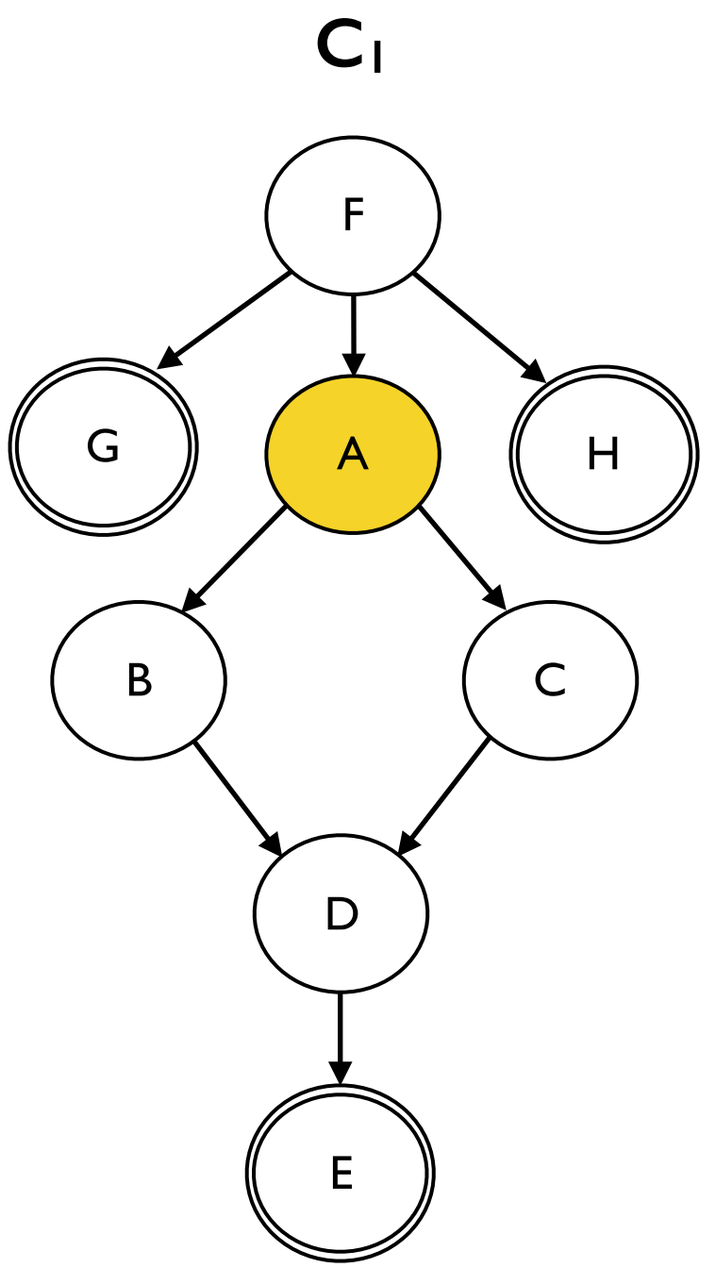
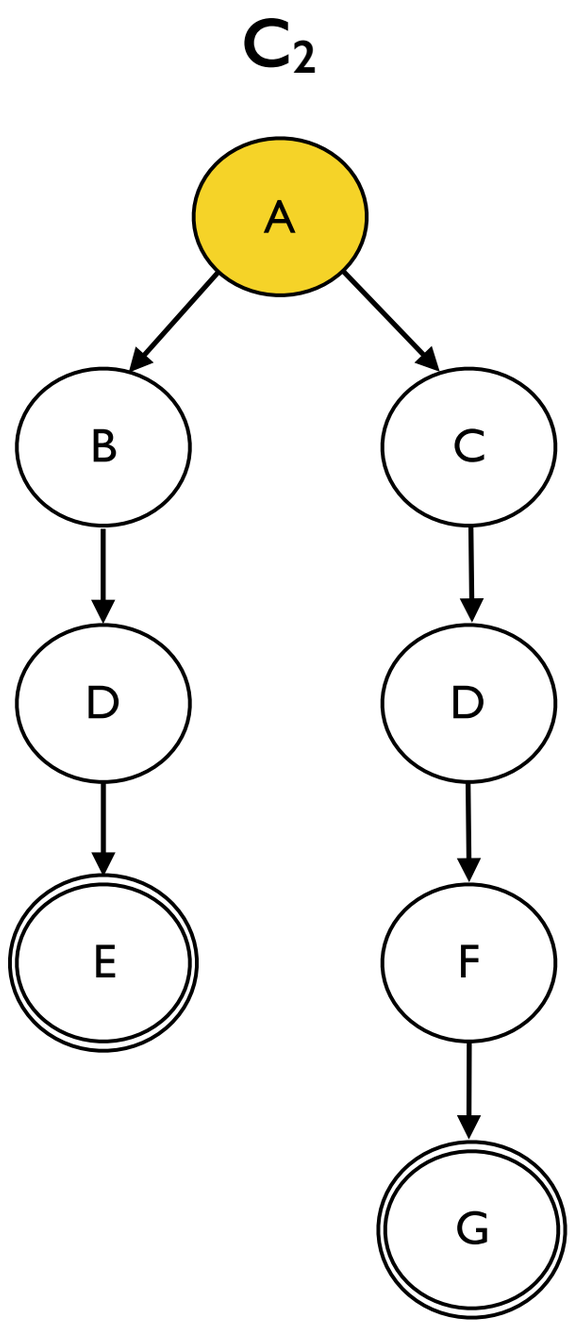
Quick Summary

- Compilation Correctness (High-level)
- Limitations of Conservative Analysis
- Other Abstract Topologies (DCell, BCube, …)
- Verification of configs

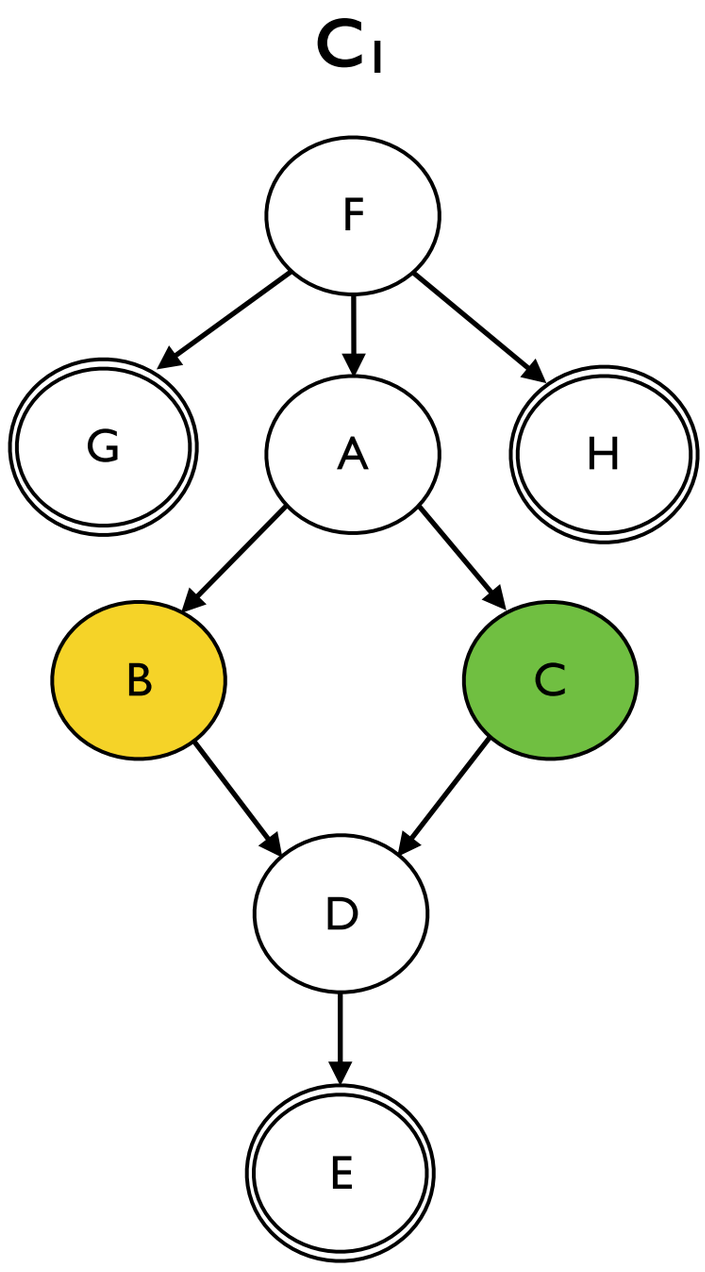
Compilation Correctness

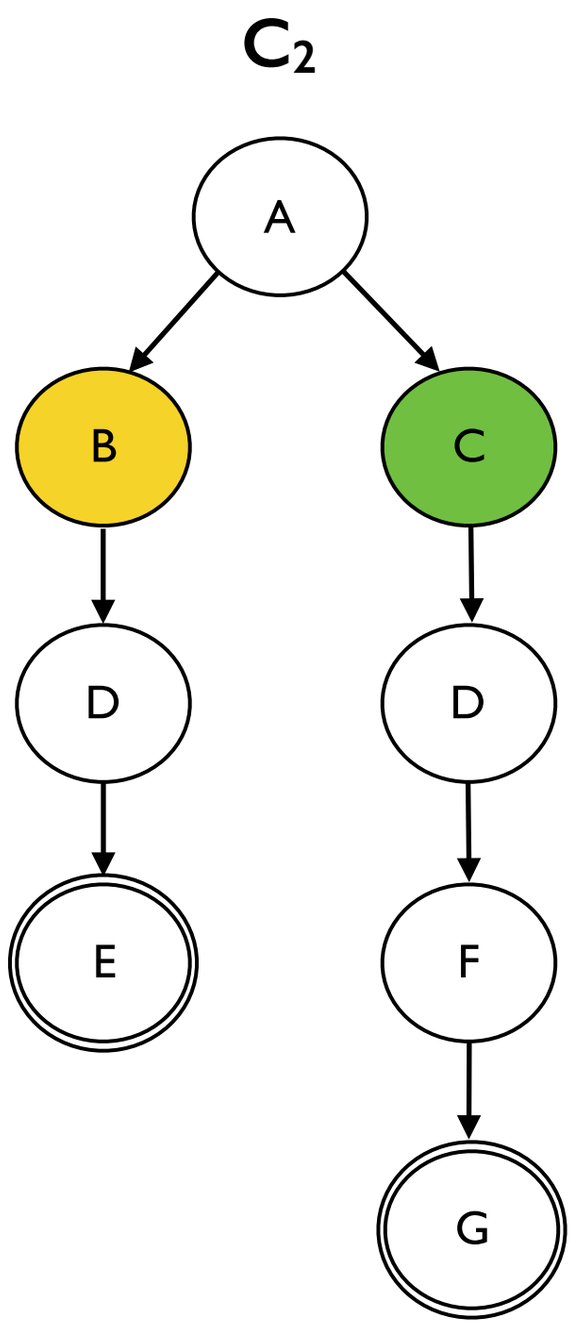
Failure Safety (Recap)



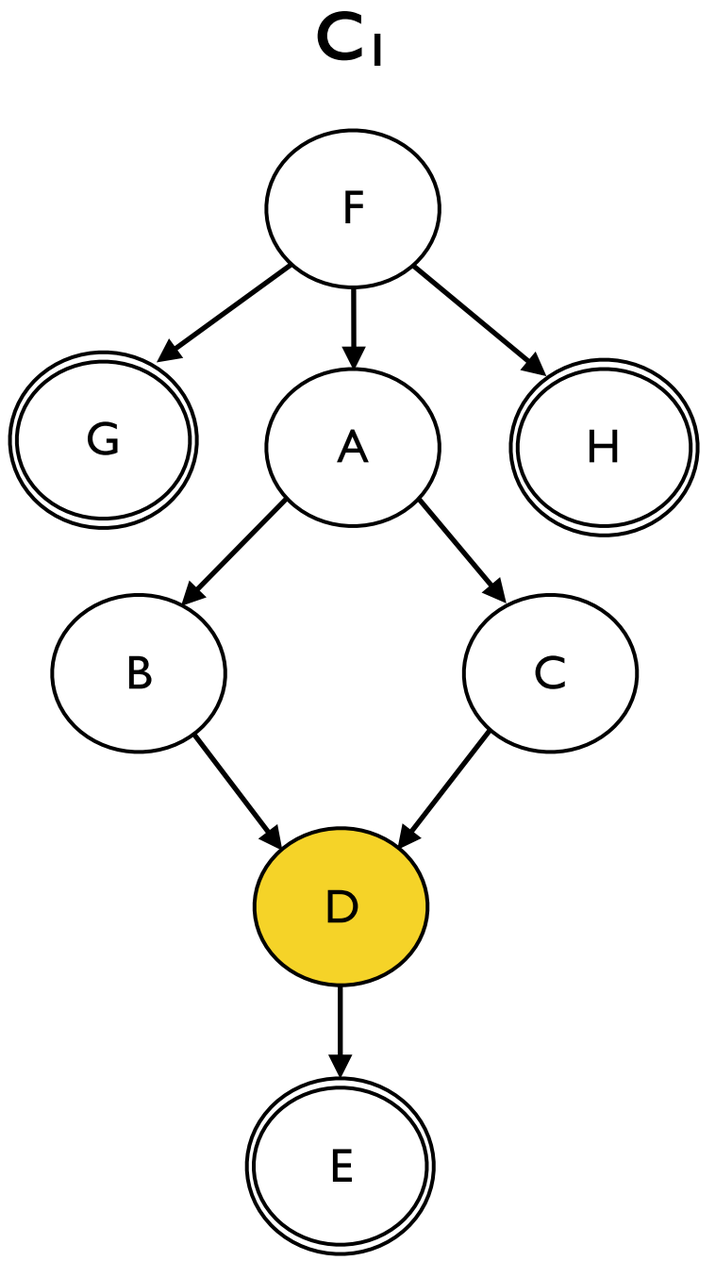


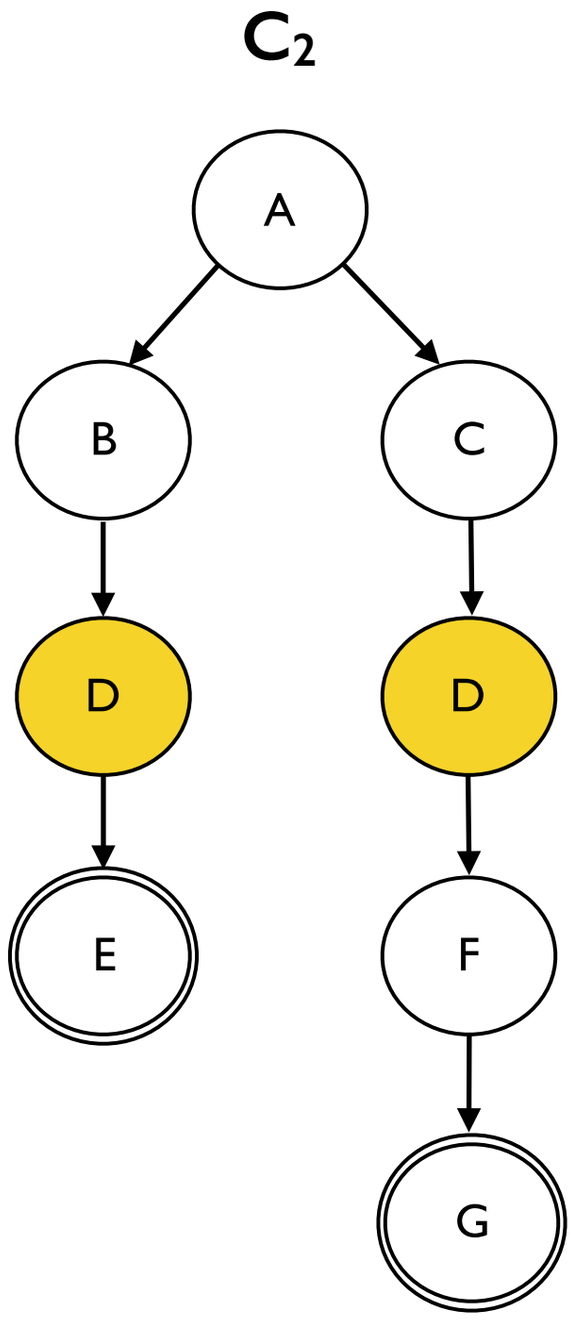
Failure Safety (Recap)



