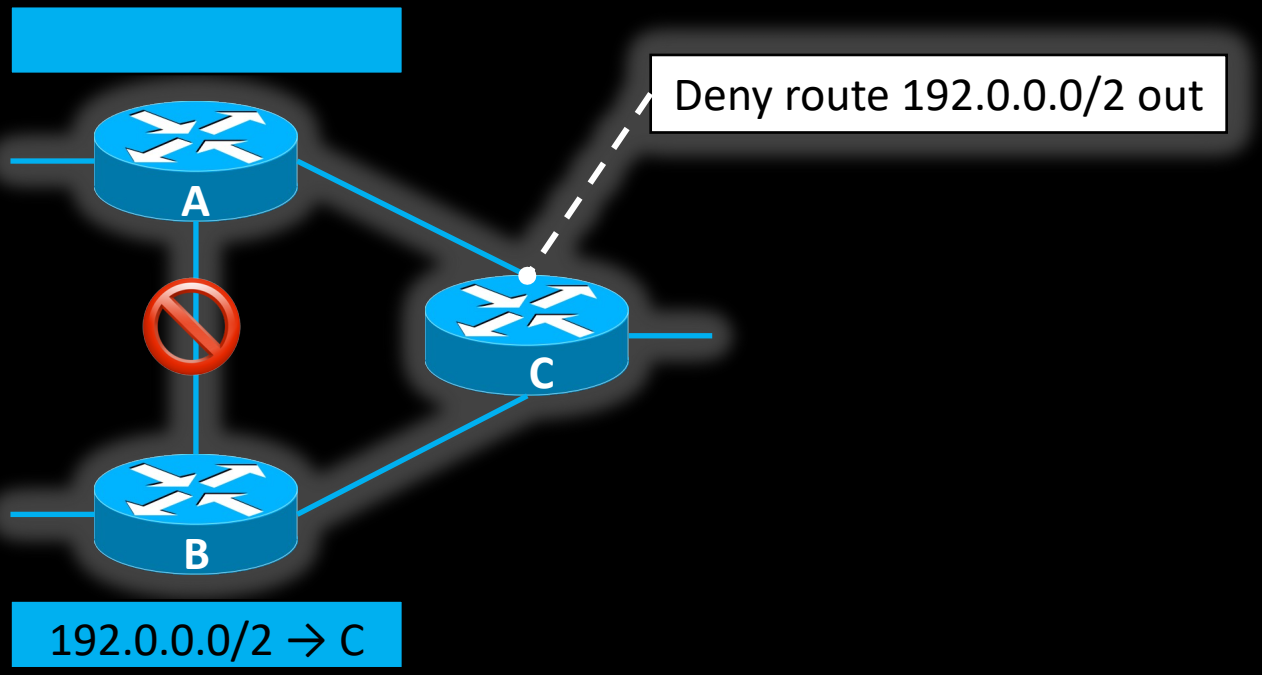


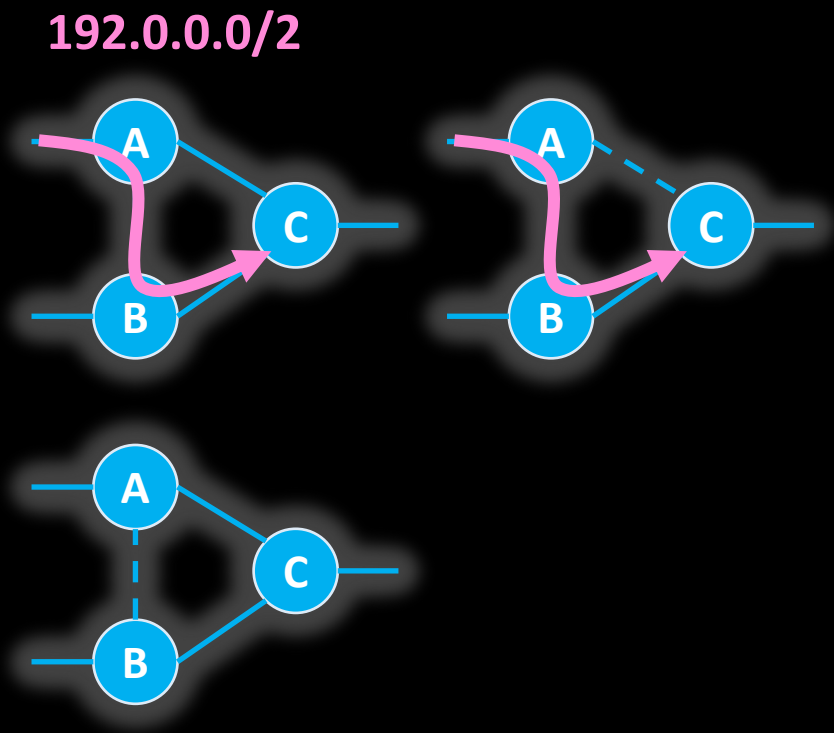
Failure Equivalence Classes



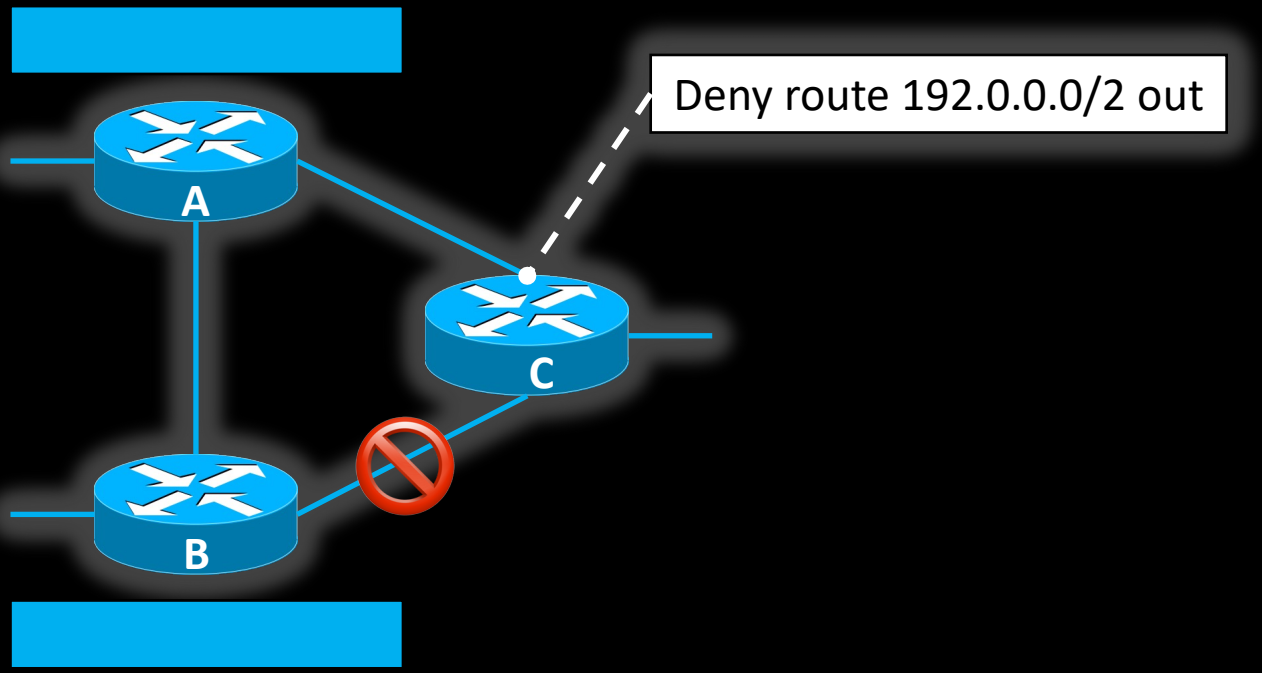


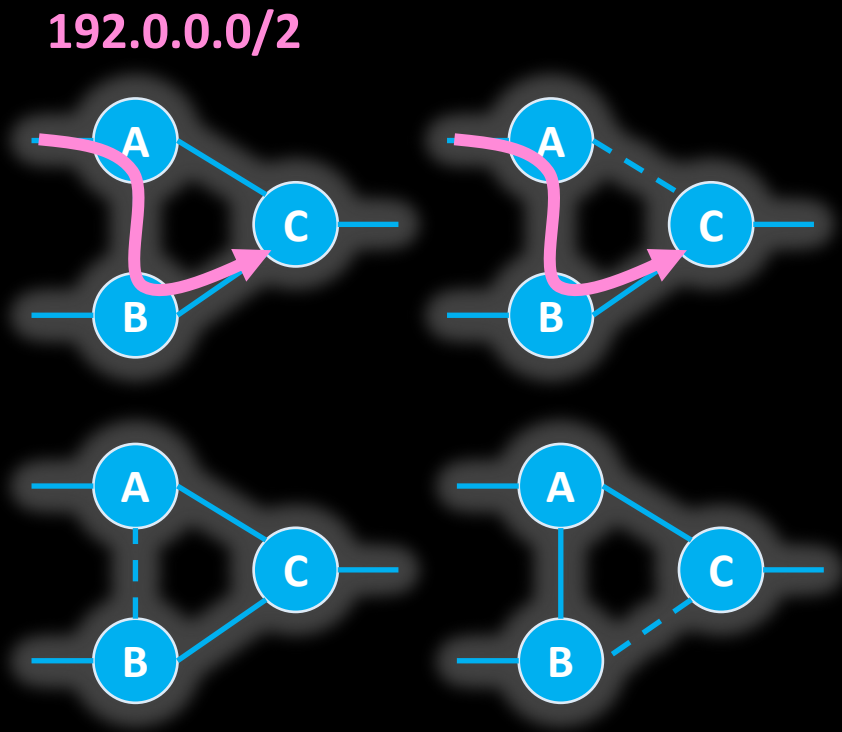
Failure Equivalence Classes



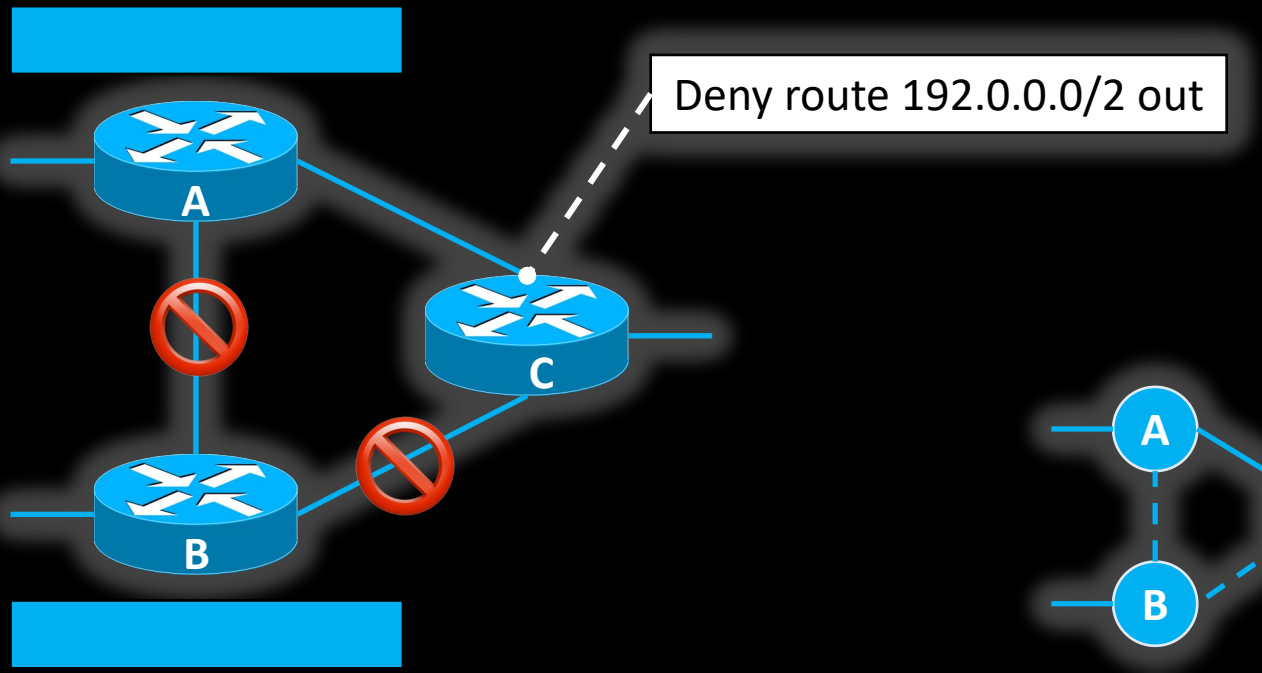


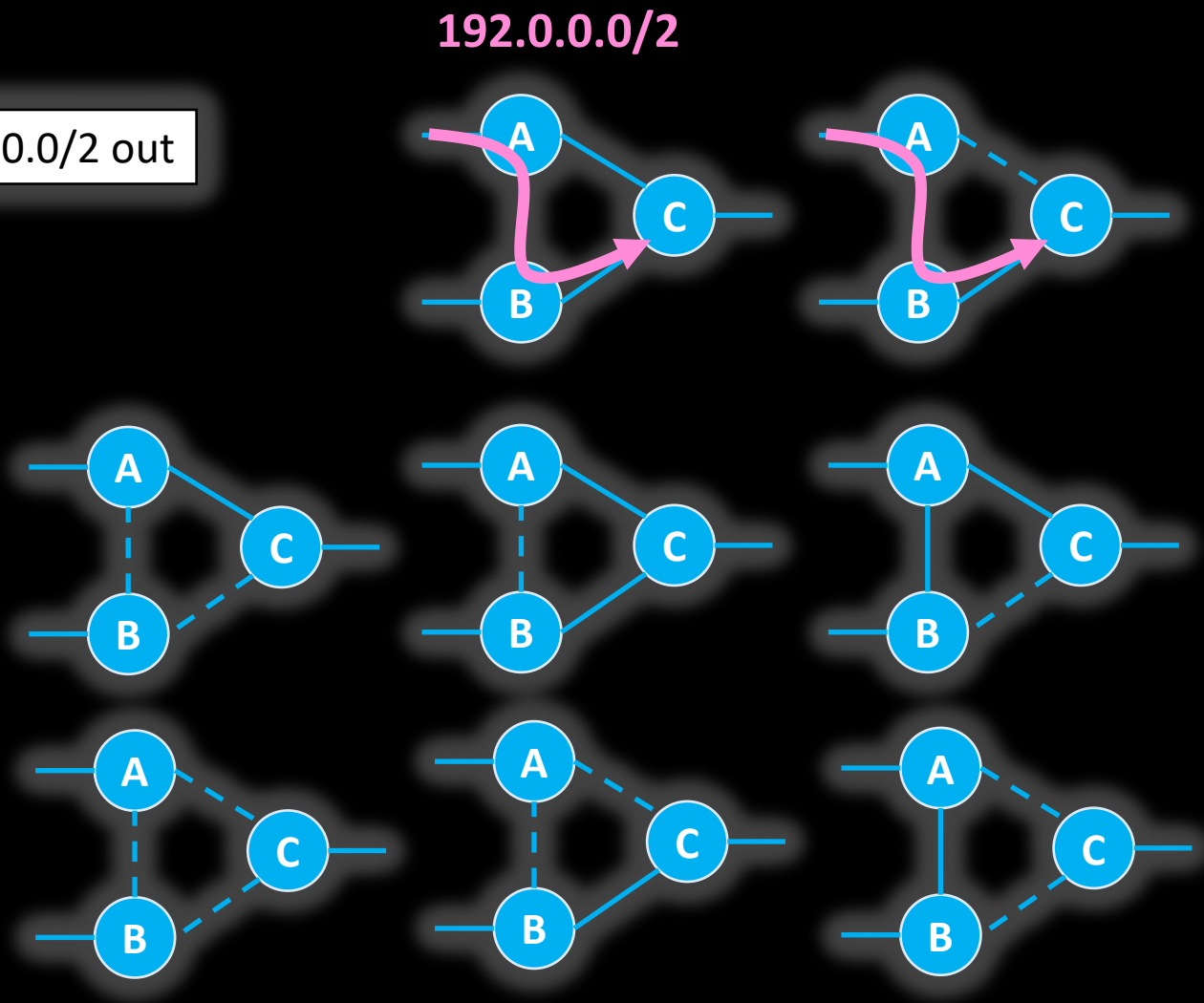
Failure Equivalence Classes



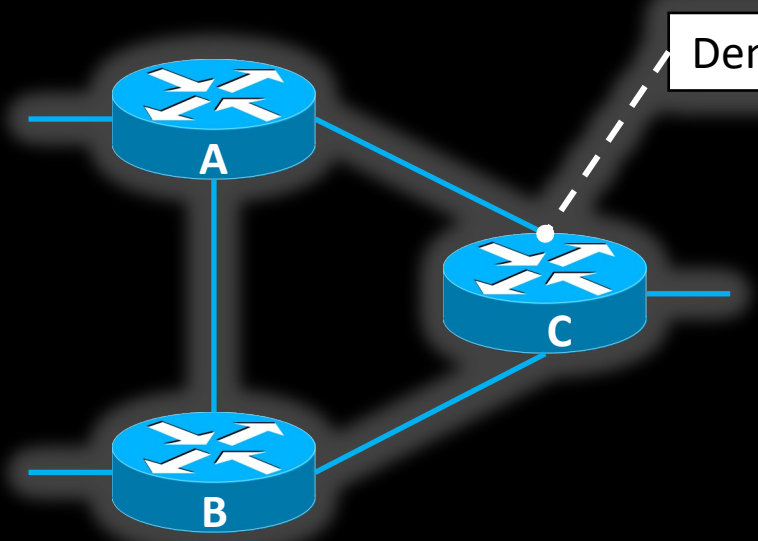


Failure Equivalence Classes



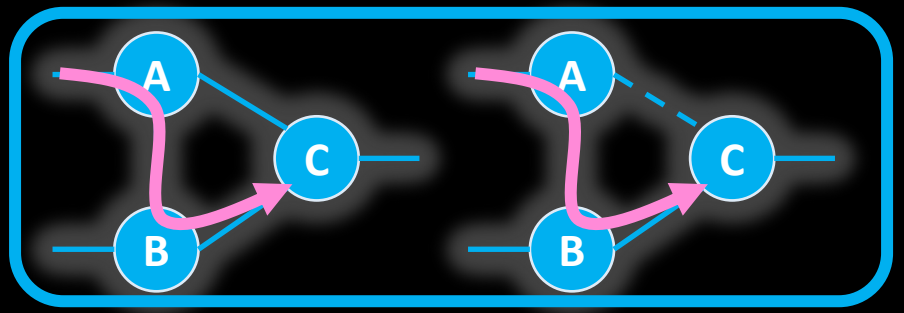


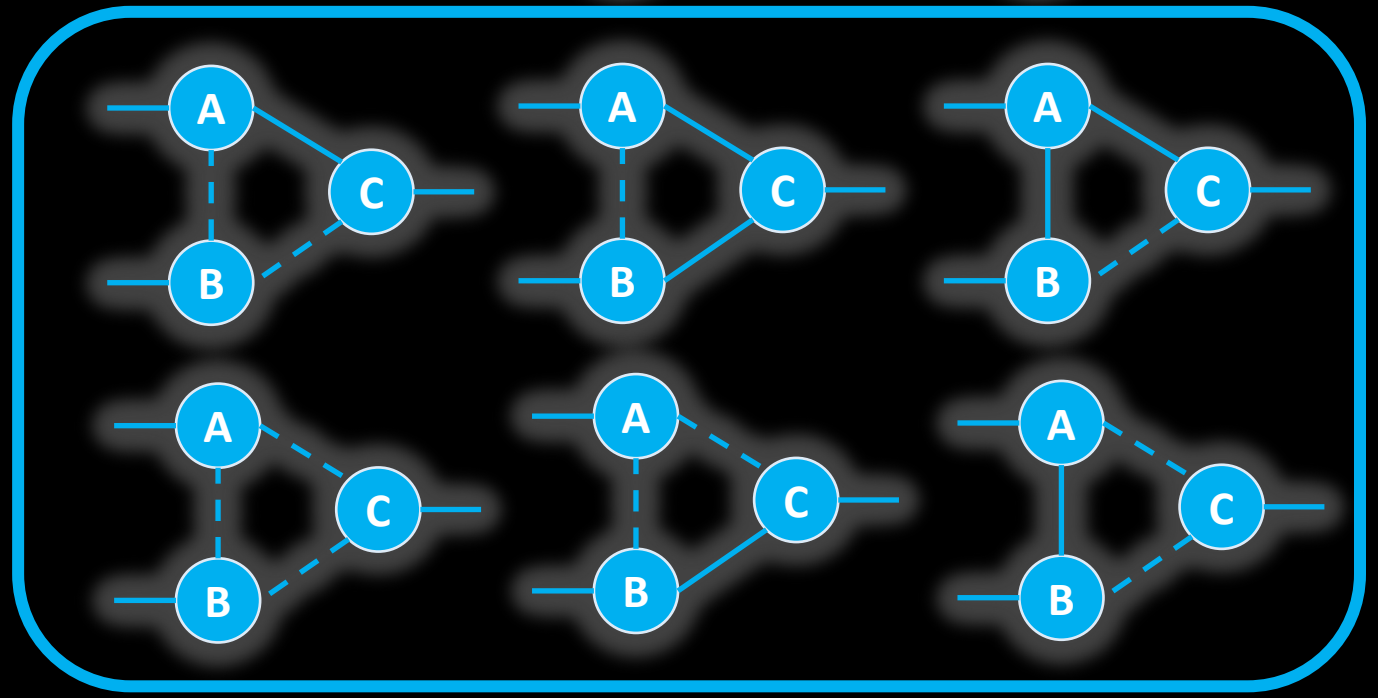
Failure Equivalence Classes



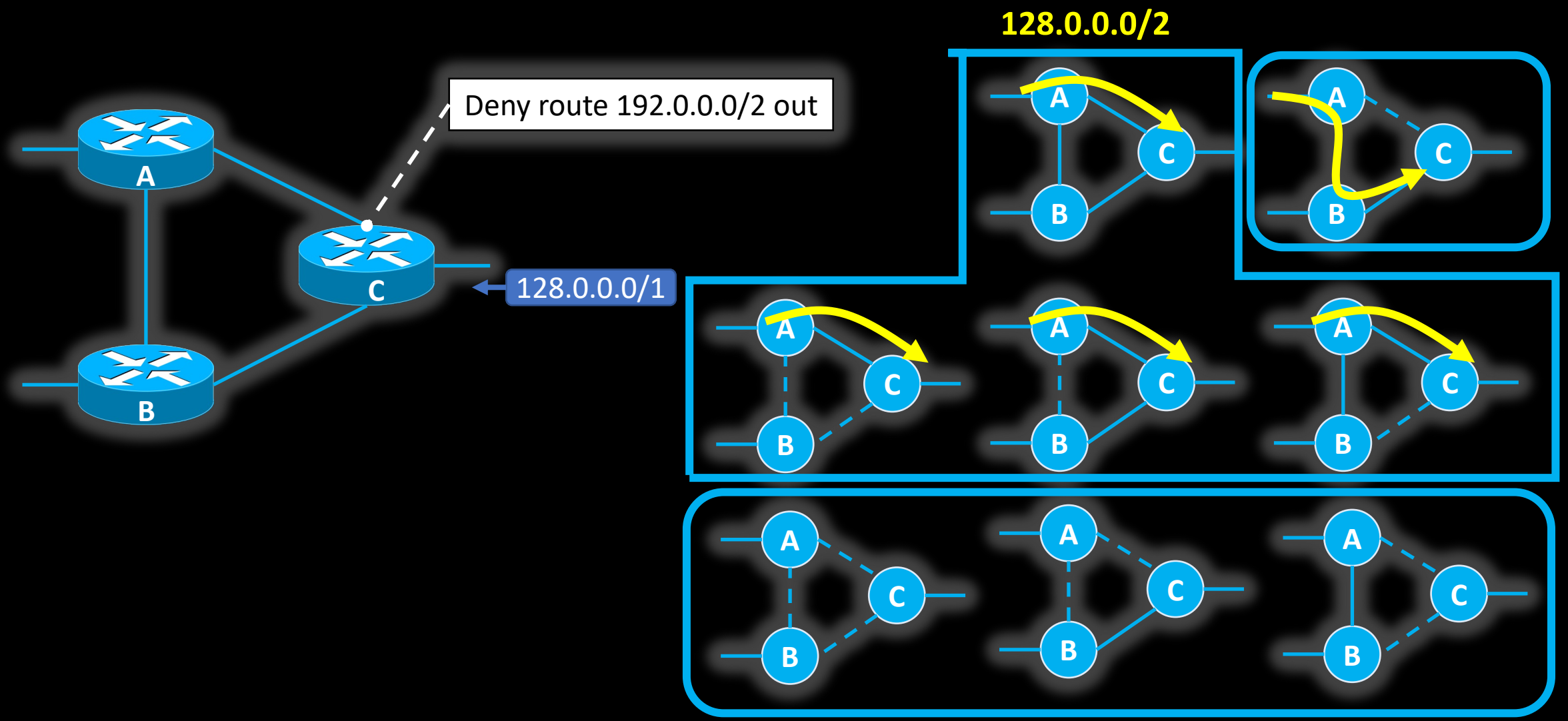
Deny route 192.0.0.0/2 out

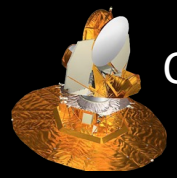
192.0.0.0/2



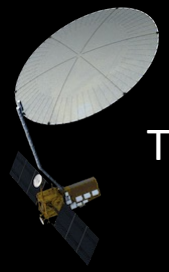


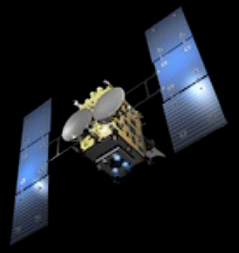
Failure Equivalence Classes




Config2Spec

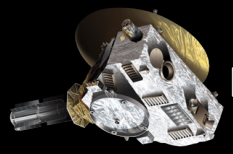

ARC


Tiramisu