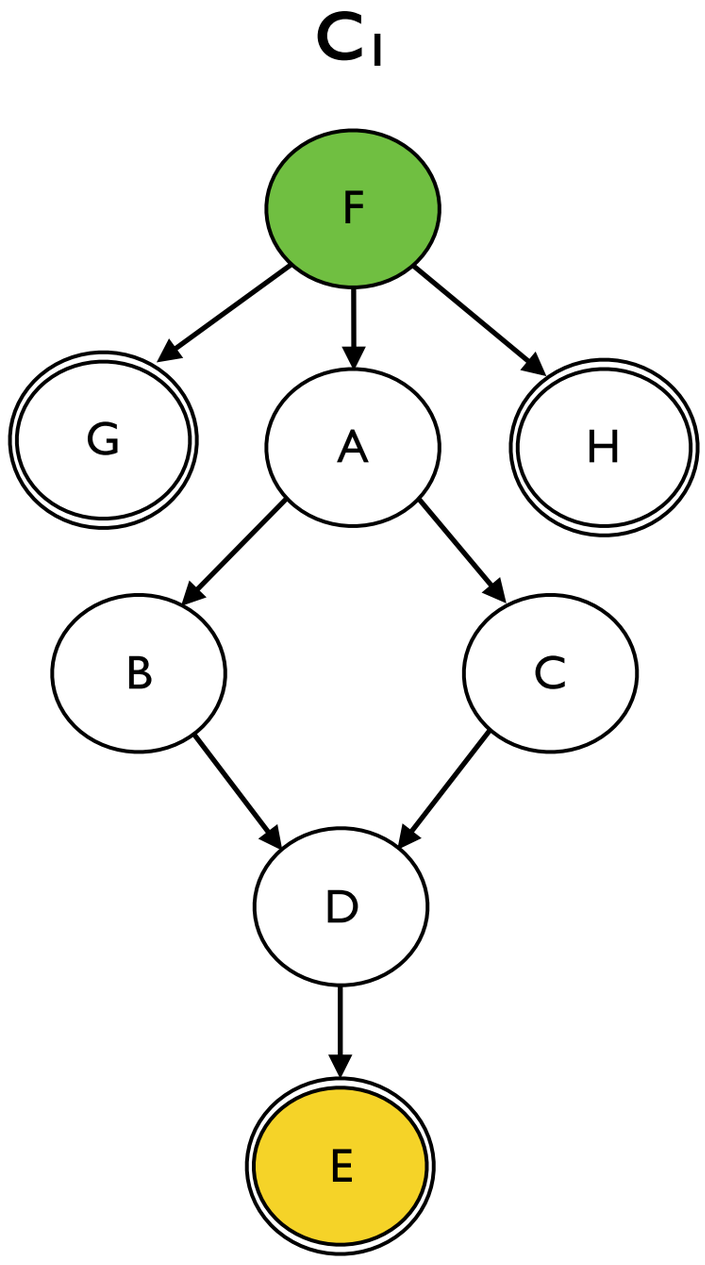


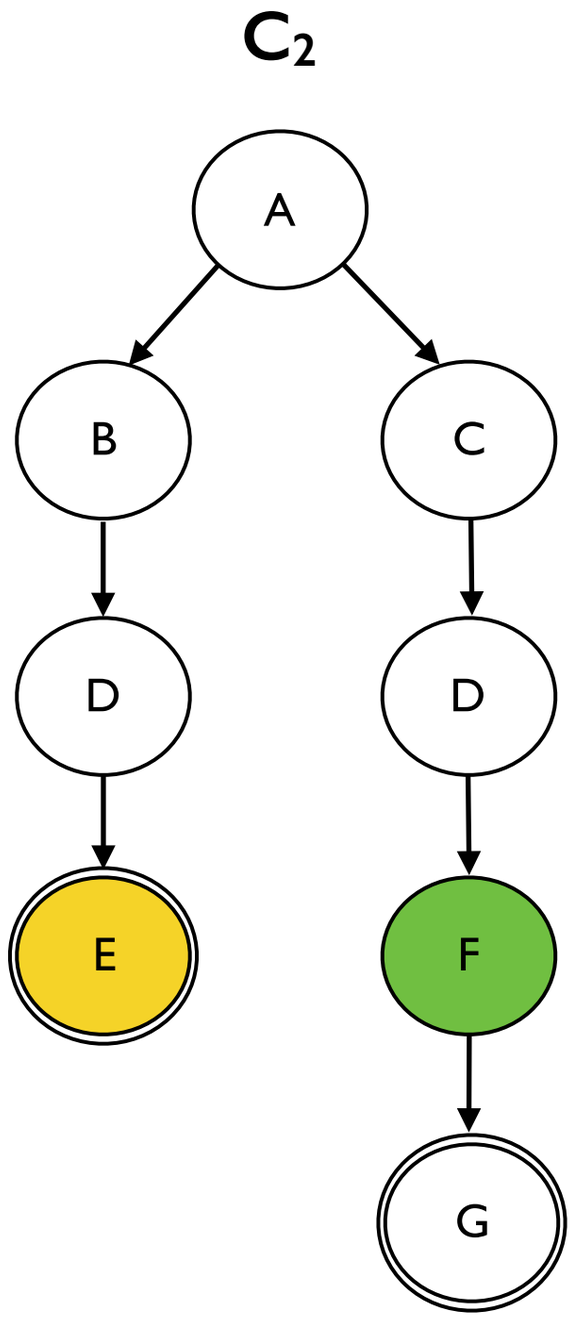
Failure Safety (Recap)



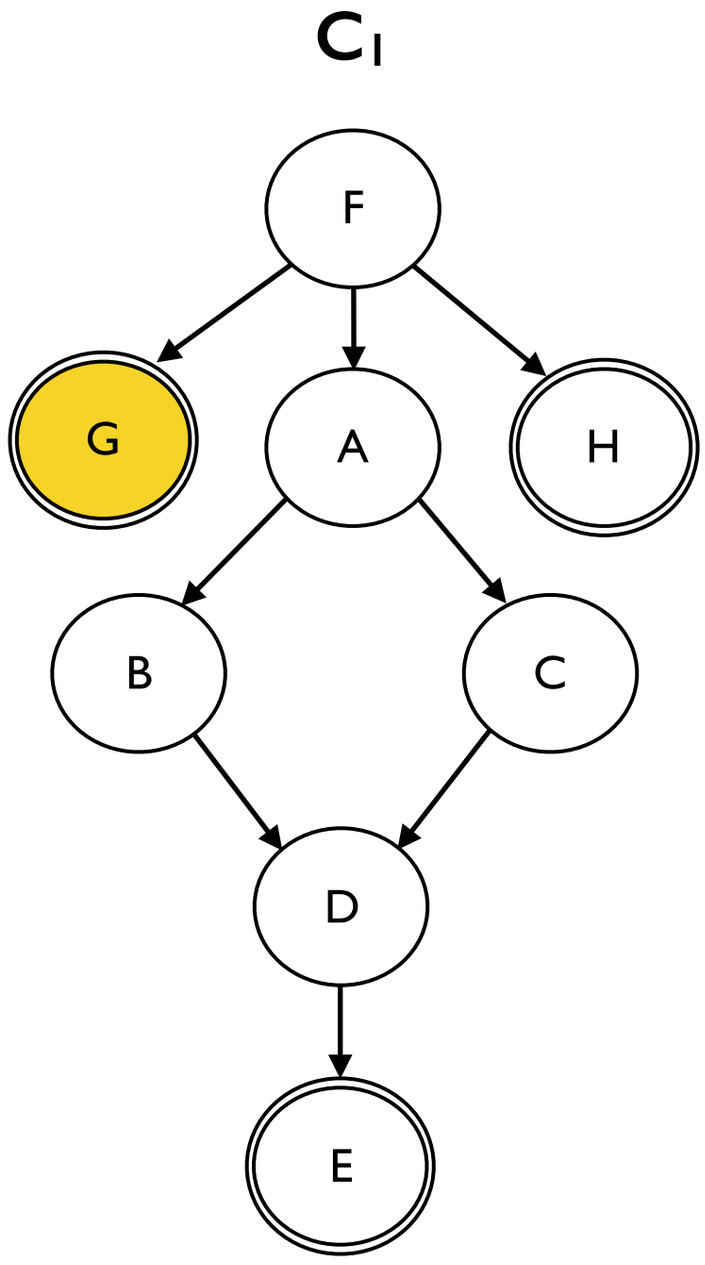


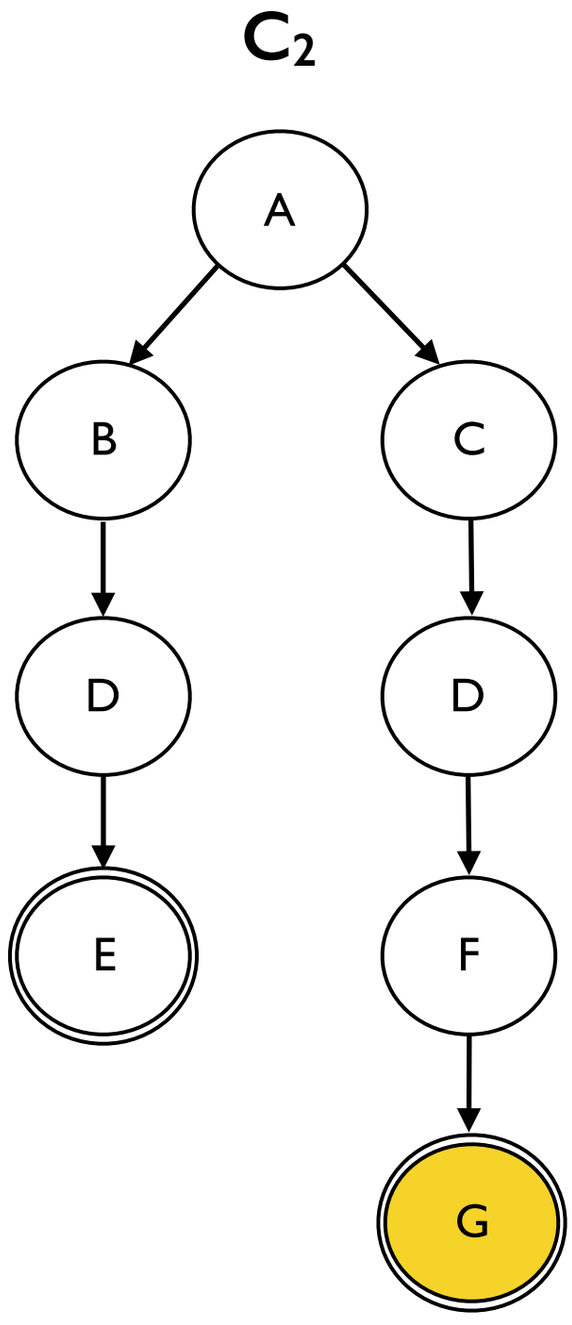
Failure Safety (Recap)



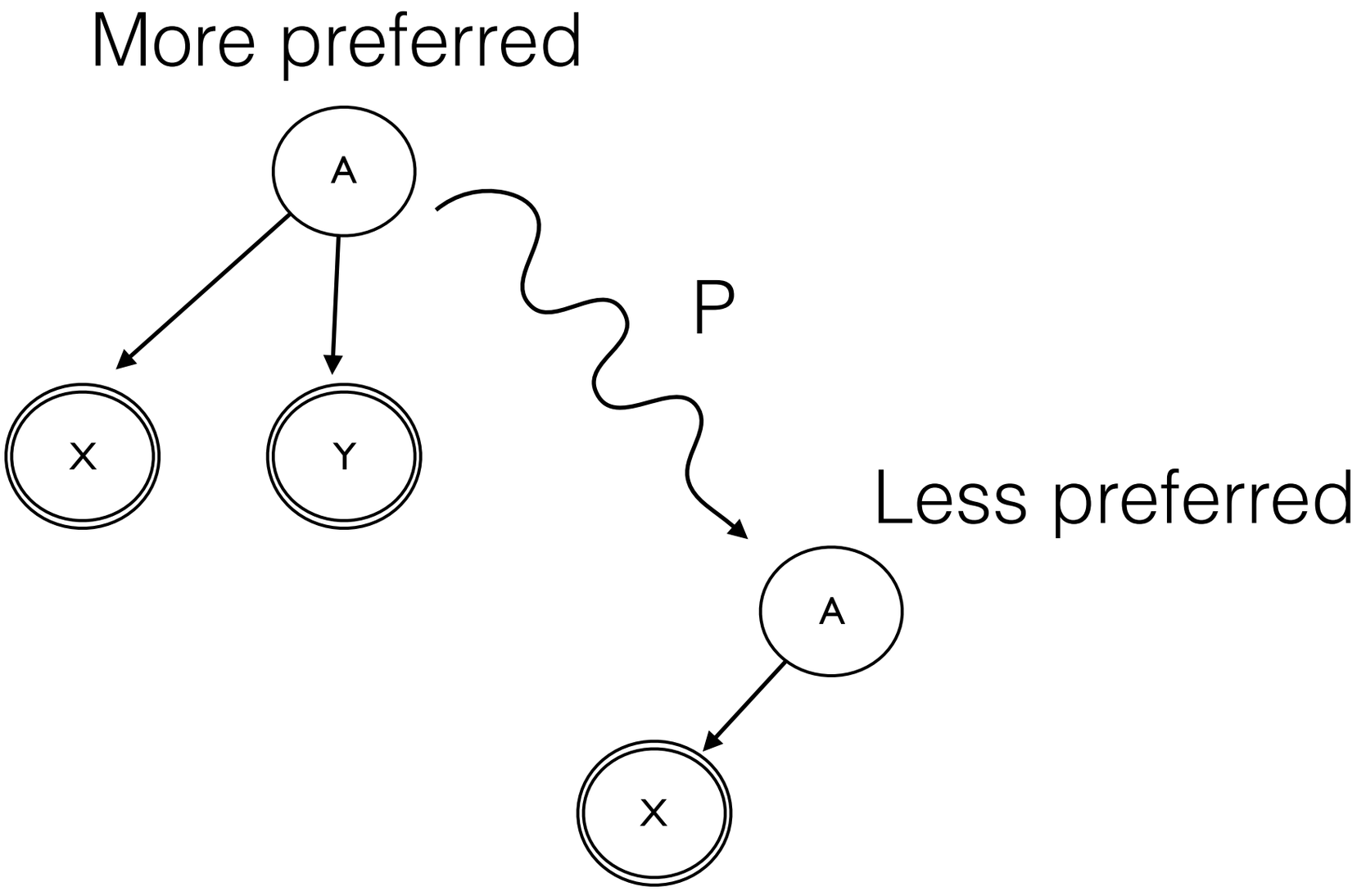


Failure Safety (Recap)