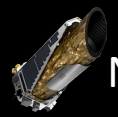

Hoyan


Minesweeper


Origami


Bagpipe

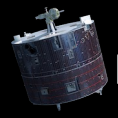

NV

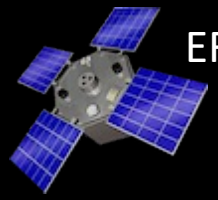

NetDice


ProbNV

Divide failures into FECs


Bagpipe


Plankton


ERA

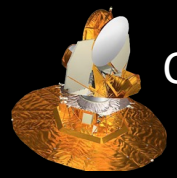

ShapeShifter


DNA

Divide headers into PECs

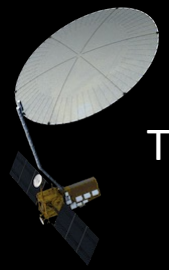
Inefficient header space exploration

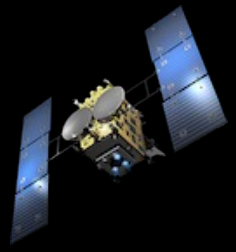
Efficient failure space exploration

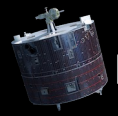

Config2Spec

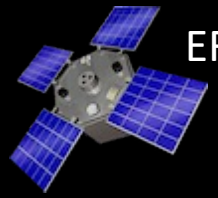

ARC


Bagpipe


Tiramisu


Hoyan


Plankton

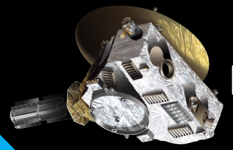

ERA


Minesweeper


ShapeShifter


Origami


DNA


Bagpipe


NV

Deterministic failures


NetDice


ProbNV

Probabilistic failures


SRE

Scales to the product space
of headers and failures