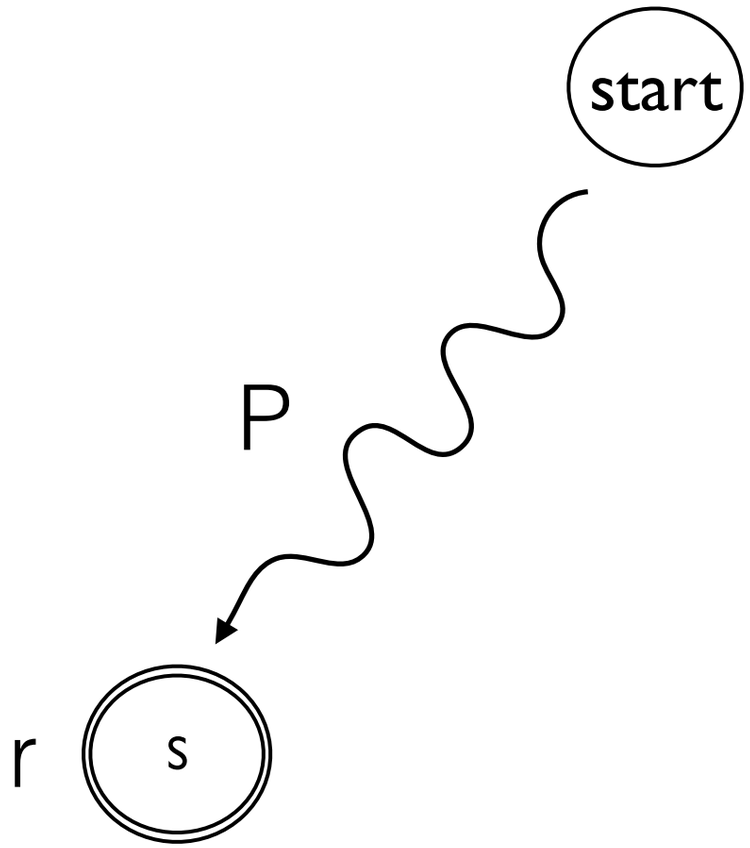
Failure Safety (Recap)



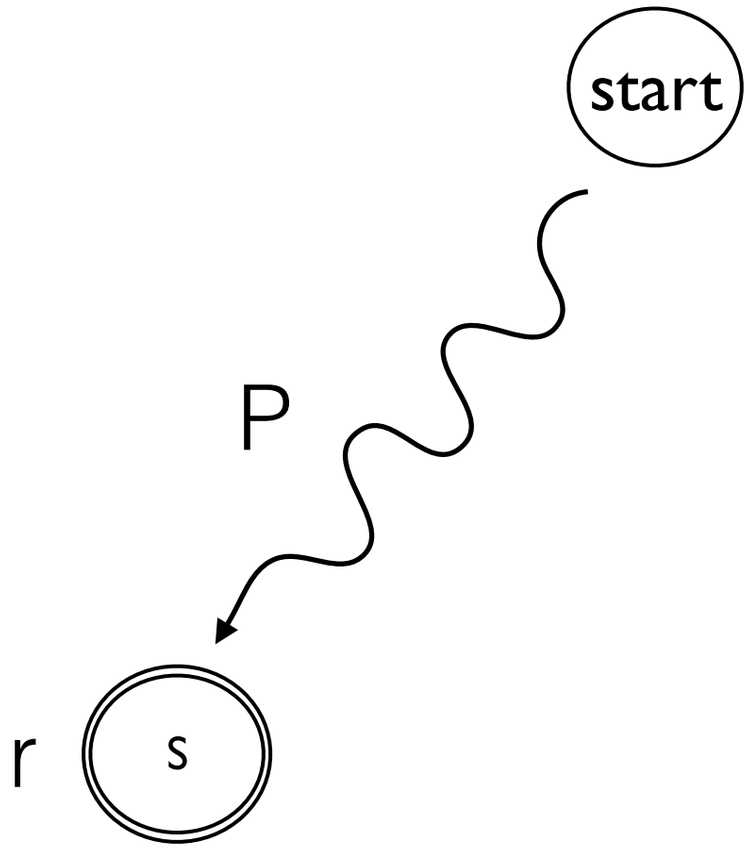
Proof of Correctness (High level)

Statement:

Traffic always flows along ***some best simple*** path to source *s* when such a path exists in the network

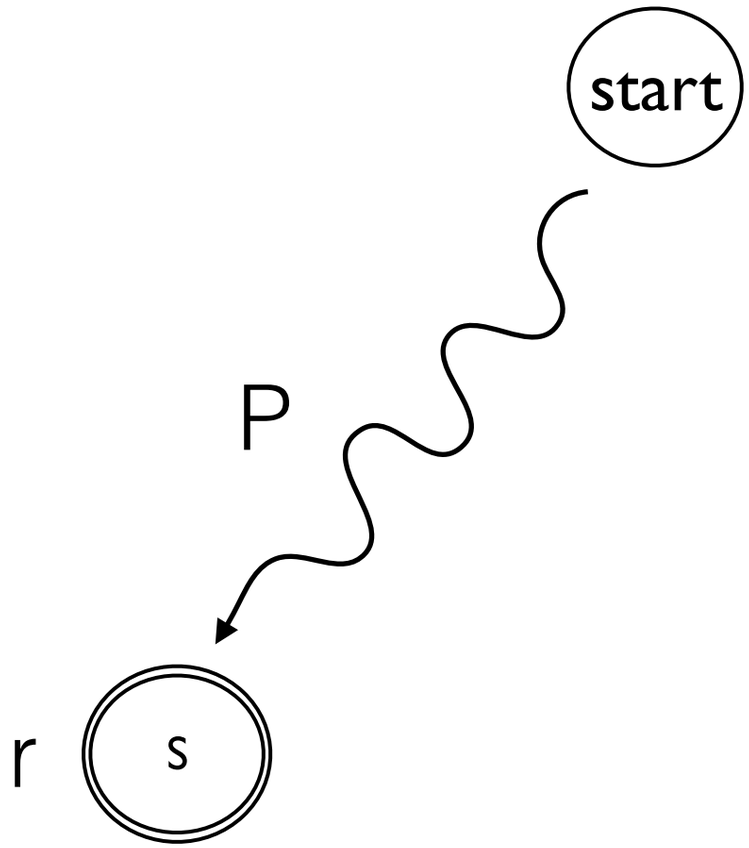


Assume path P is one of the highest rank simple paths in the network given the failures

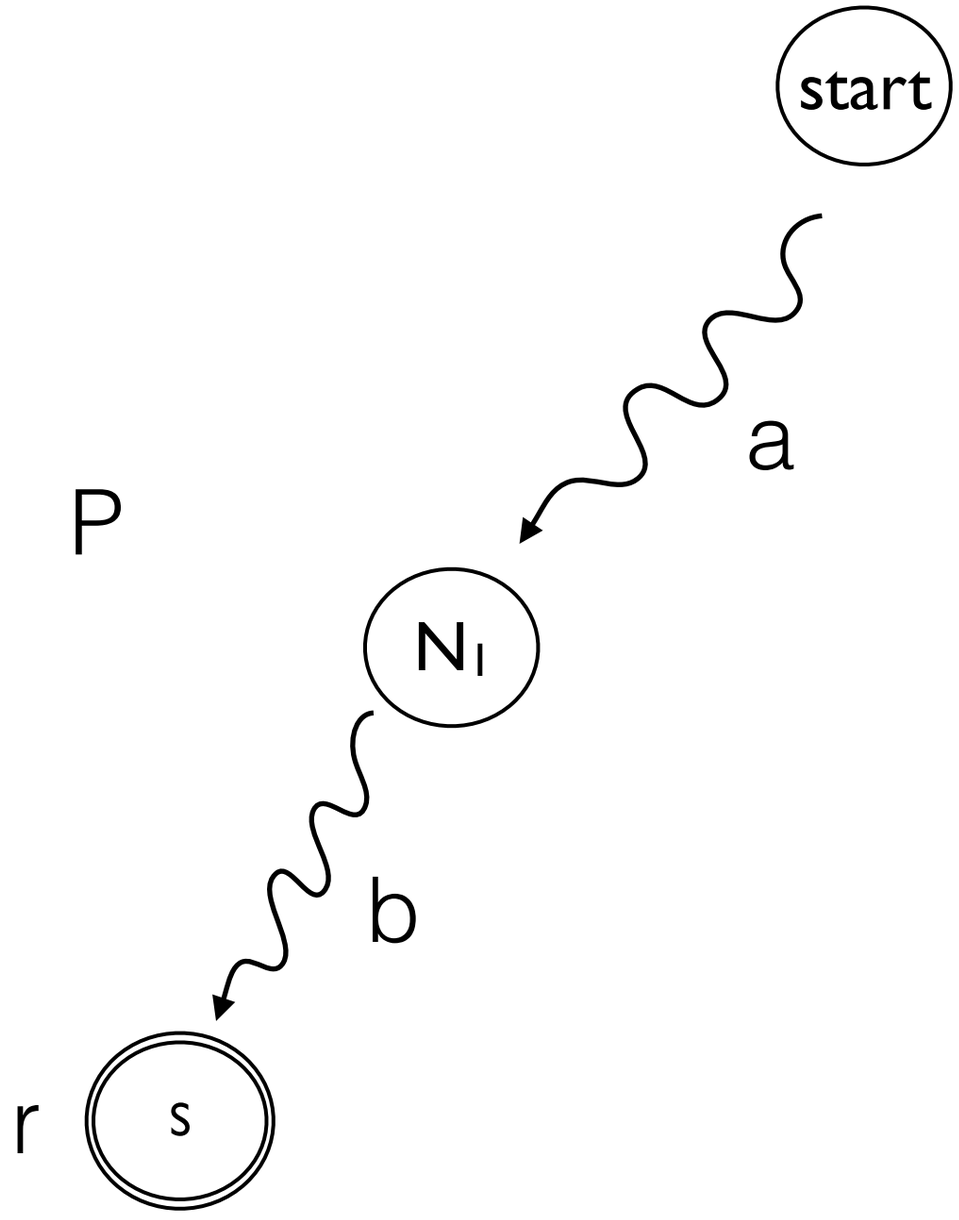


Assume path P is one of the highest rank simple paths in the network given the failures

Then path P exists in the PG with (best) rank of r for source s



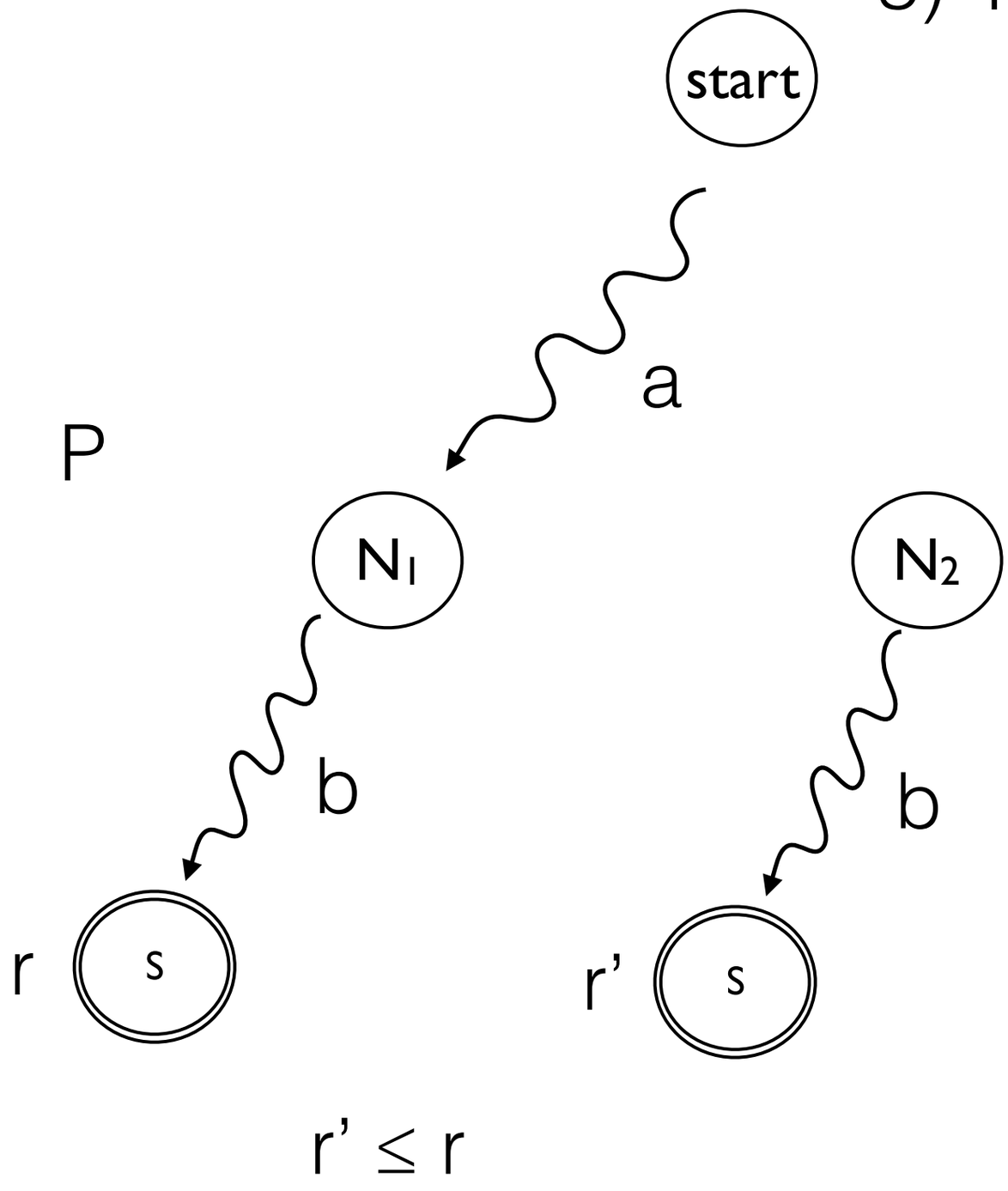
The only way traffic does not flow along path P is if some node on P prefers a different advertisement



The only way traffic does not flow along path P is if some node on P prefers a different advertisement

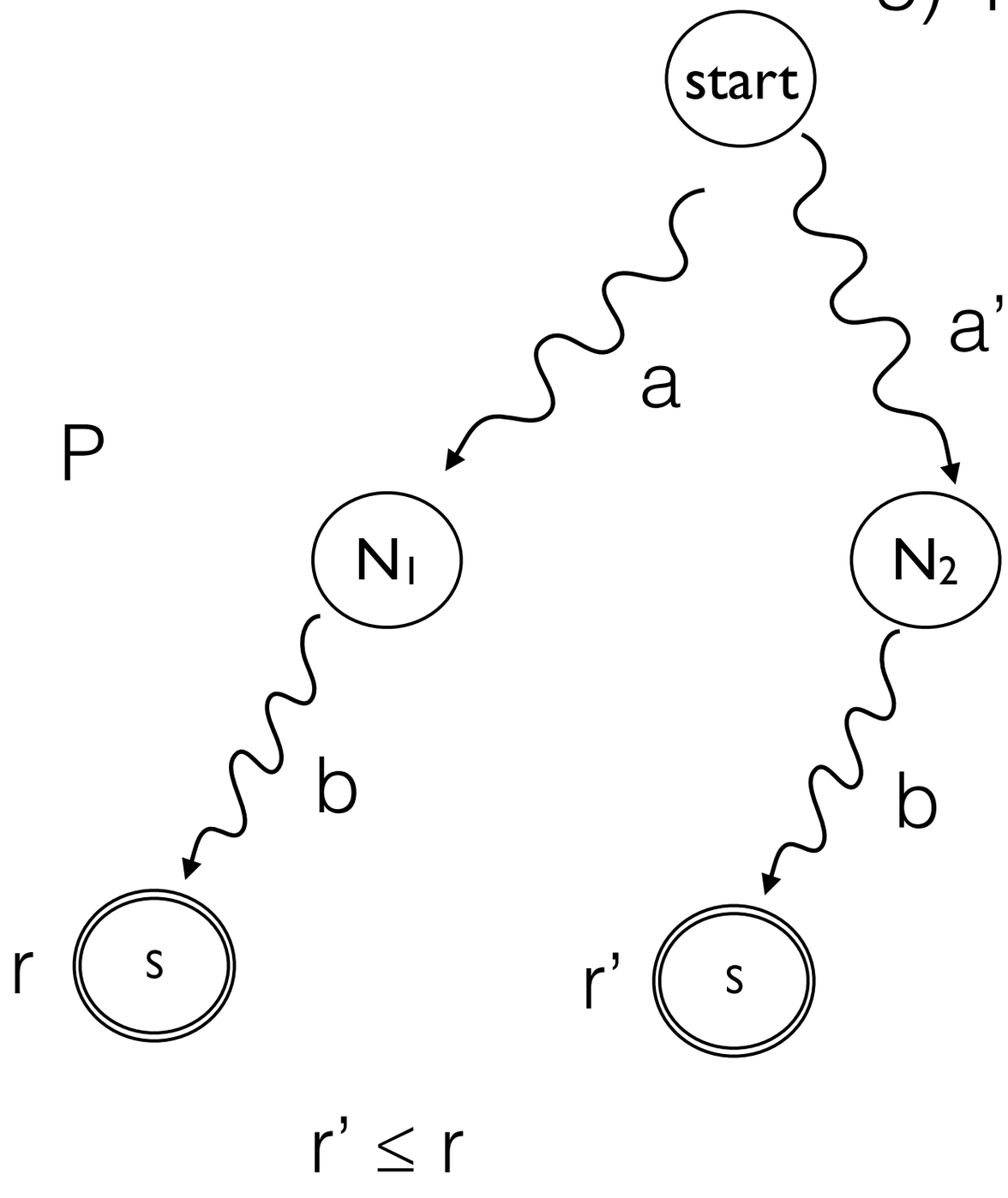
What we know:

1) advertisement reaches N_2
2) Failure analysis prefers $N_2 \geq N_1$
3) N_2 has the same path b to r' $\leq$ r



What we know:

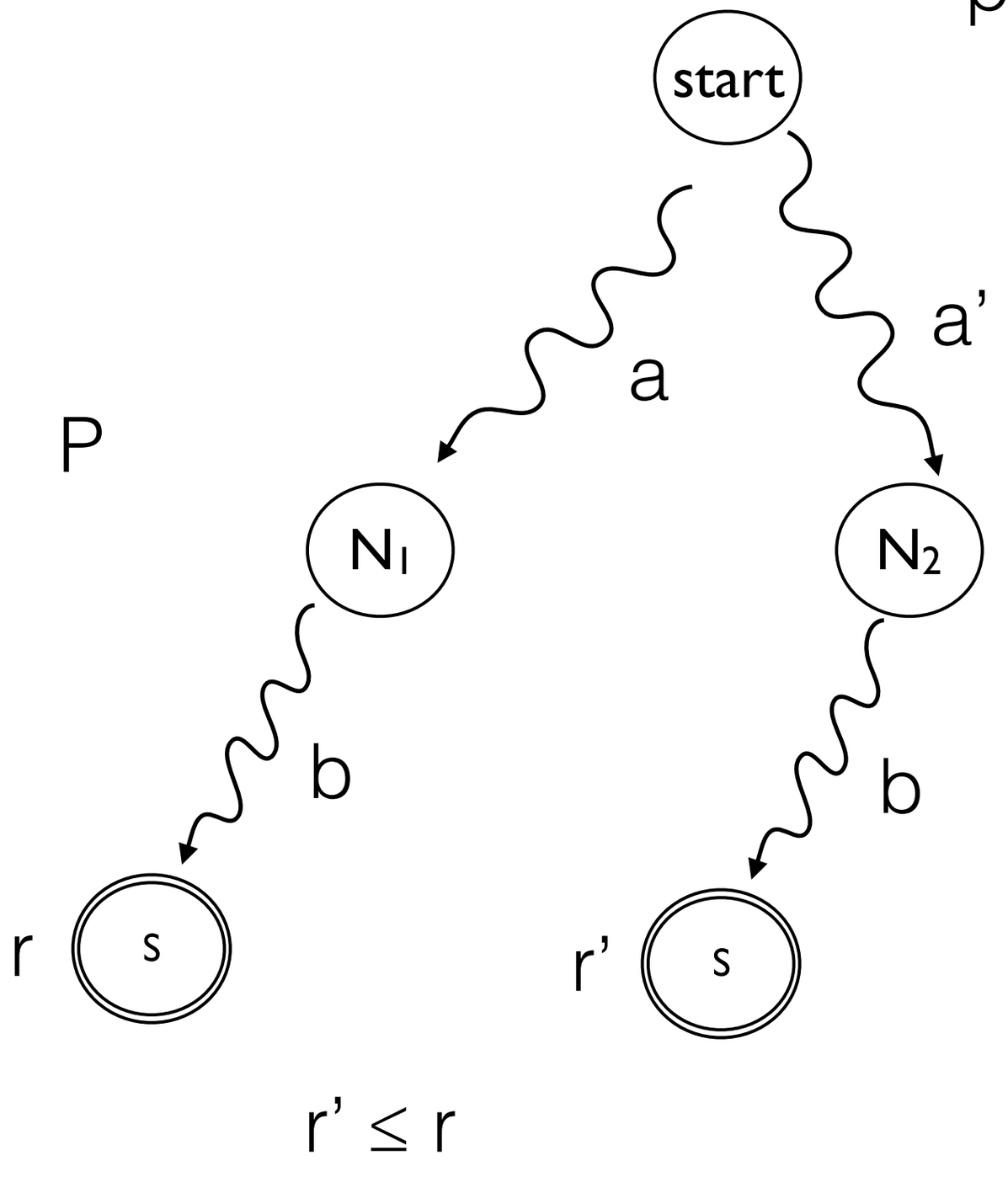
1) advertisement reaches N_2
2) Failure analysis prefers $N_2 \geq N_1$
3) N_2 has the same path b to r' $\leq$ r



r' $\leq$ r

Case 1 (a'.b is a simple path)

Proceed by induction on path length of b

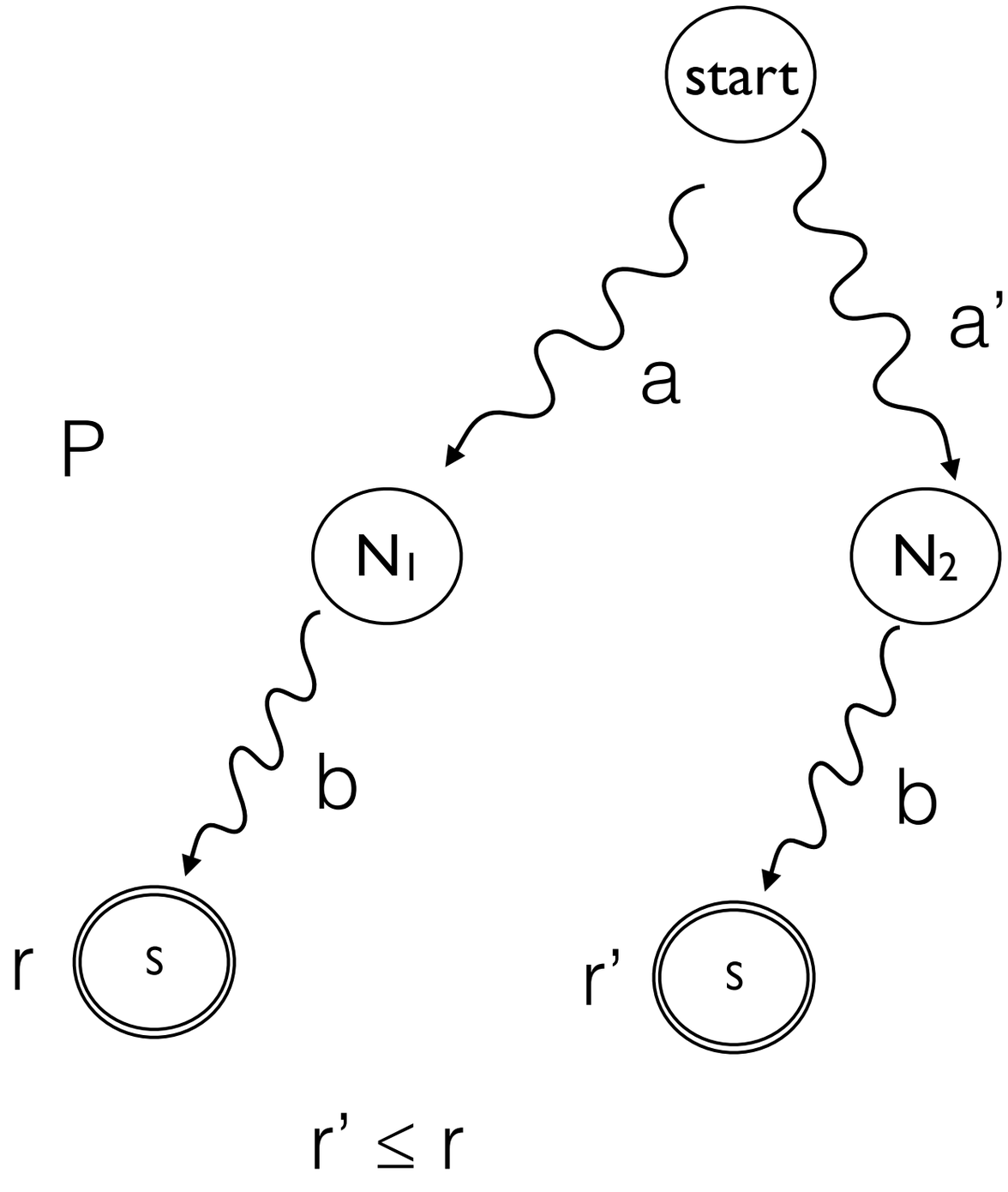


$r' \leq r$

Case 2 (a'.b is **not** simple)

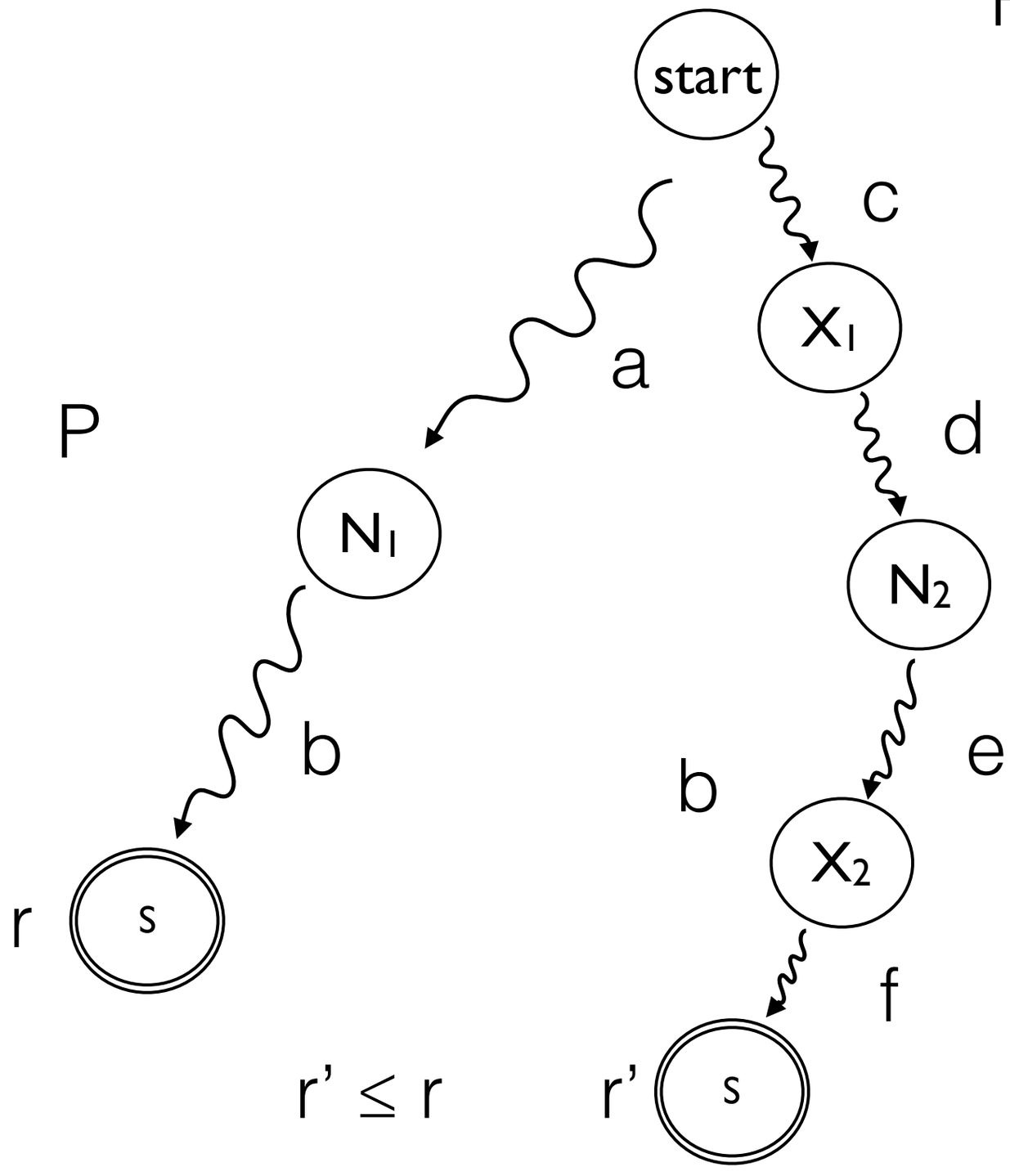
a' must be simple (advertisement)

b must be simple (since P is simple)