by computing PFECs

Scales to the product space
of headers and failures
by computing PFECs

Generalizes to deterministic and
probabilistic failures by delaying
binding to a failure model

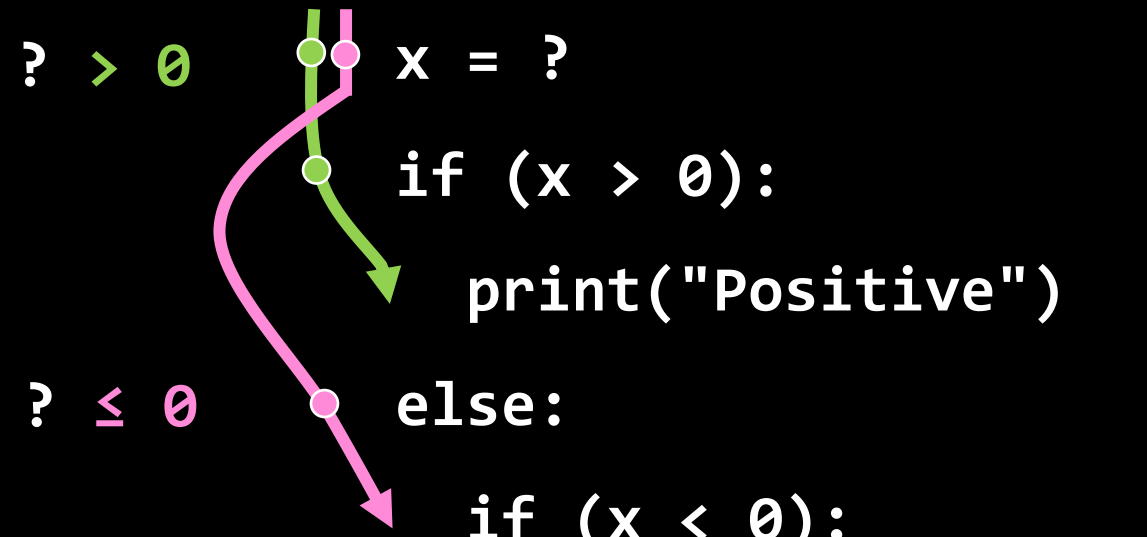
Basic idea: Symbolic execution

```
x = input("Integer:")
if (x > 0):
  print("Positive")
else:
  if (x < 0):
    print("Negative")
  else:
    print("Zero")
```

Basic idea: Symbolic execution

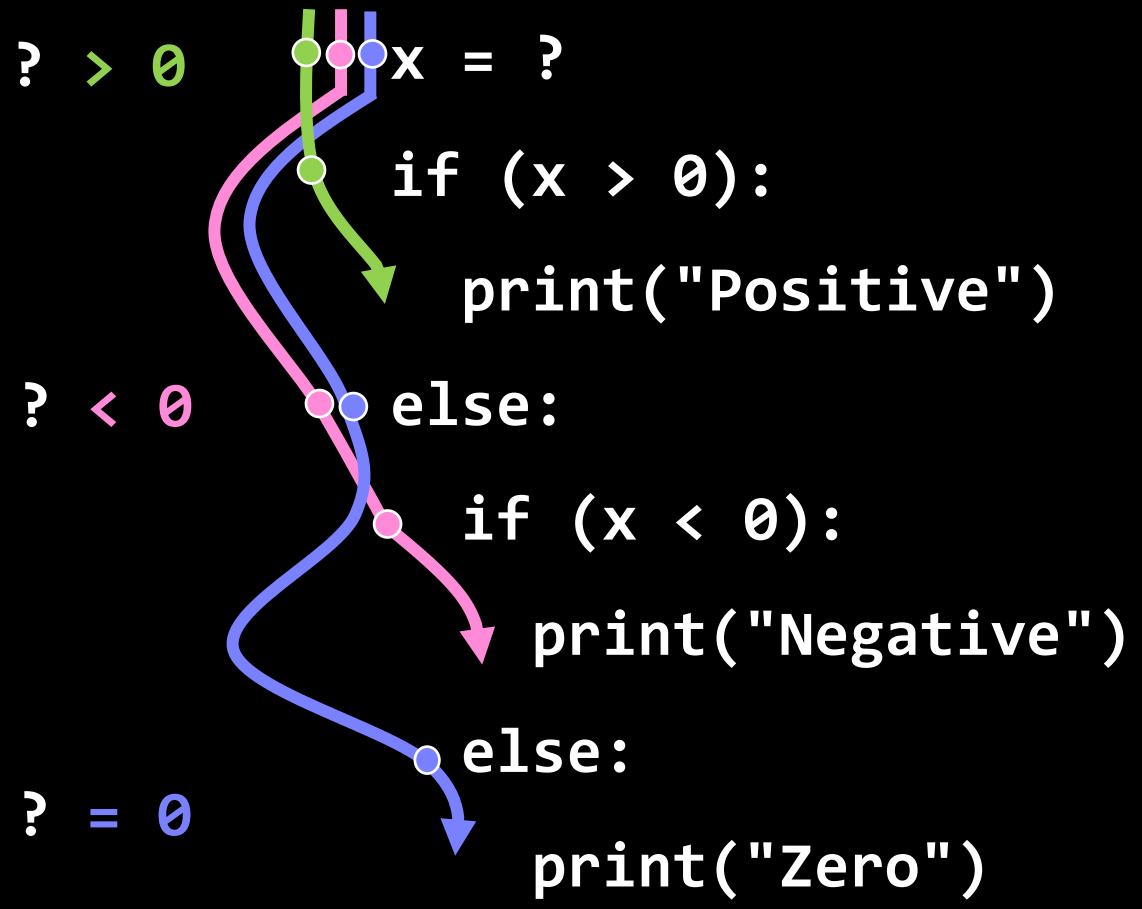
```
x = ?
if (x > 0):
  print("Positive")
else:
  if (x < 0):
    print("Negative")
  else:
    print("Zero")
```

Basic idea: Symbolic execution




```
x = ?
if (x > 0):
  print("Positive")
else:
  if (x < 0):
    print("Negative")
  else:
    print("Zero")
```

Basic idea: Symbolic execution



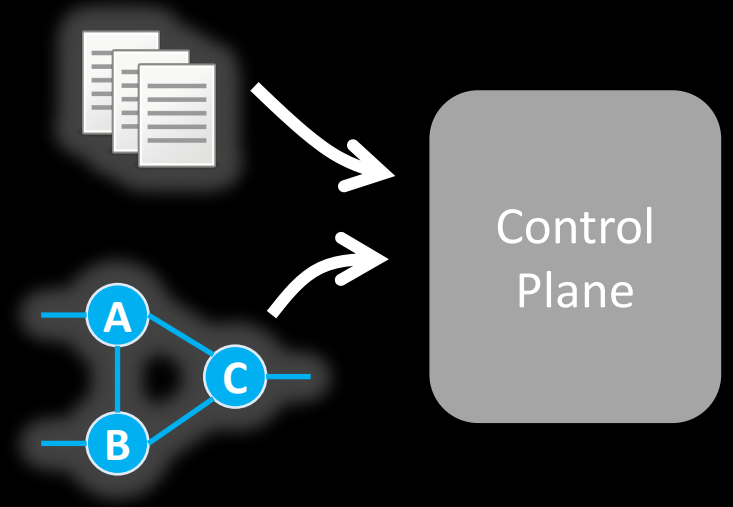
Symbolic execution of control and data planes