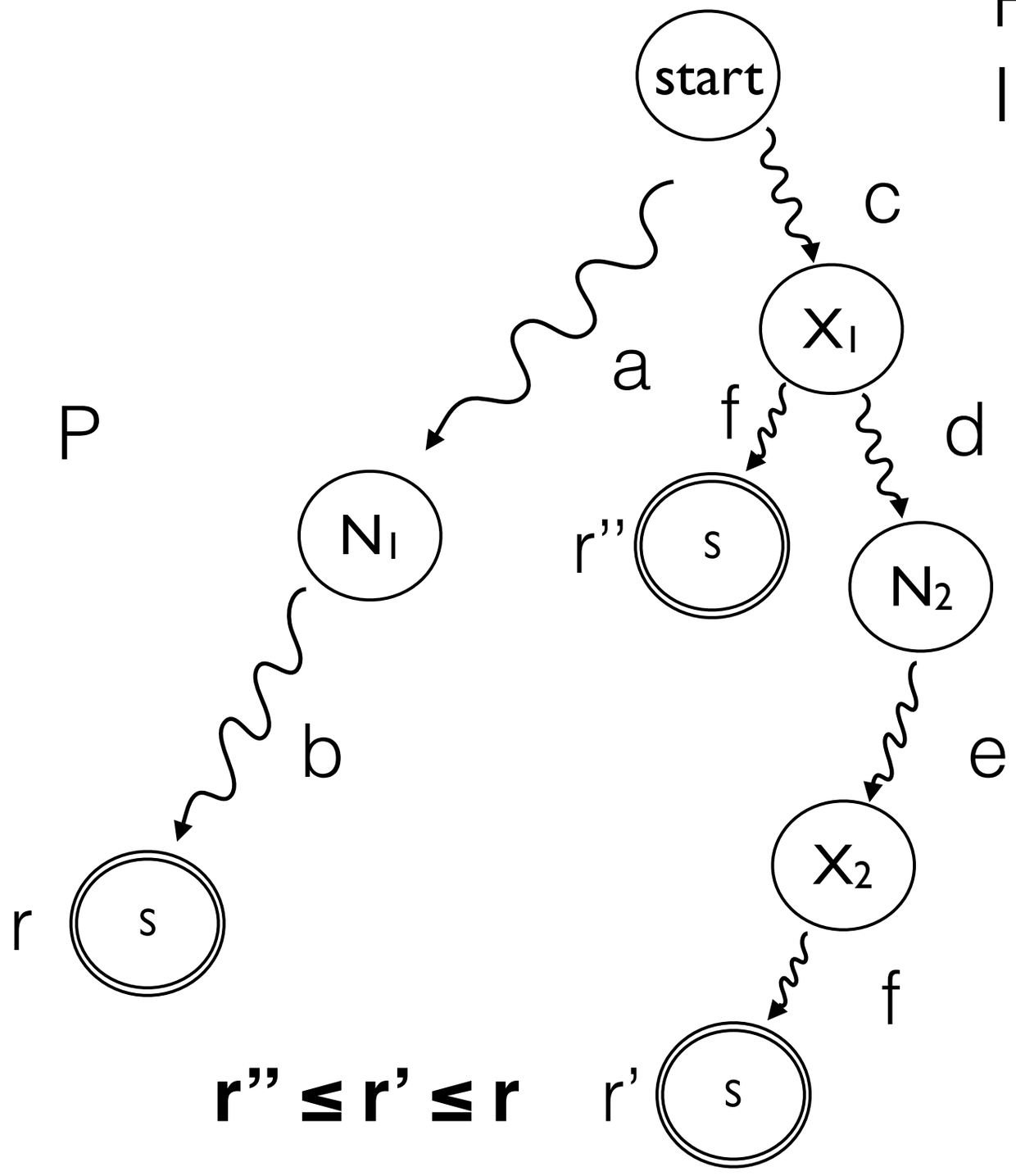
Case 2 (a'.b is **not** simple)

From failure analysis, $X_1 > X_2$



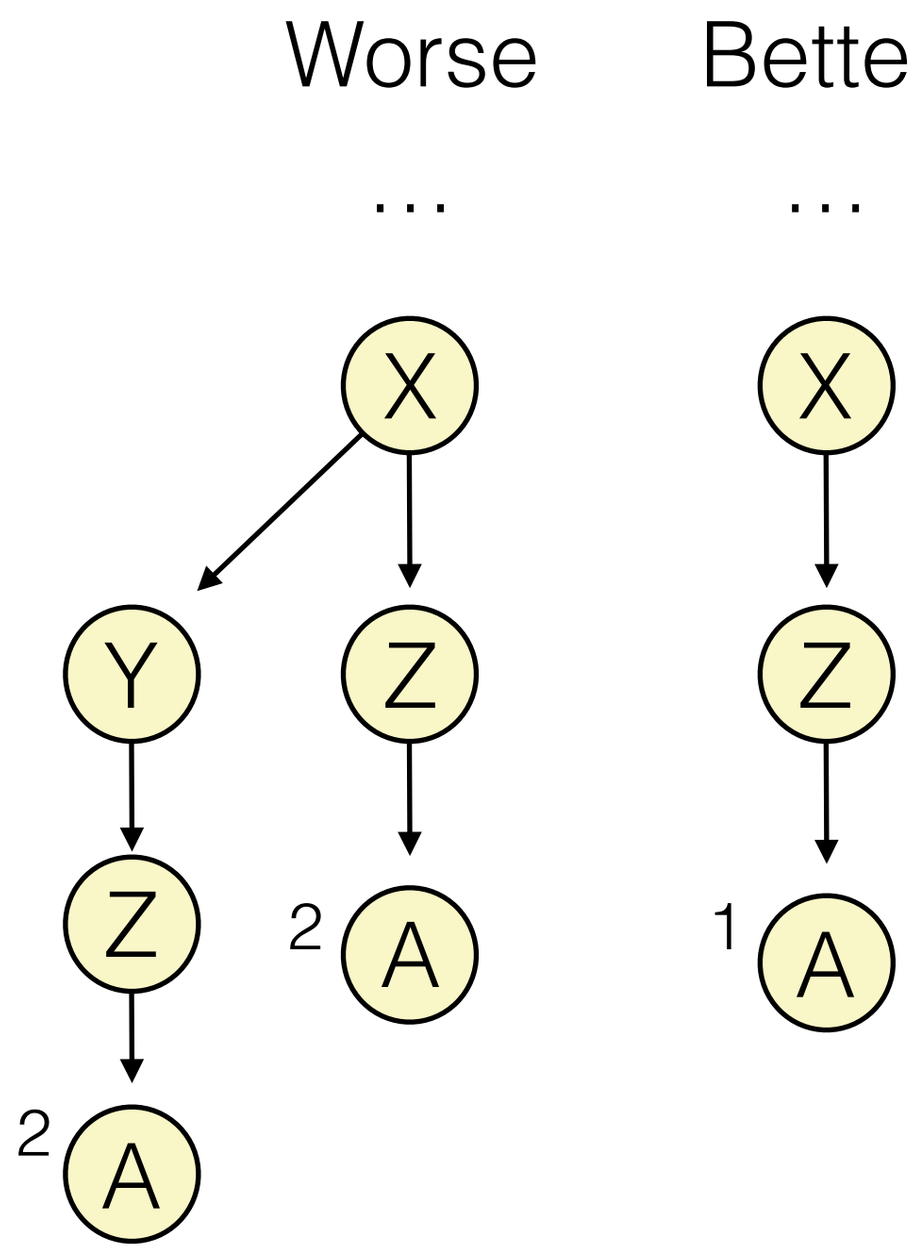
Case 2 (a'.b is **not** simple)

From failure analysis, $X_1 > X_2$
Induction on smaller paths c,f



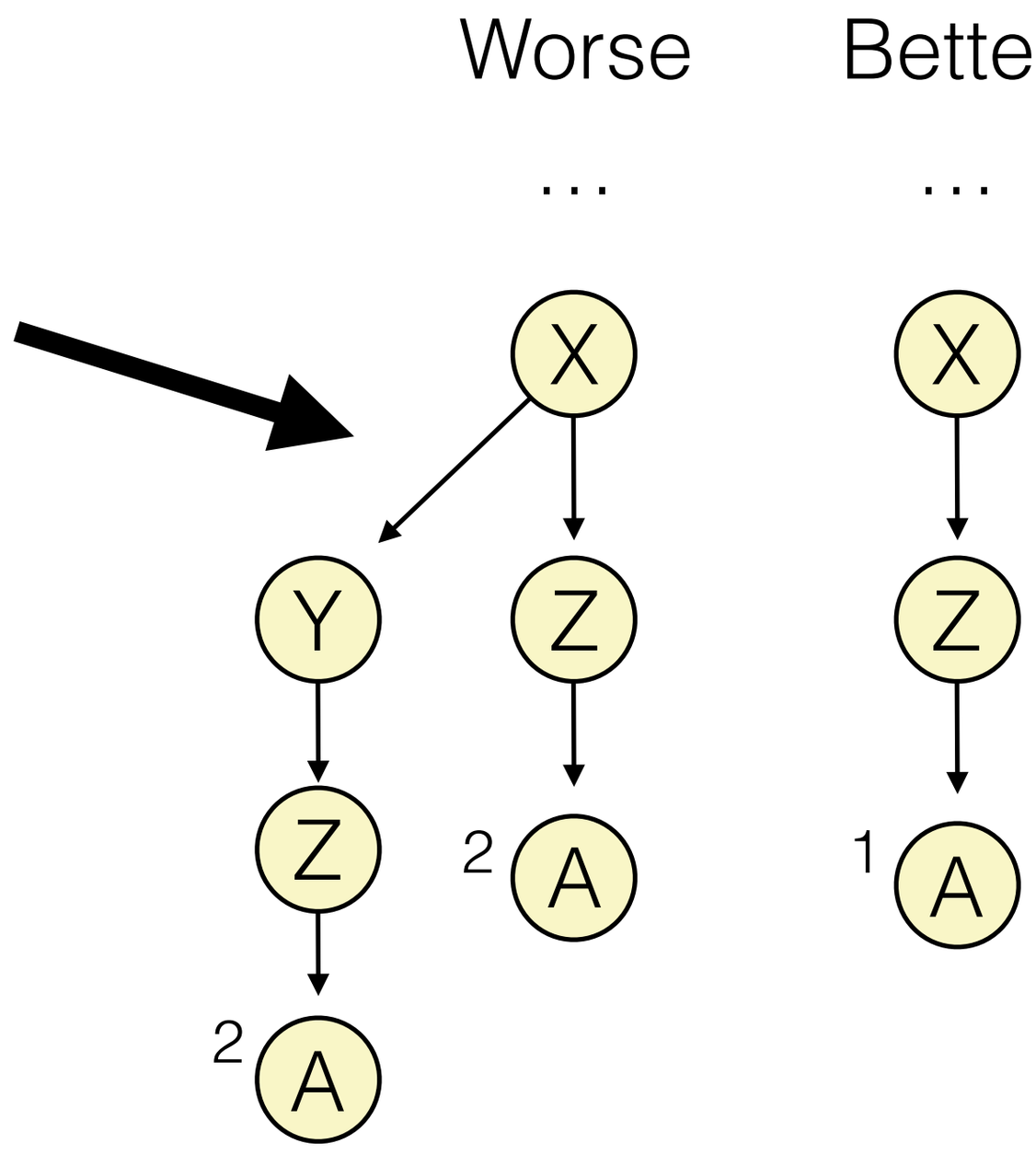
Conservative Analysis

Conservative Analysis Limits



Conservative Analysis Limits

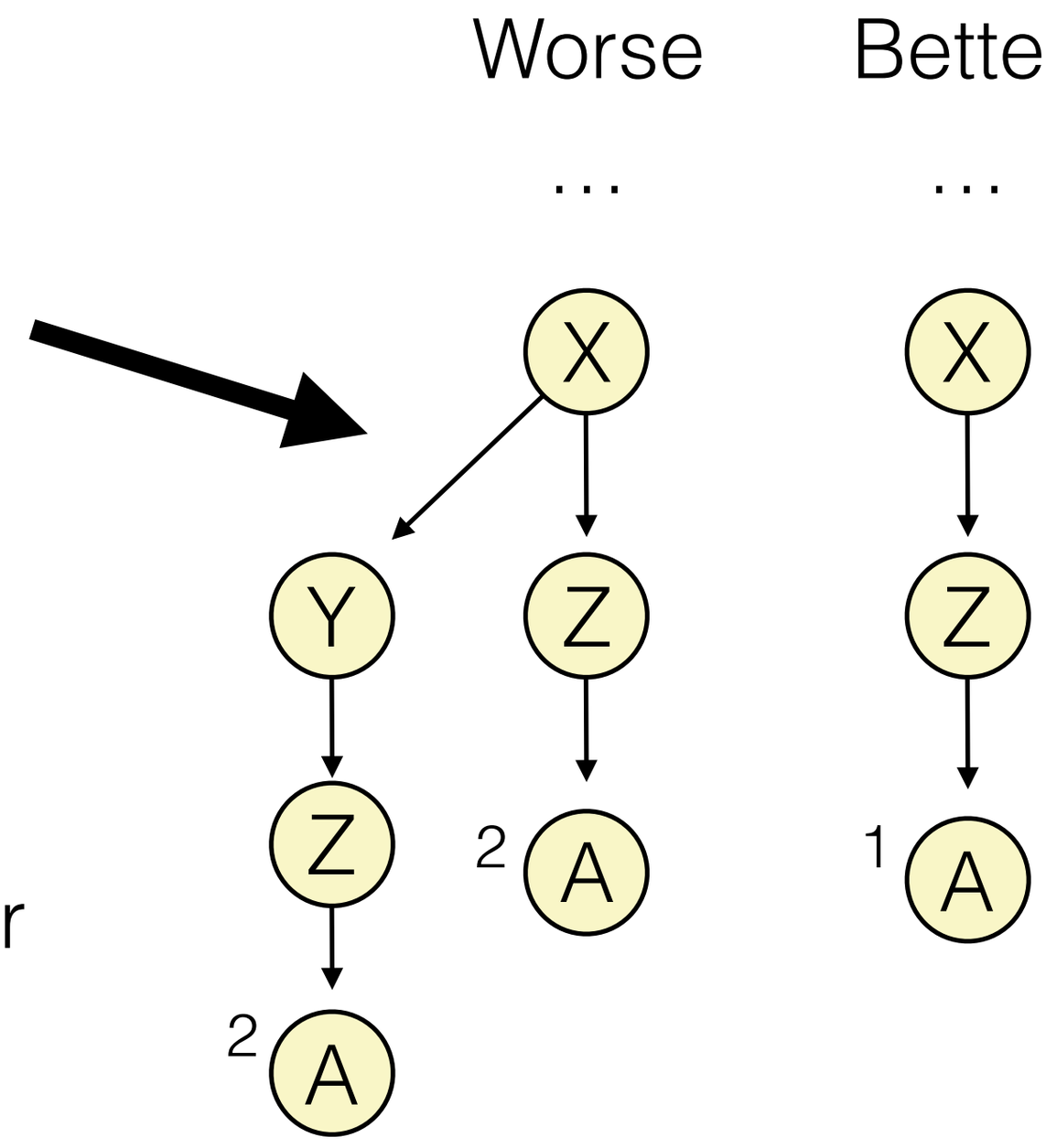
Fails because the more preferred X does not have a next hop to Y