Control Plane

Data Plane

Symbolic execution of control and data planes

Configurations

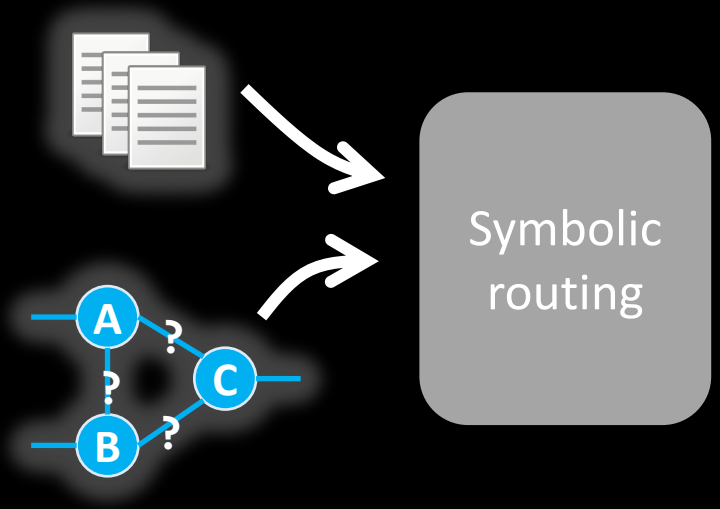


Available links

Data Plane

Symbolic execution of control and data planes

Configurations

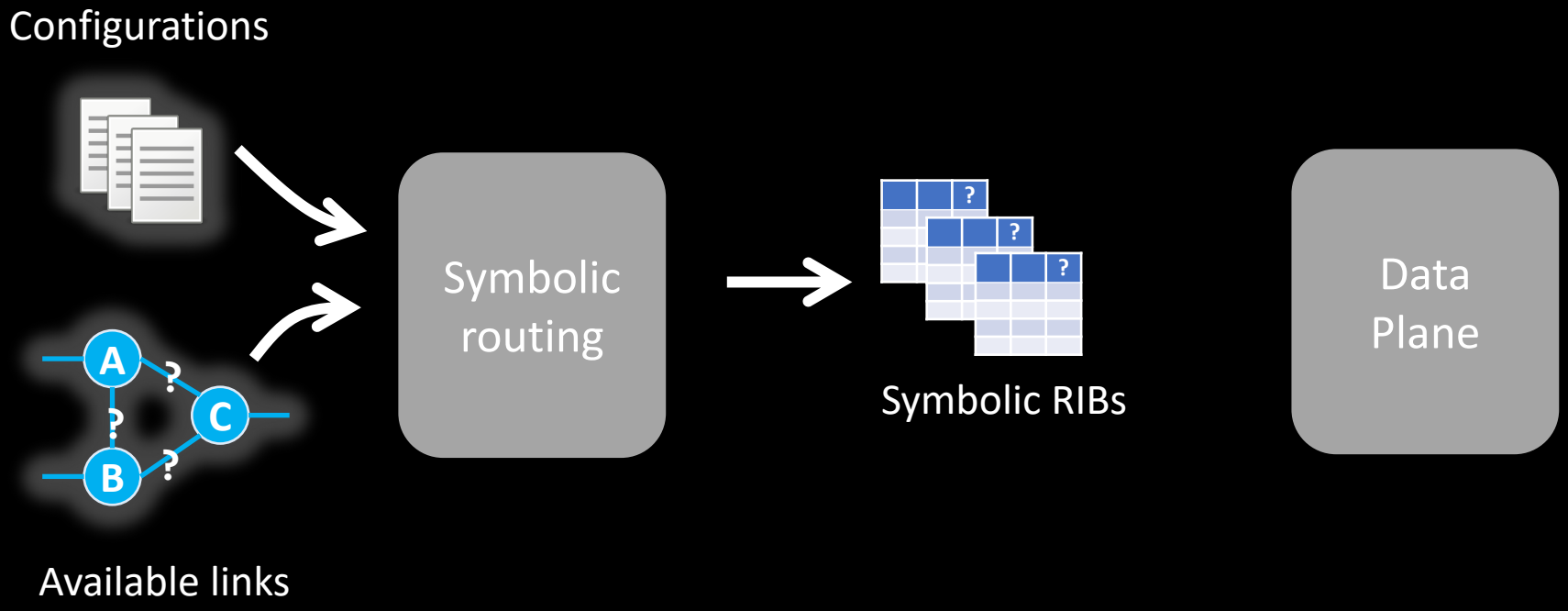


Available links

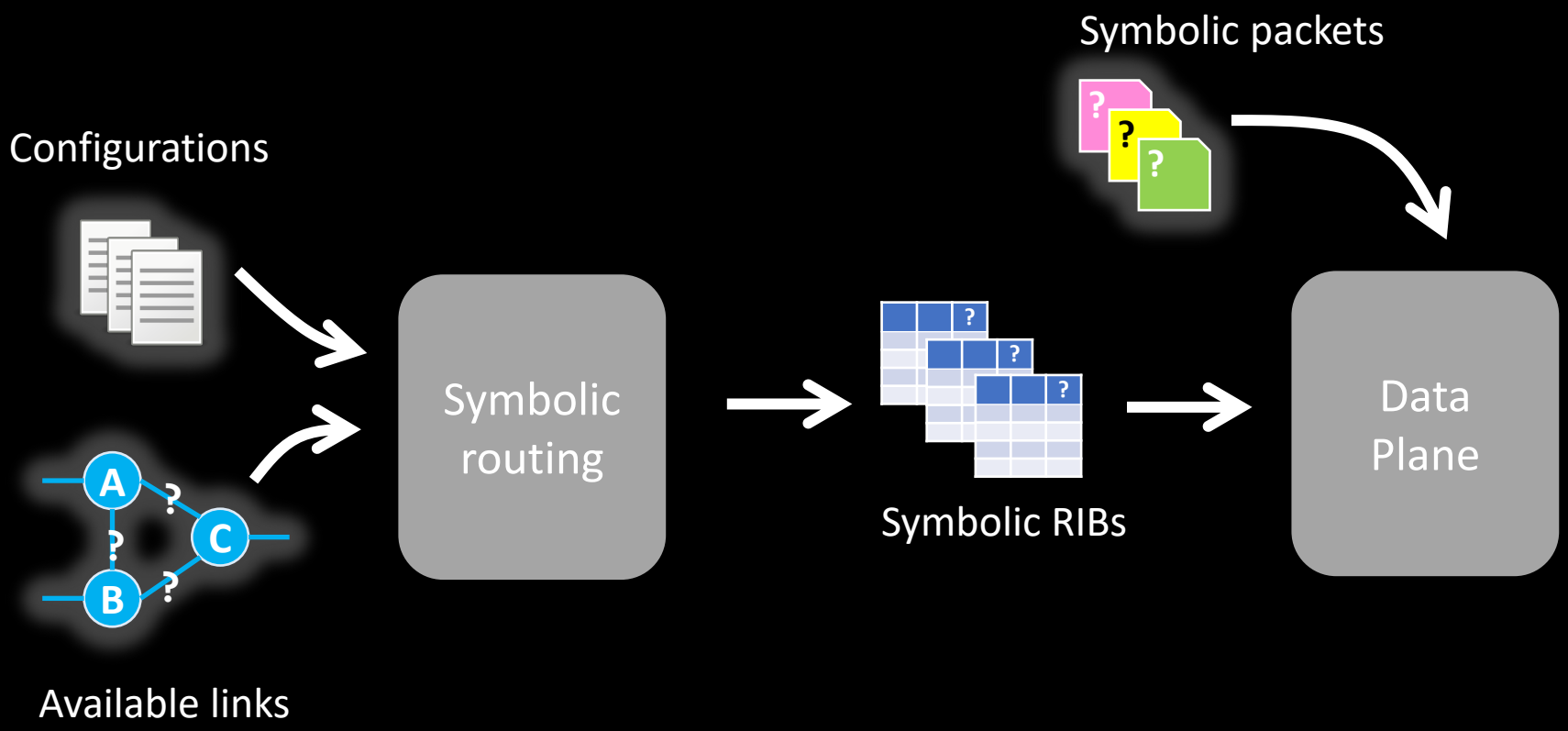
Symbolic routing

Data Plane

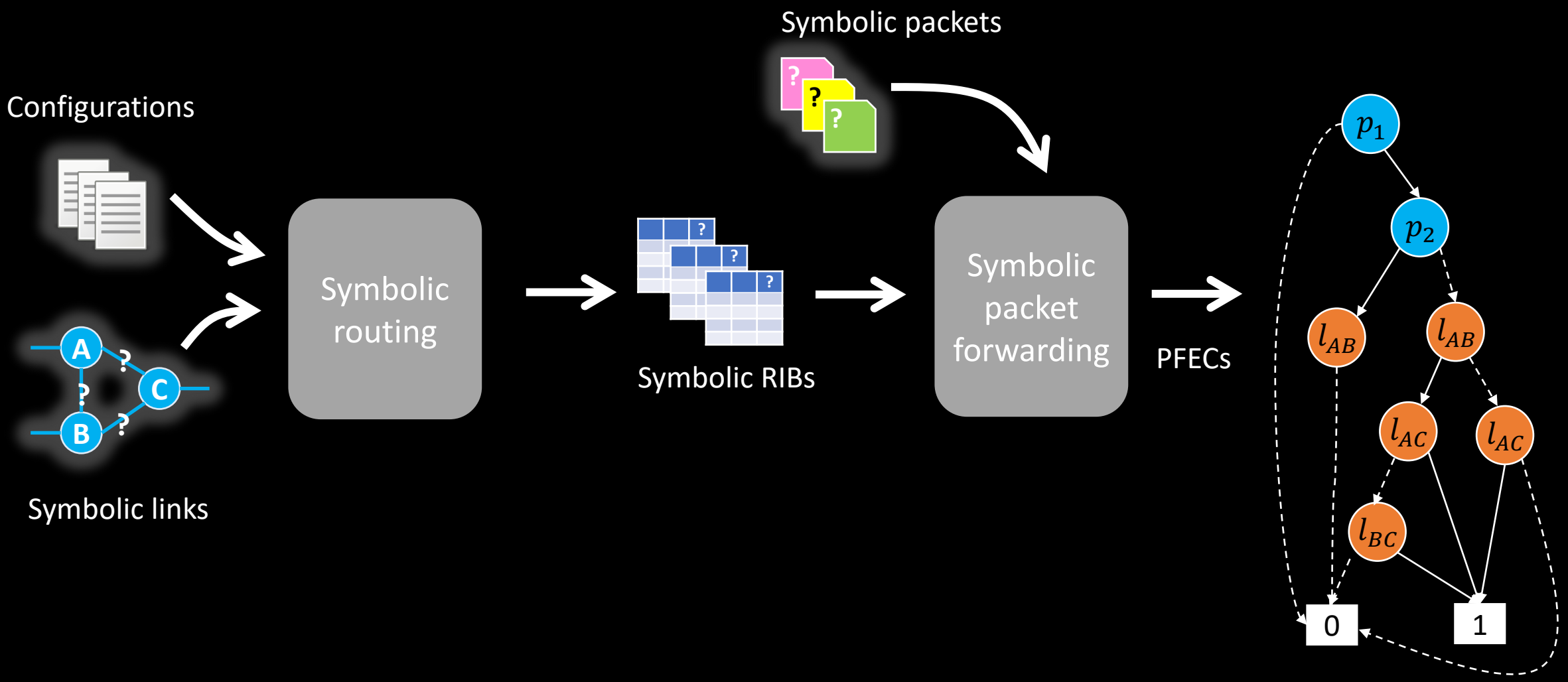
Symbolic execution of control and data planes