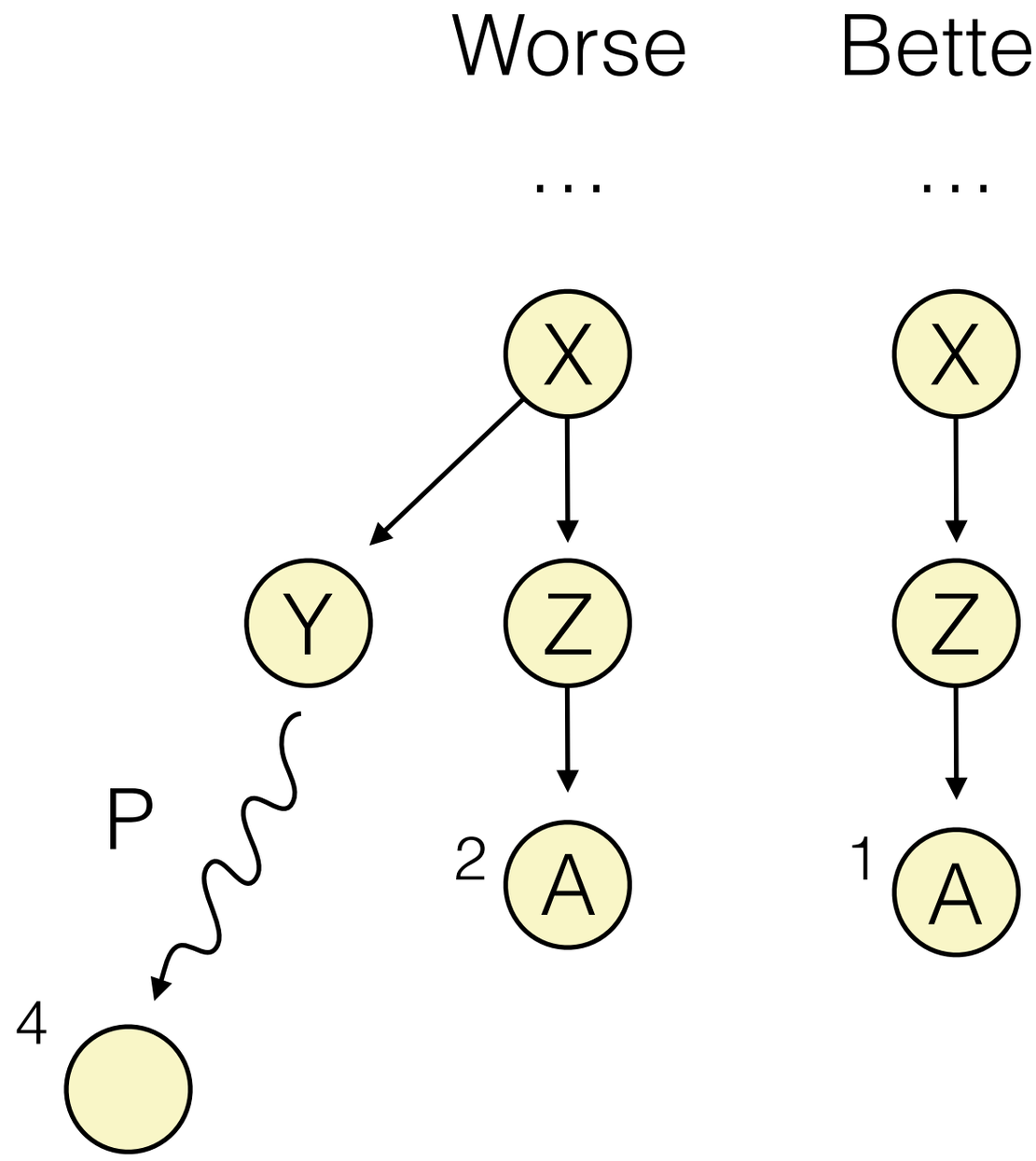
Conservative Analysis Limits

Fails because the more preferred X does not have a next hop to Y

In general, there must be an edge here that the better X does not have.

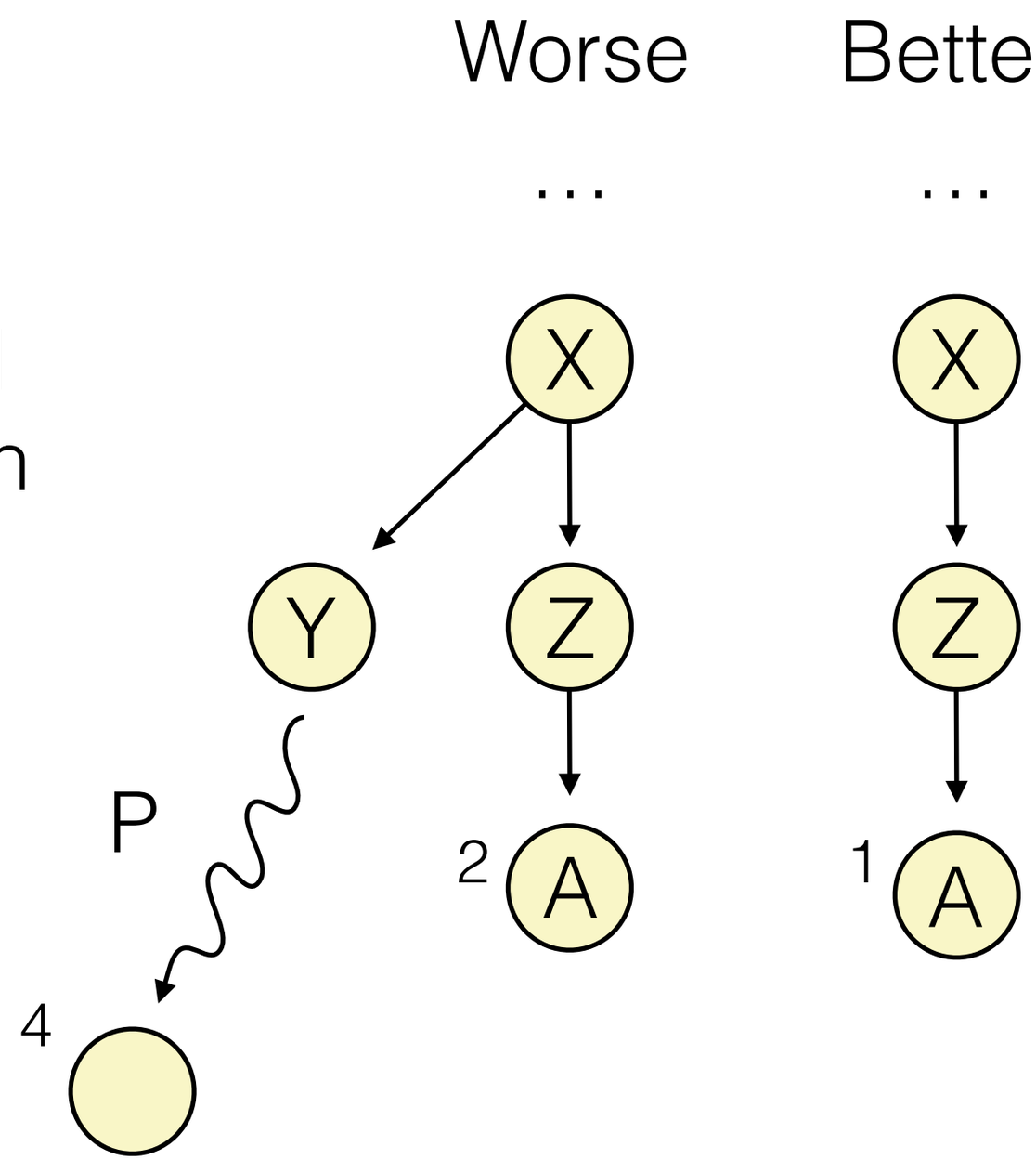


Conservative Analysis Limits



Conservative Analysis Limits

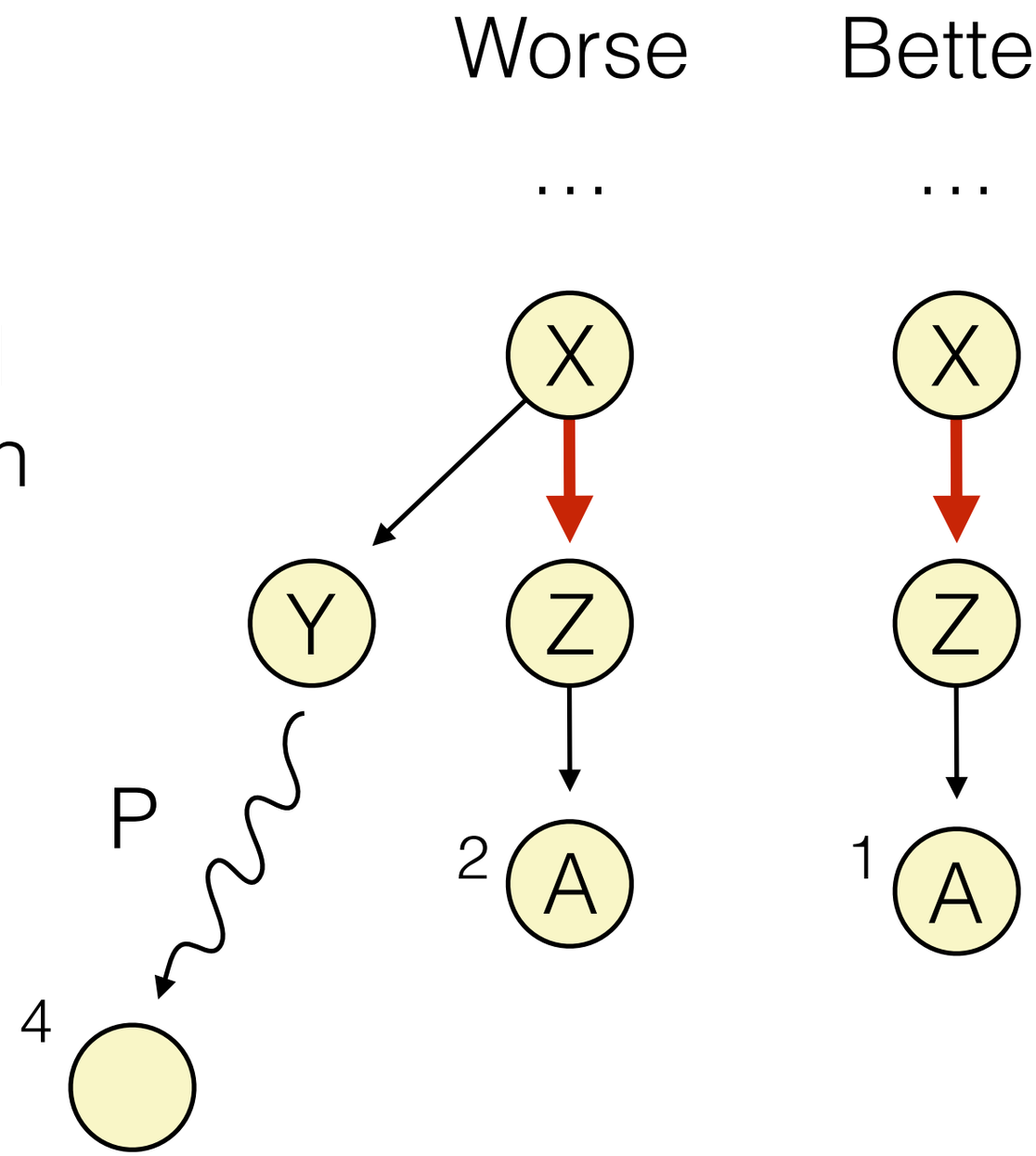
If no path P to an accepting node, we would have removed Y during minimization