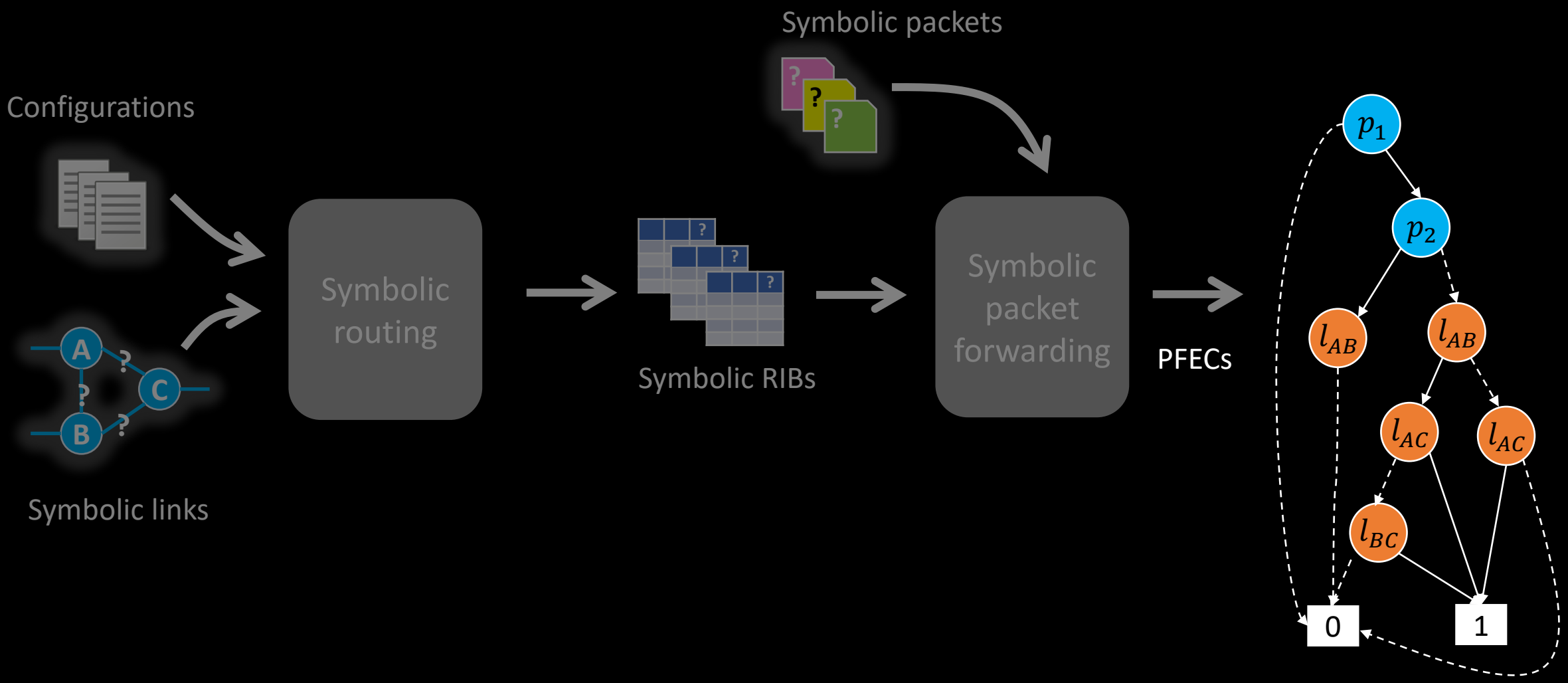
Symbolic execution of control and data planes



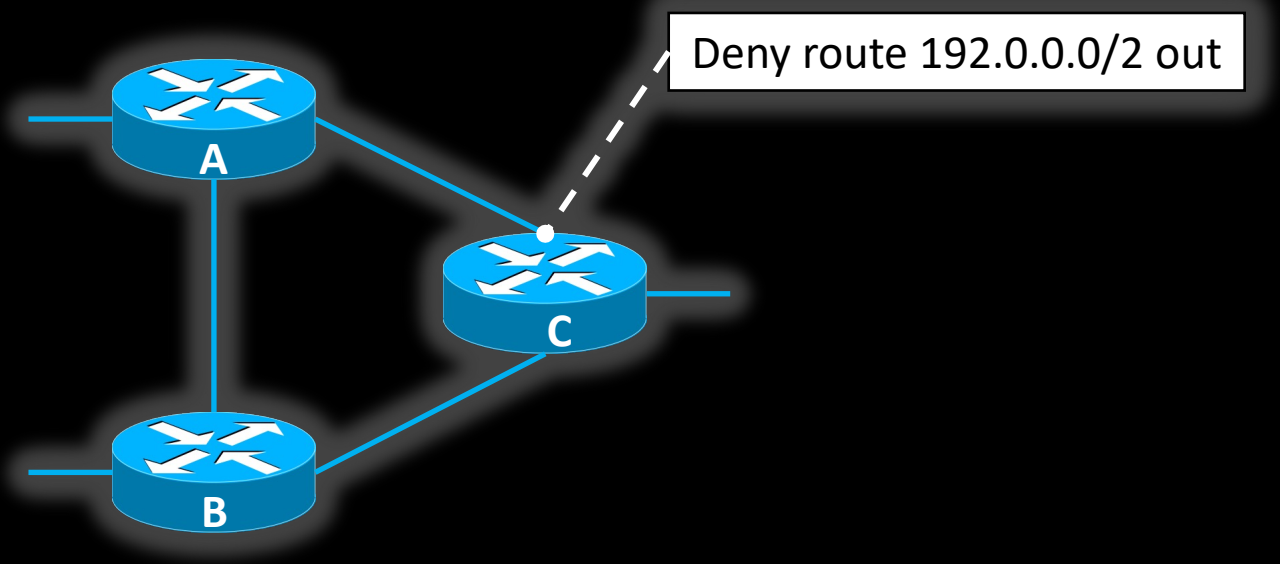
Symbolic execution of control and data planes

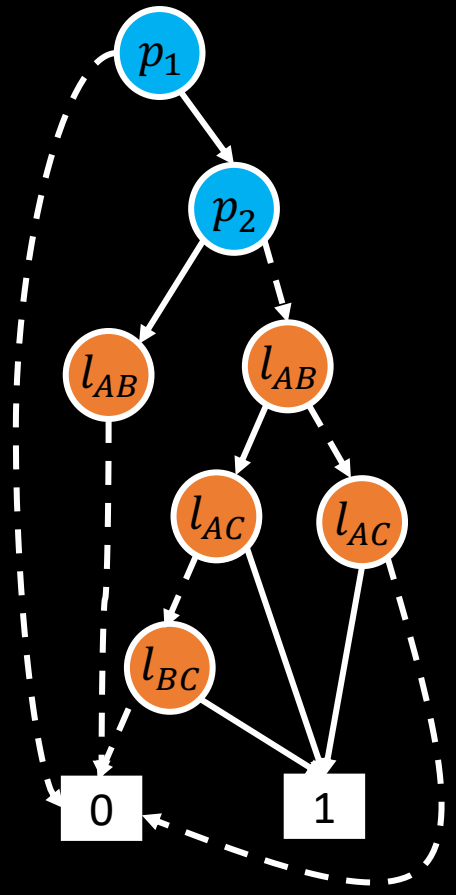


Symbolic execution of control and data planes

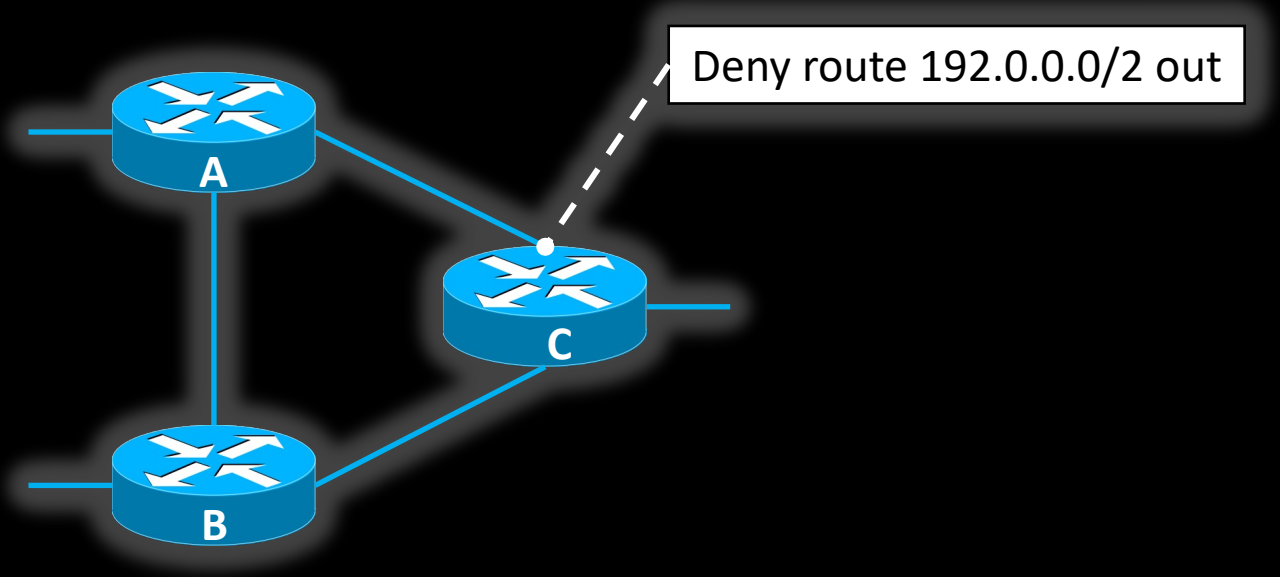


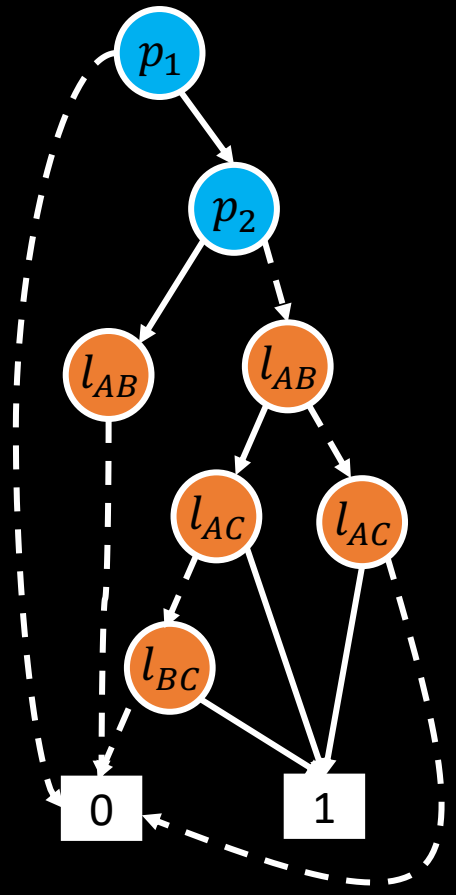
How many failures can occur and A still reach 128.0.0.0/2?