Conservative Analysis Limits

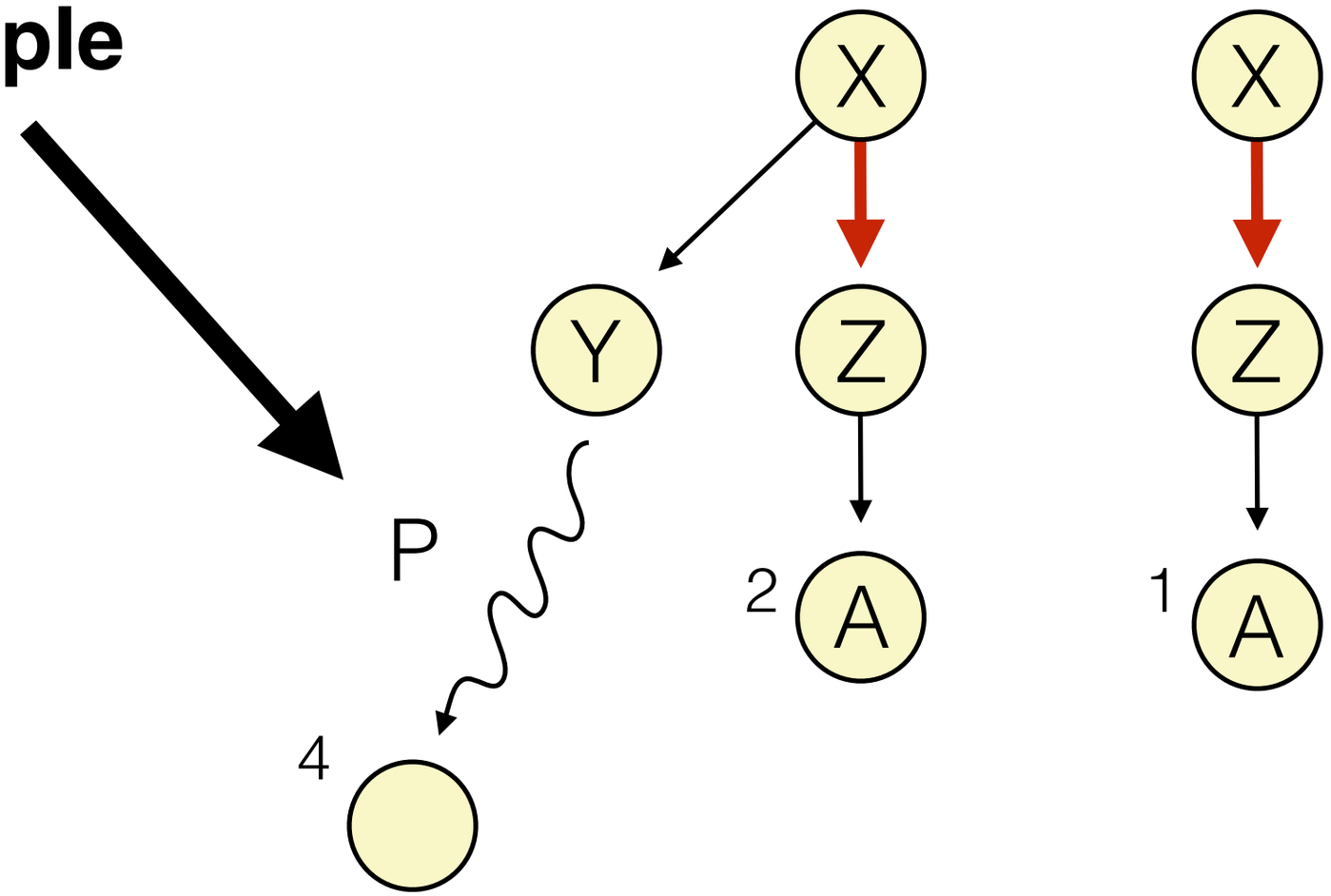
If no path P to an accepting node, we would have removed Y during minimization