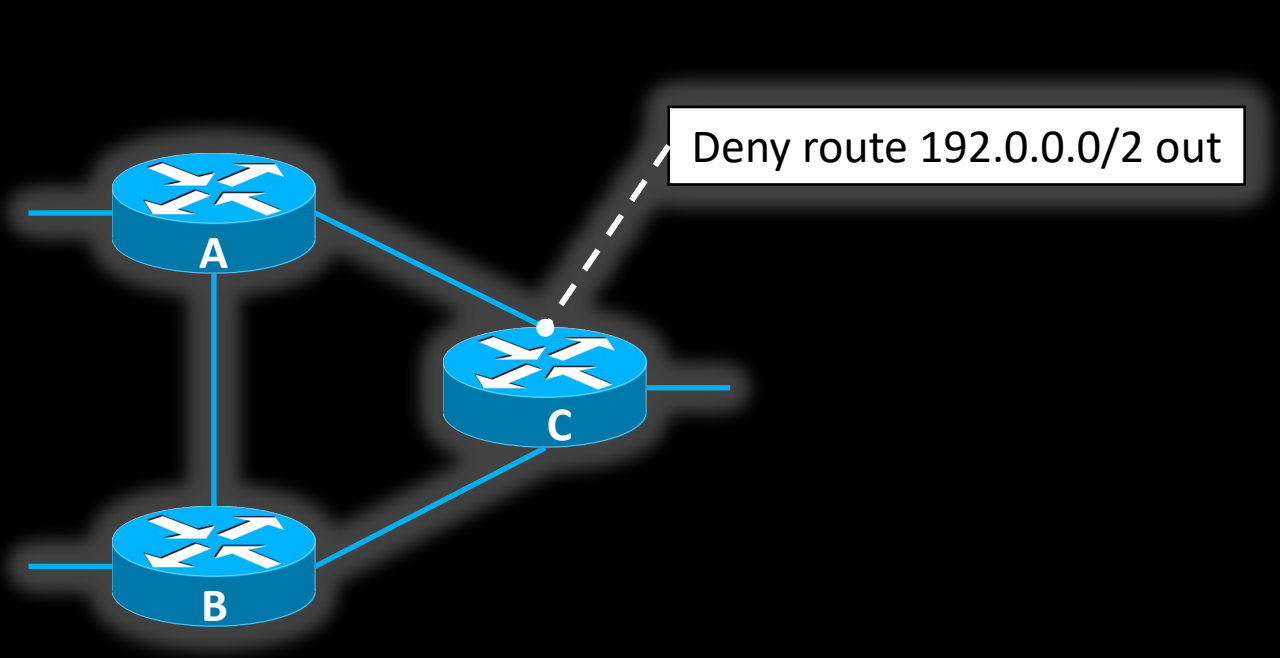


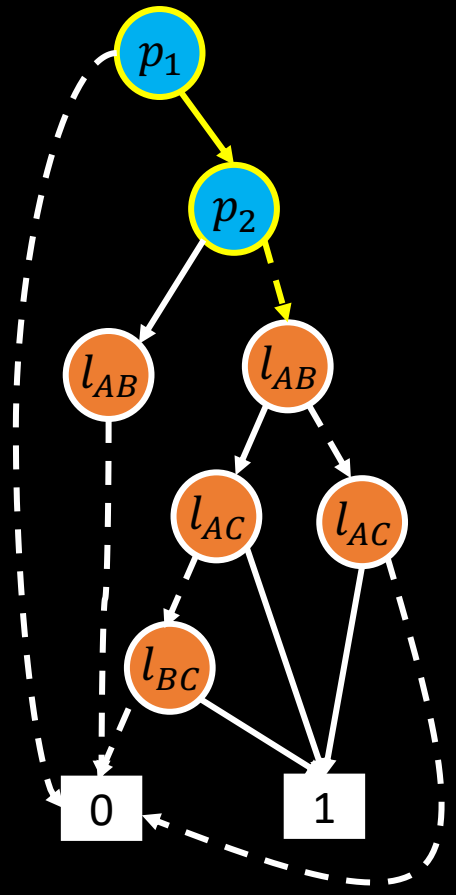
How many failures can occur and A still reach 128.0.0.0/2?



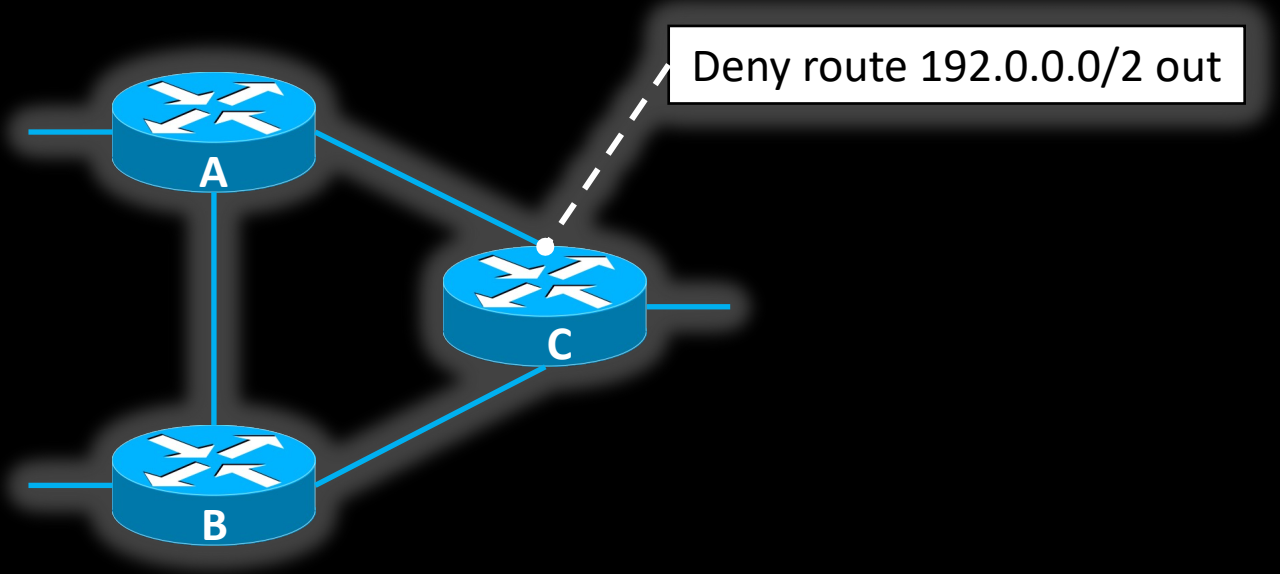


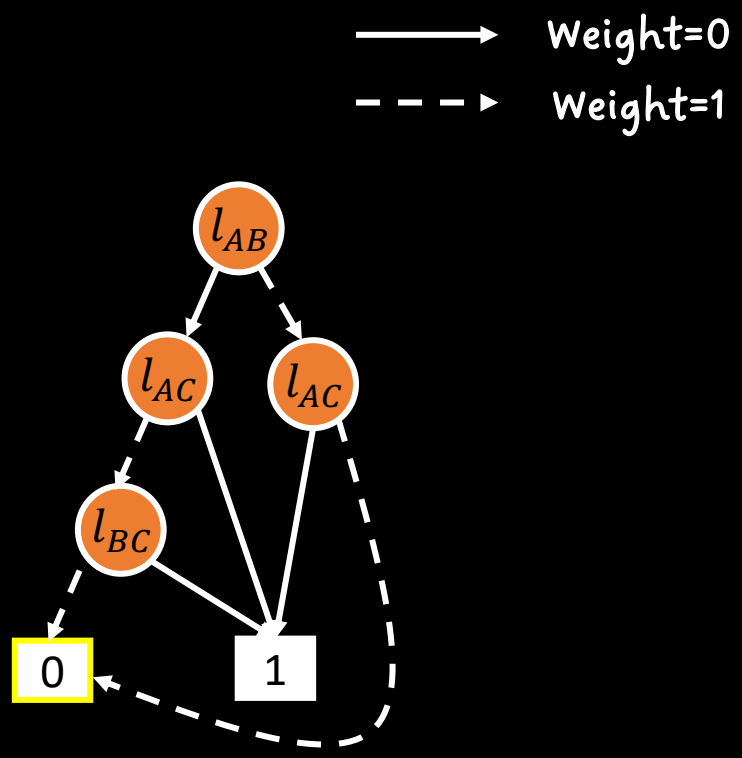
How many failures can occur and A still reach 128.0.0.0/2?





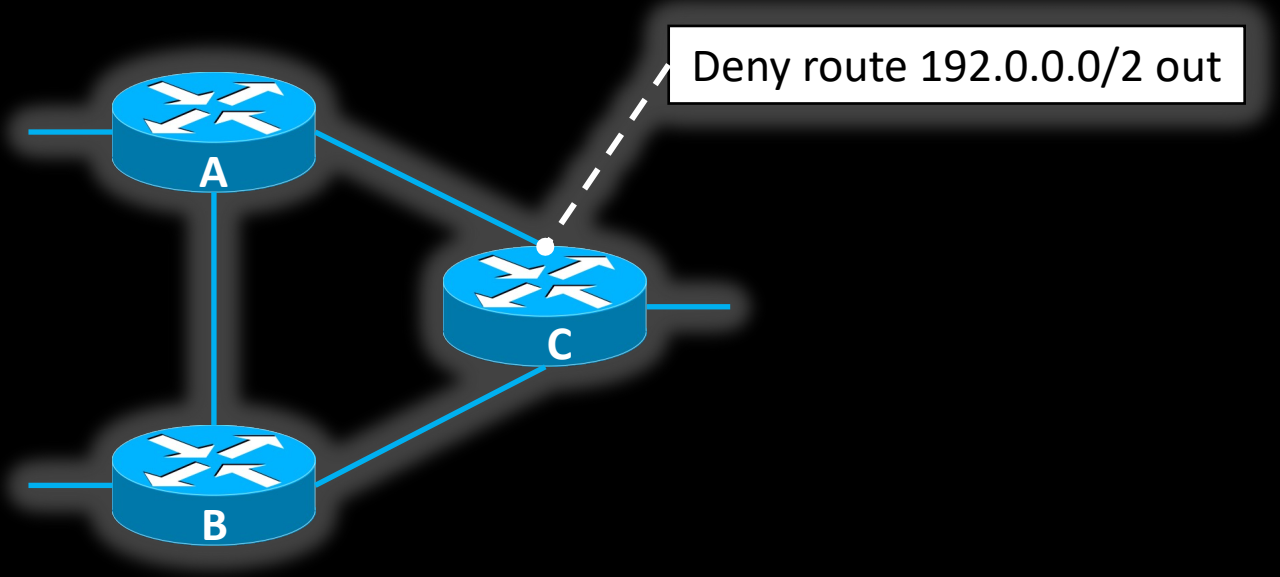
How many failures can occur and A still reach 128.0.0.0/2?

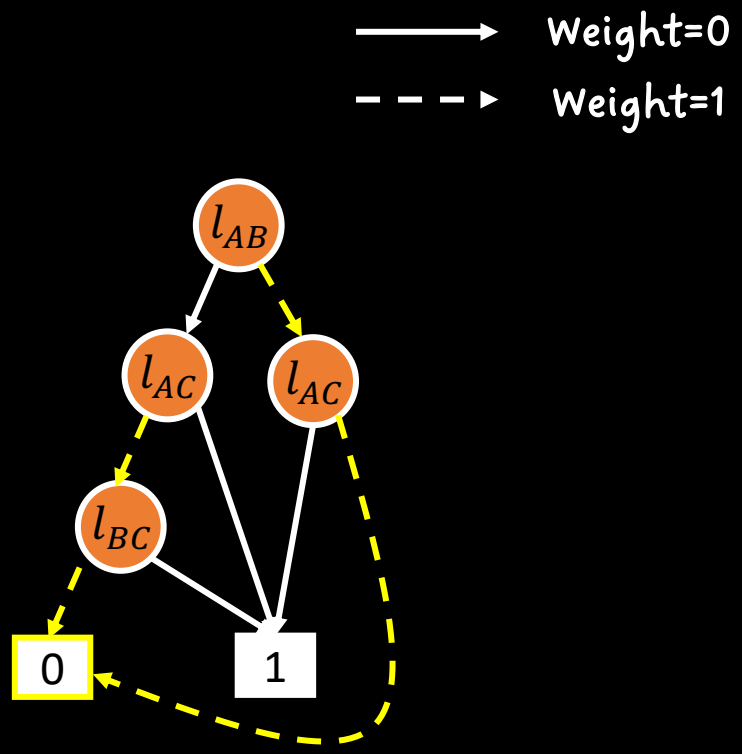




Failure tolerance =
(Len of shortest path to terminal 0) - 1

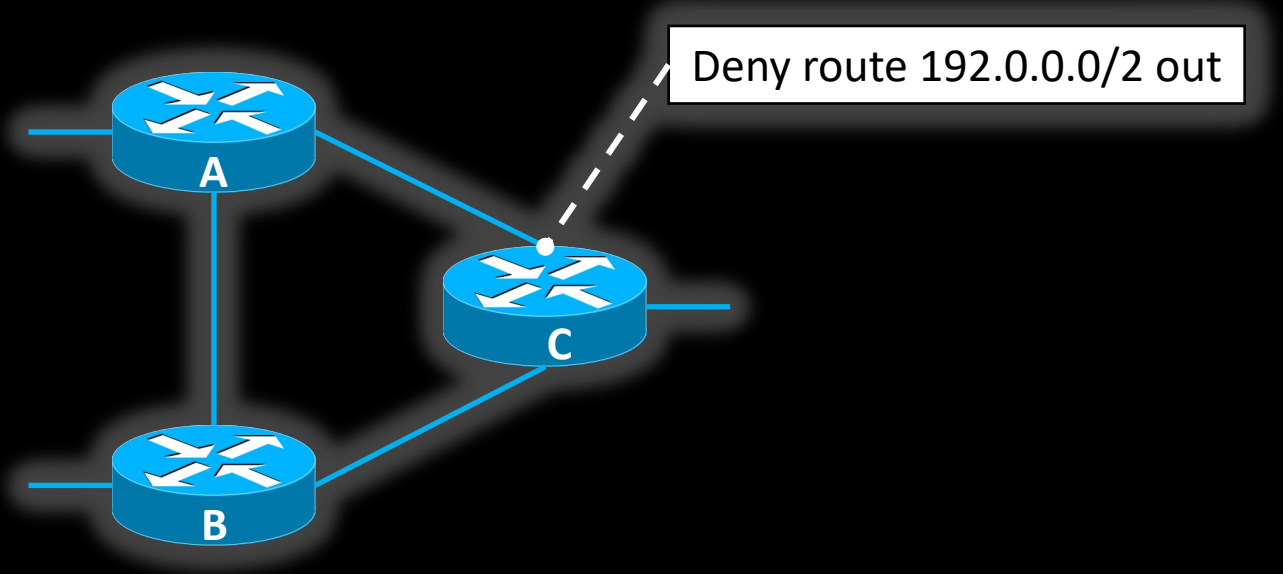
How many failures can occur and A still reach 128.0.0.0/2?

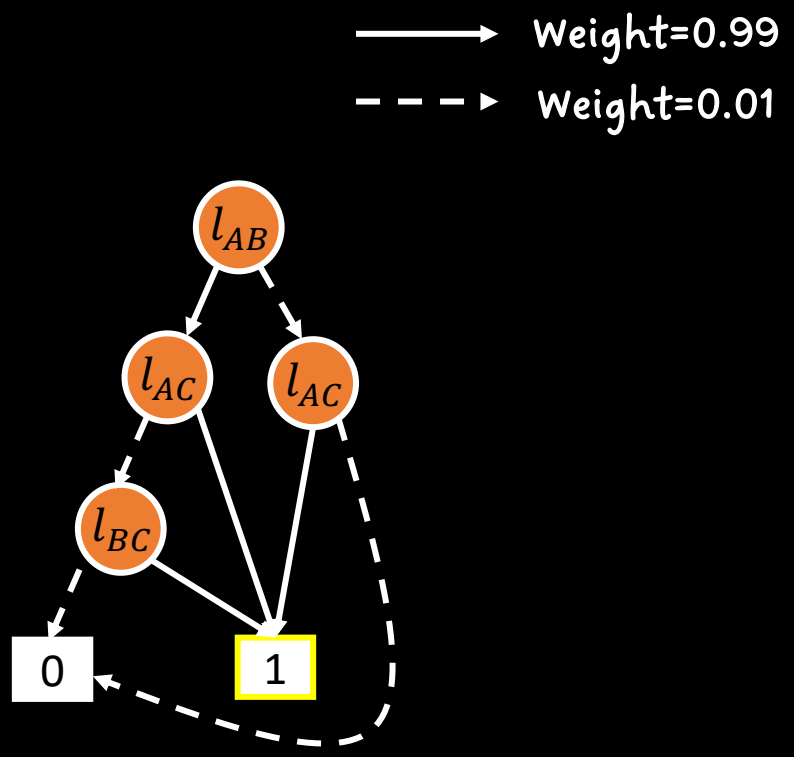




Failure tolerance =
2 - 1

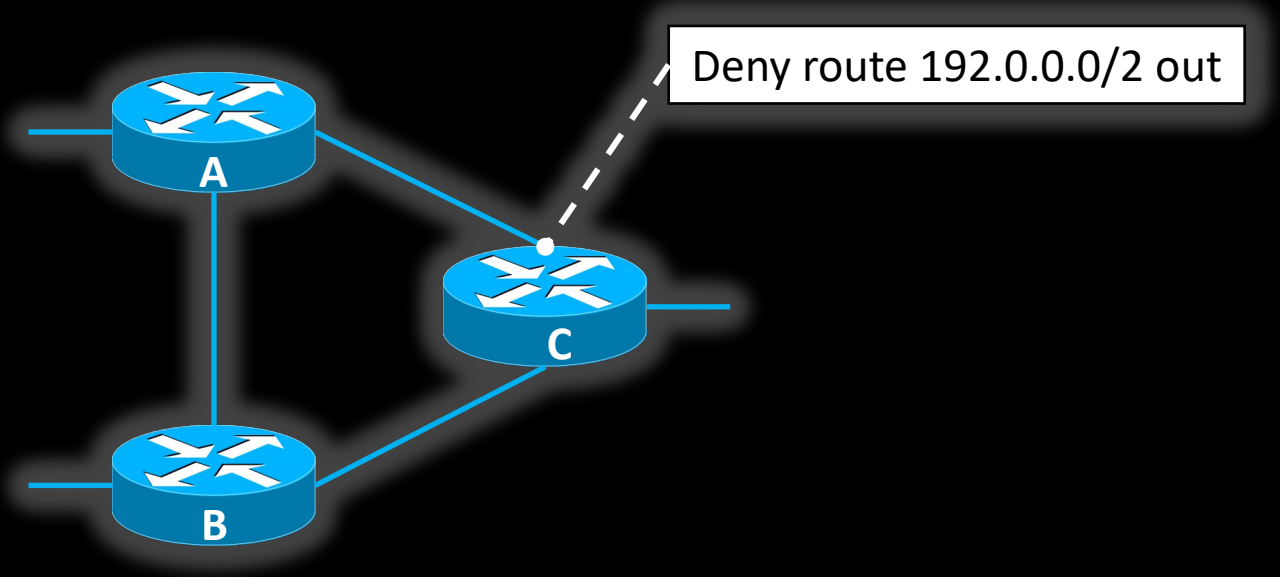
What is the probability A can reach 128.0.0.0/2?

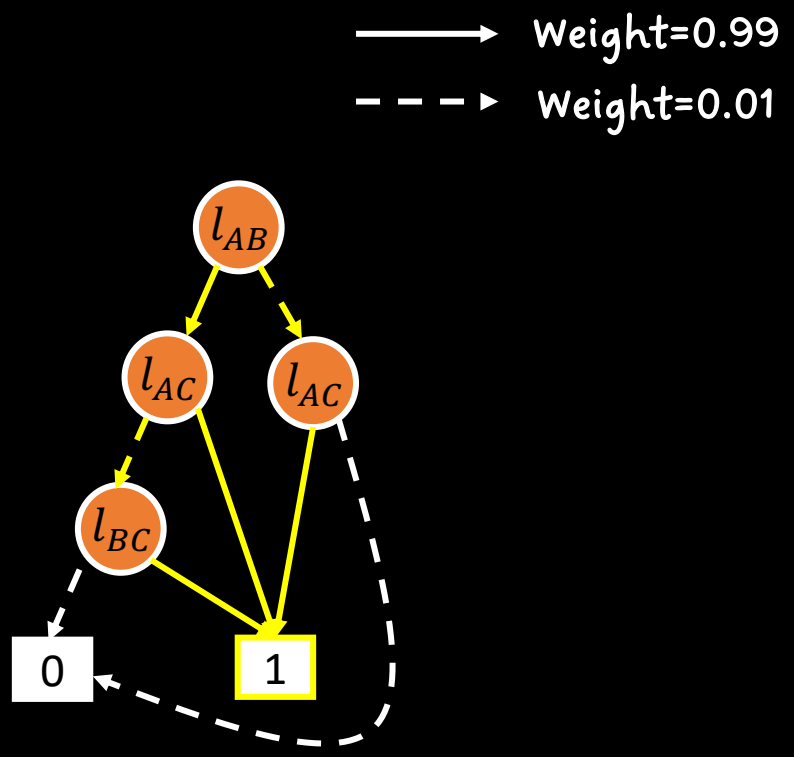




Probability =
Weighted sum of all paths to terminal 1

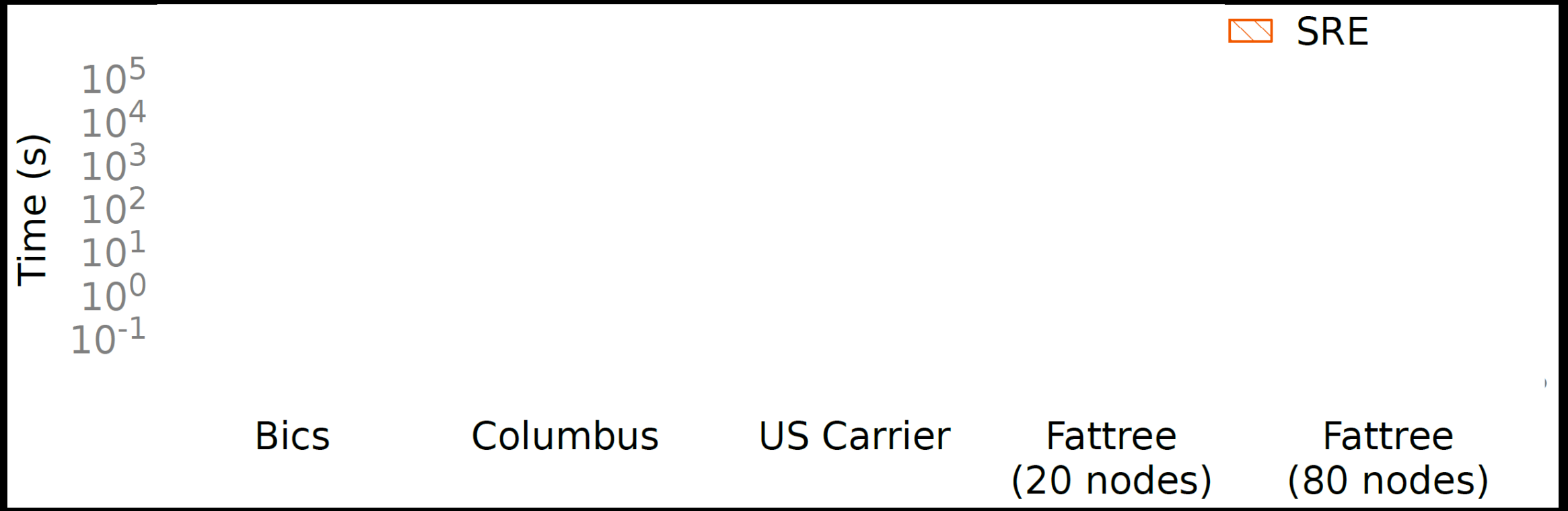
What is the probability A can reach 128.0.0.0/2?



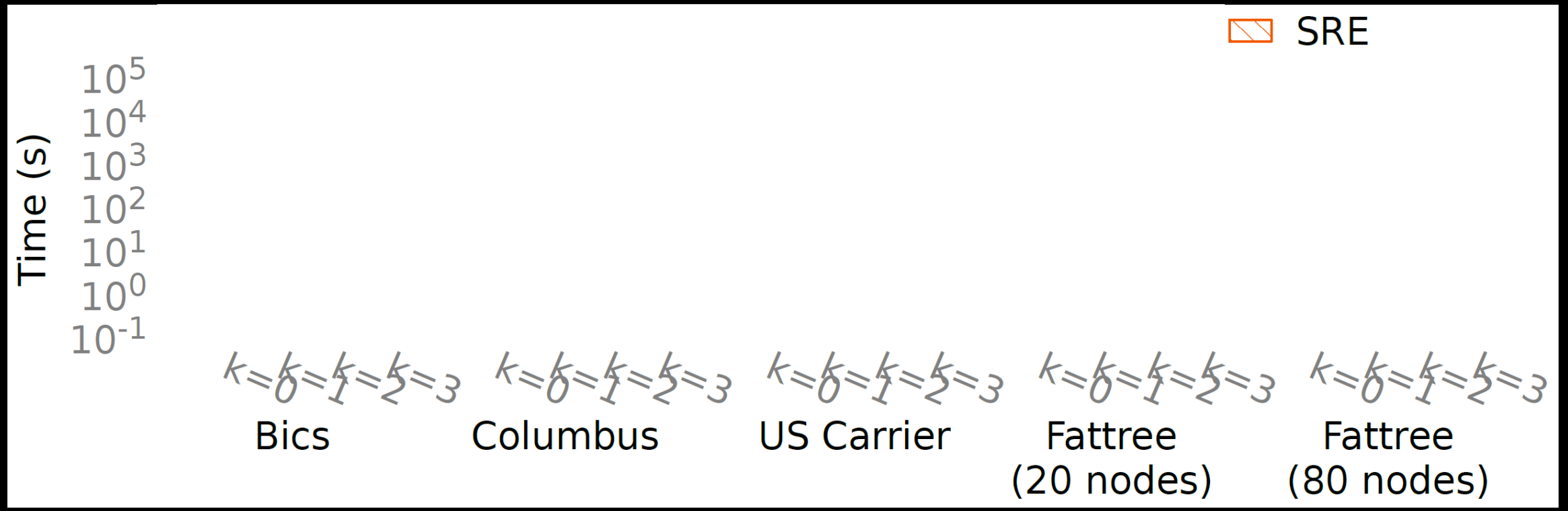


Probability =
0.99 * (0.01 * 0.99 + 0.99) + 0.01 * 0.99

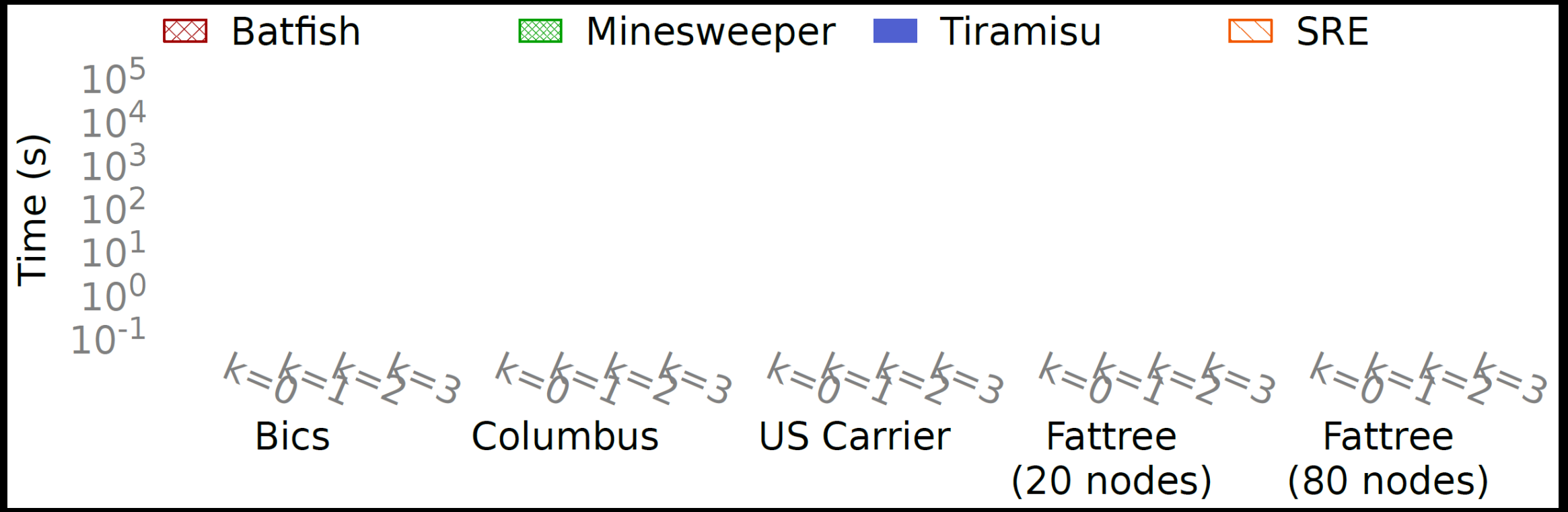
Experiment: compute all-pairs reachability



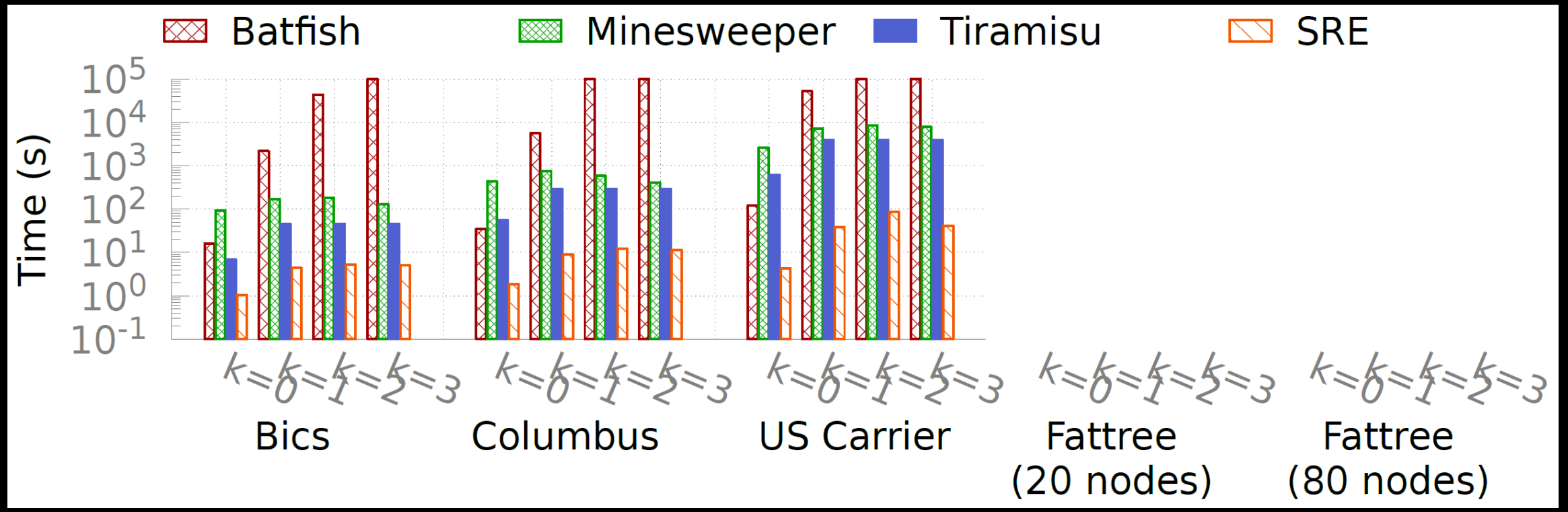
Experiment: compute all-pairs reachability



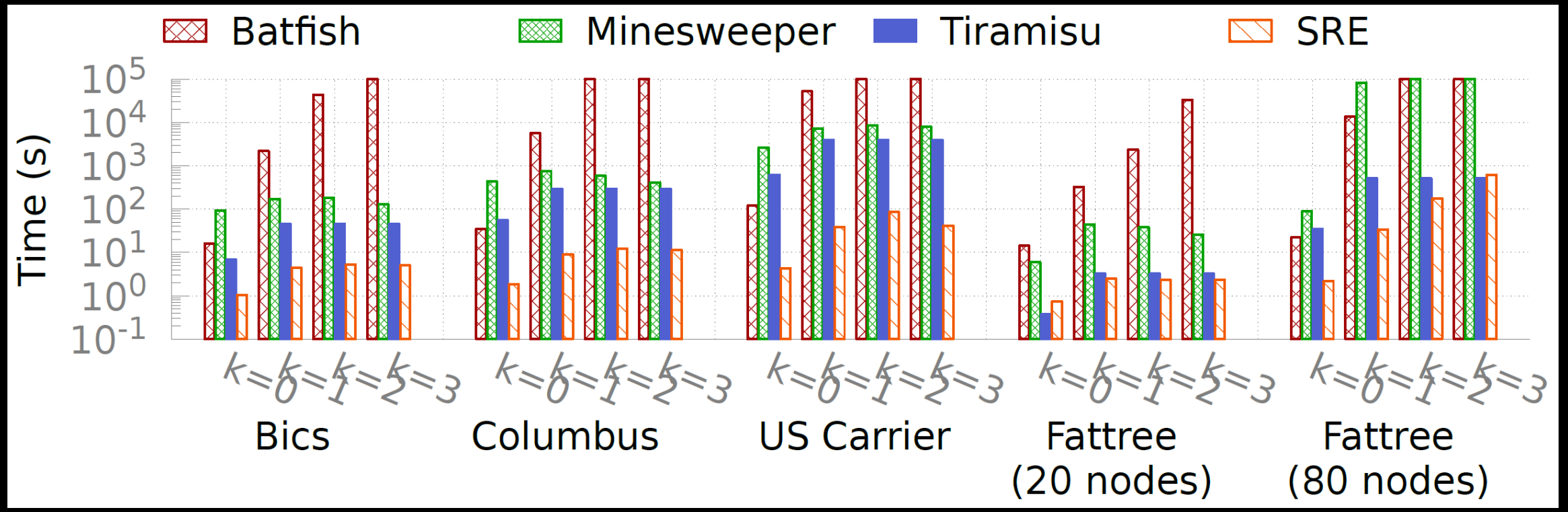
Experiment: compute all-pairs reachability



Experiment: compute all-pairs reachability



Experiment: compute all-pairs reachability



Scales to the product space of headers and failures by computing PFECs

Generalizes to deterministic and probabilistic failures by delaying binding to a failure model