Break all outgoing X links except the one different



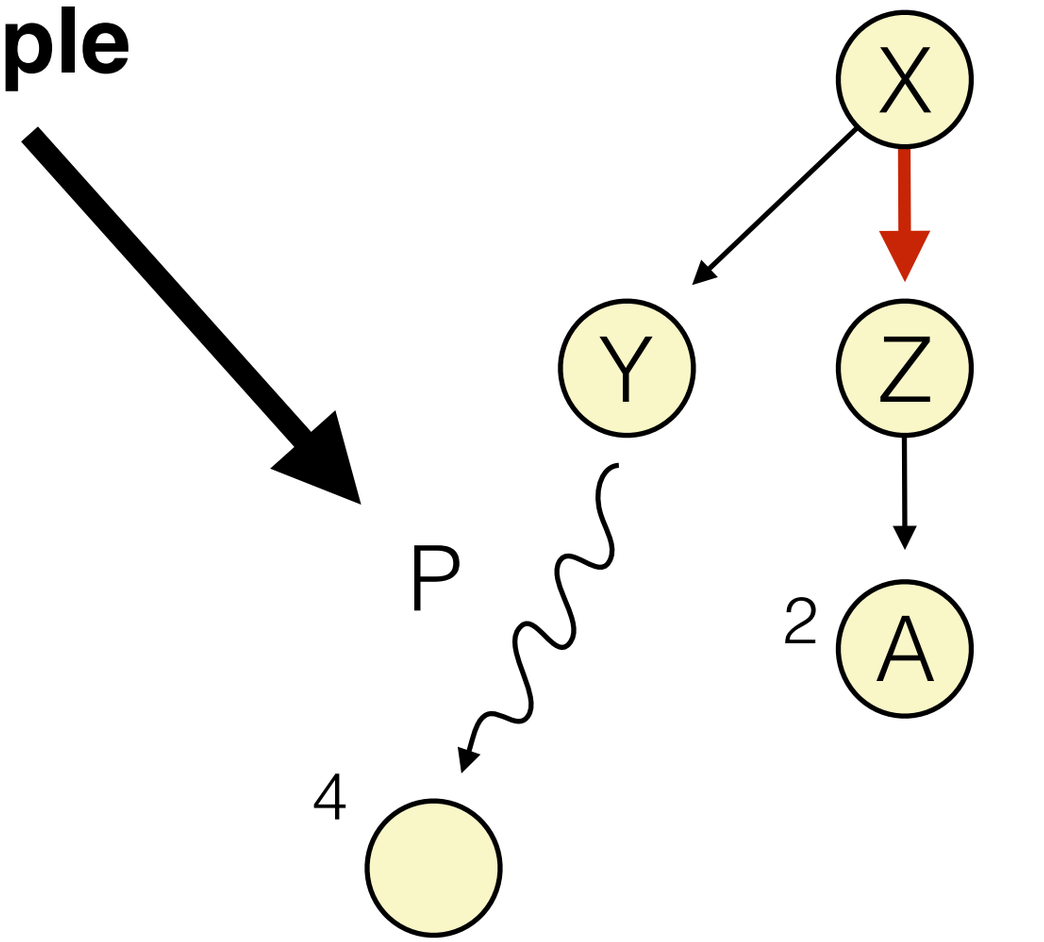
Conservative Analysis Limits

This is safe if P is ***never*** is **simple**



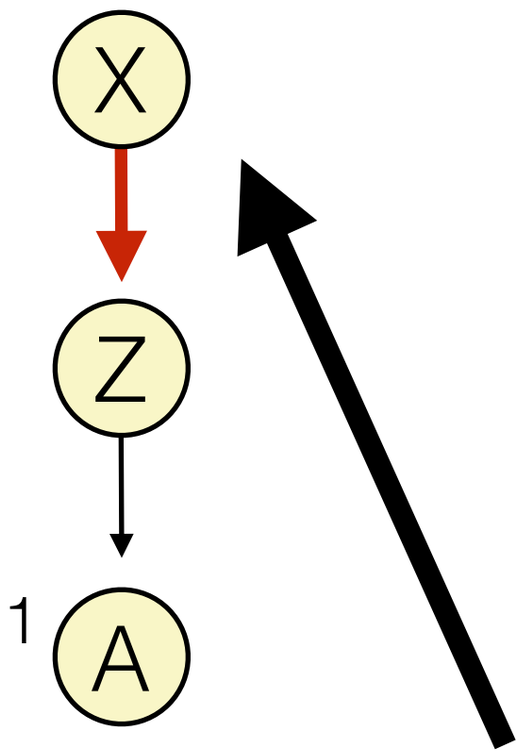
Conservative Analysis Limits

This is safe if P is ***never*** is **simple**



Worse Better

… …

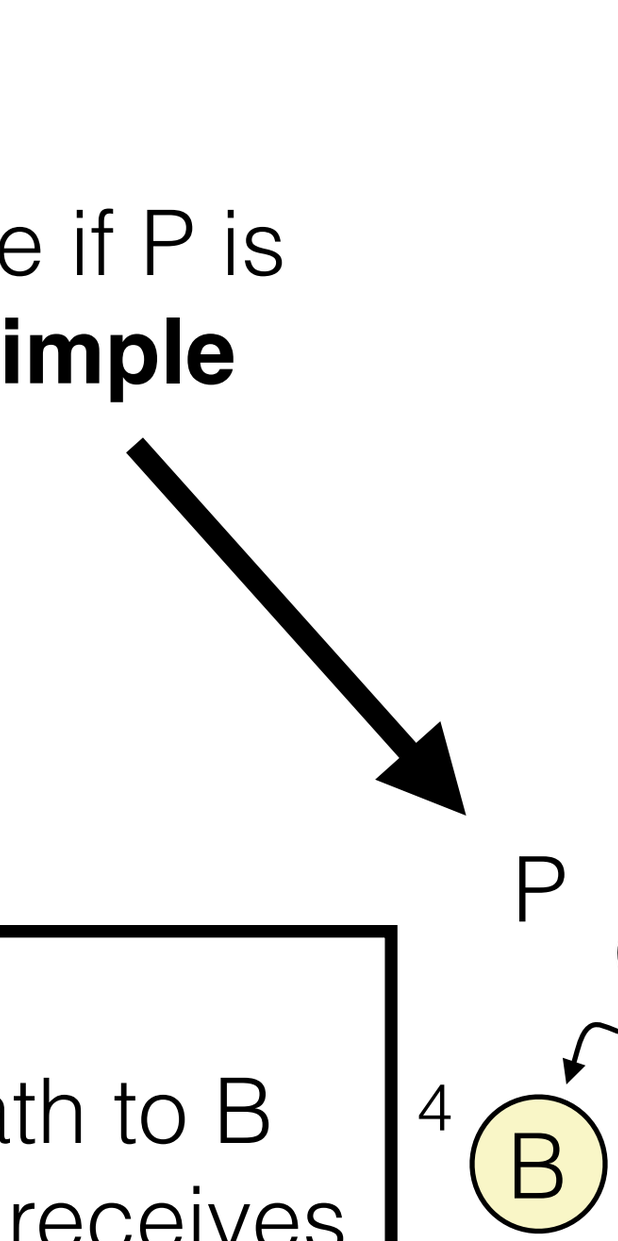


Or if there is **never** a **simple** path to X

Conservative Analysis Limits

This is safe if P is ***never*** is **simple**

Or if there is **always** a path to B whenever X receives an advertisement

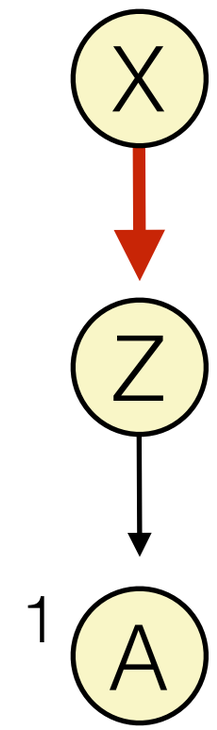


Worse

…

Better

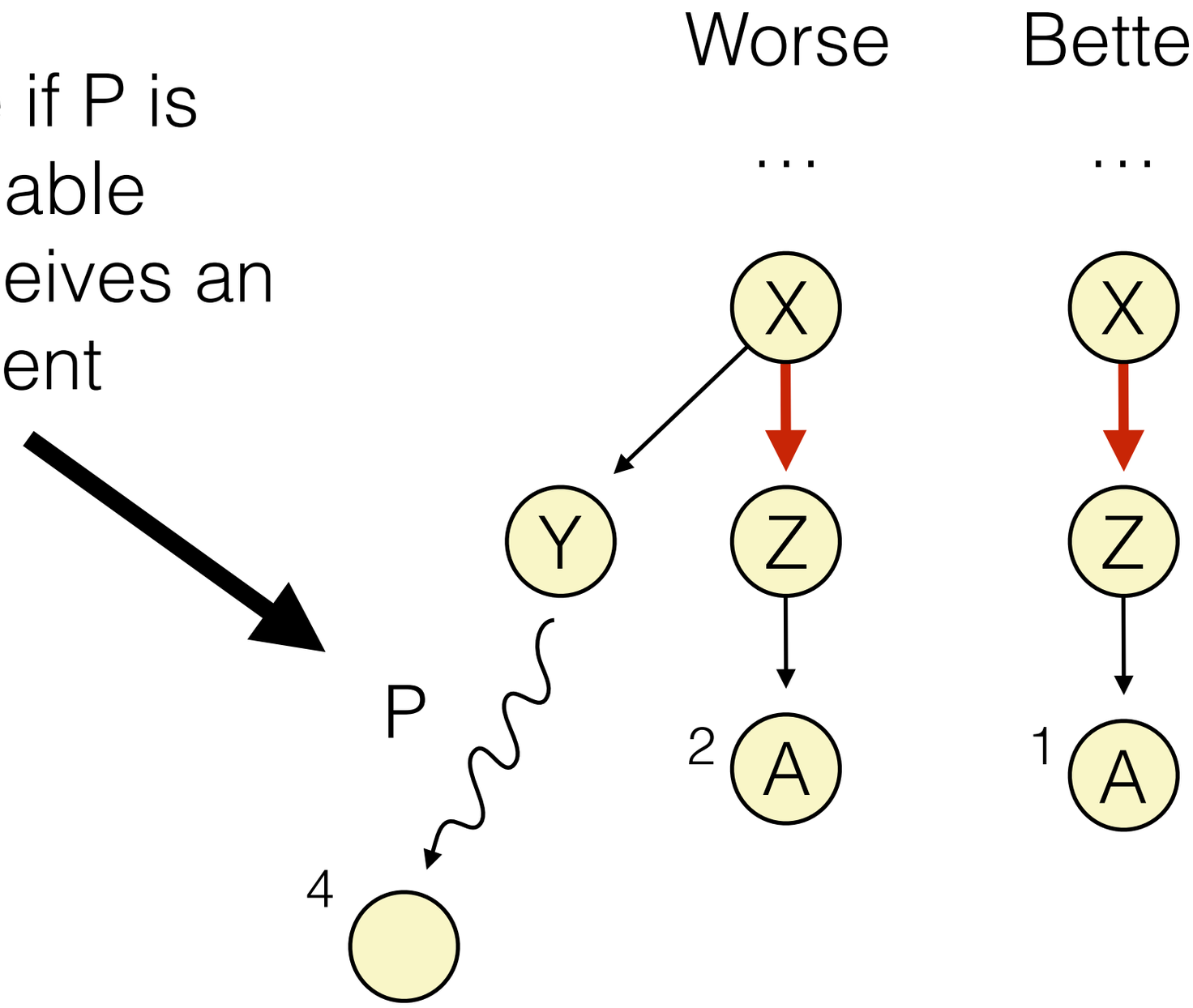
…



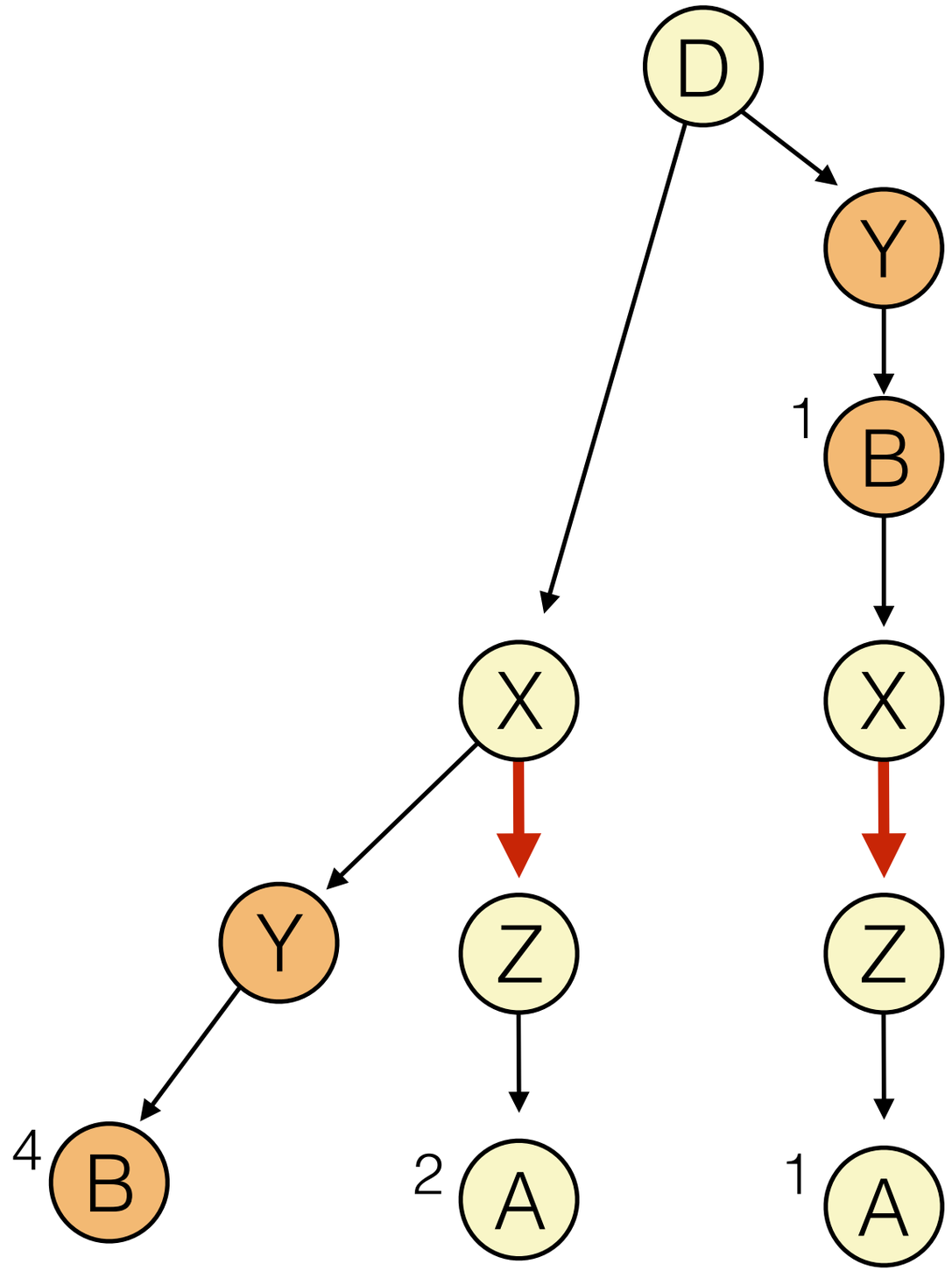
Or if there is **never** a **simple** path to X

Conservative Analysis Limits

This is safe if P is ***never*** available when X receives an advertisement

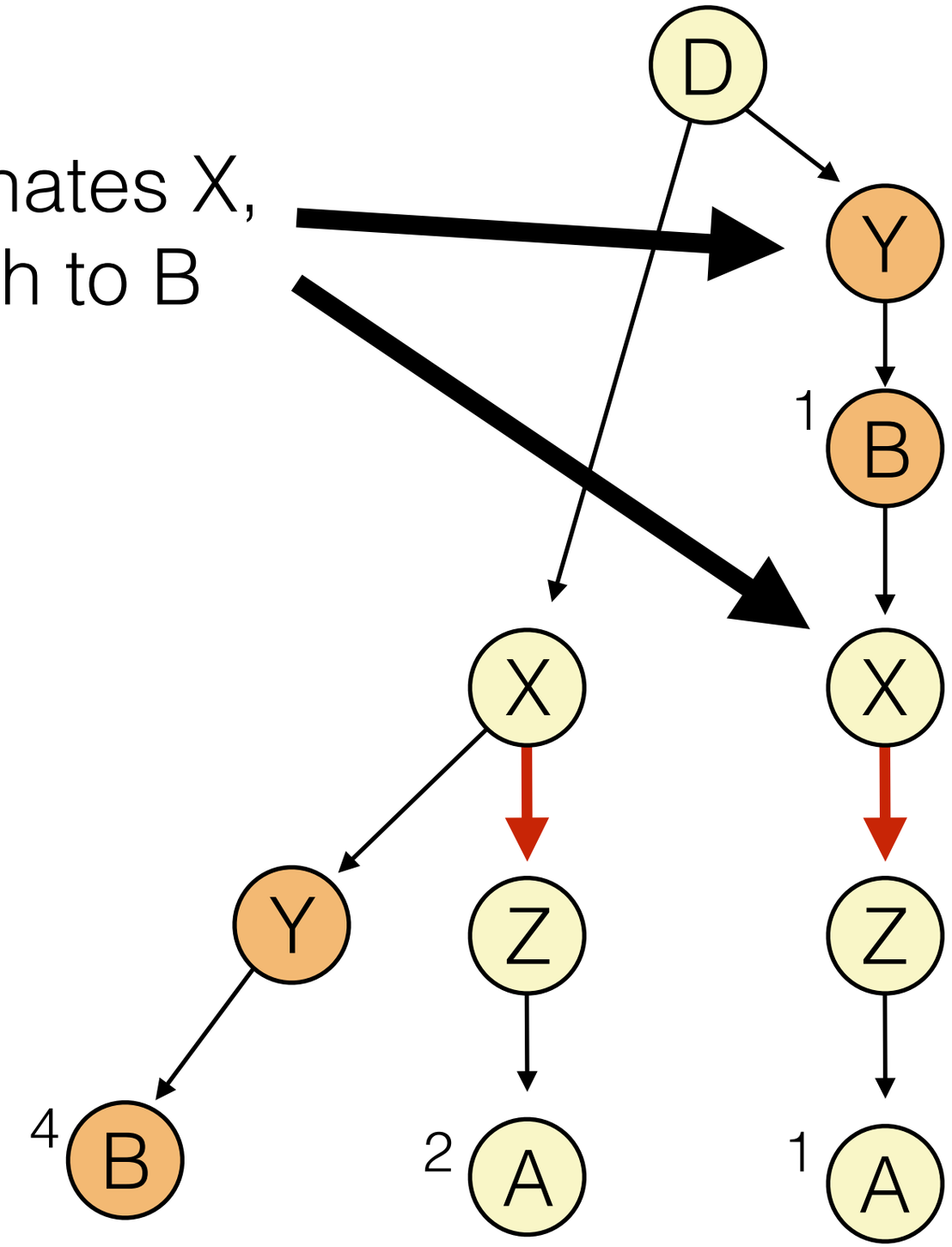


Conservative Analysis Limits

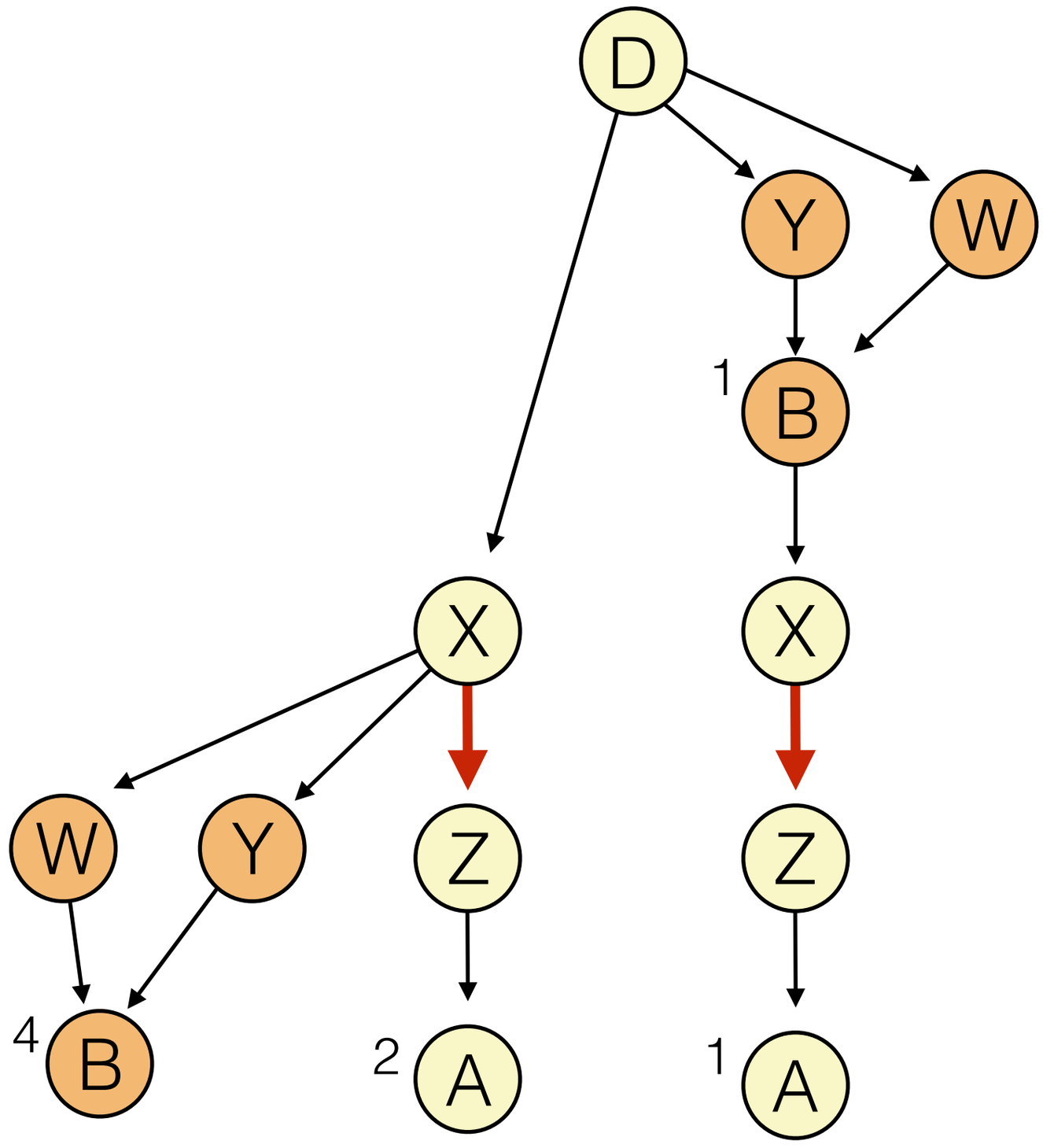


Conservative Analysis Limits

But now Y dominates X,
So there is a path to B



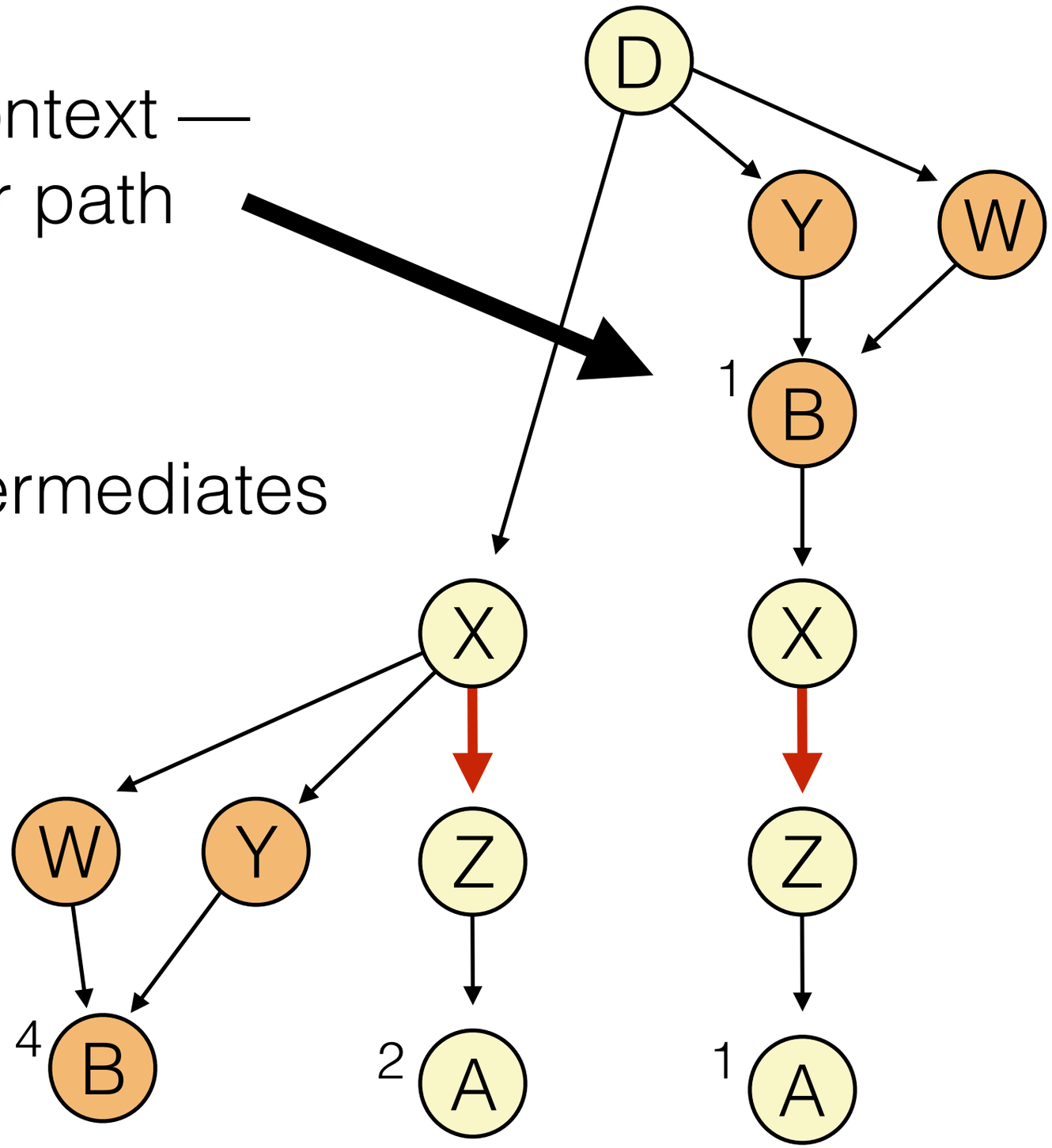
Conservative Analysis Limits



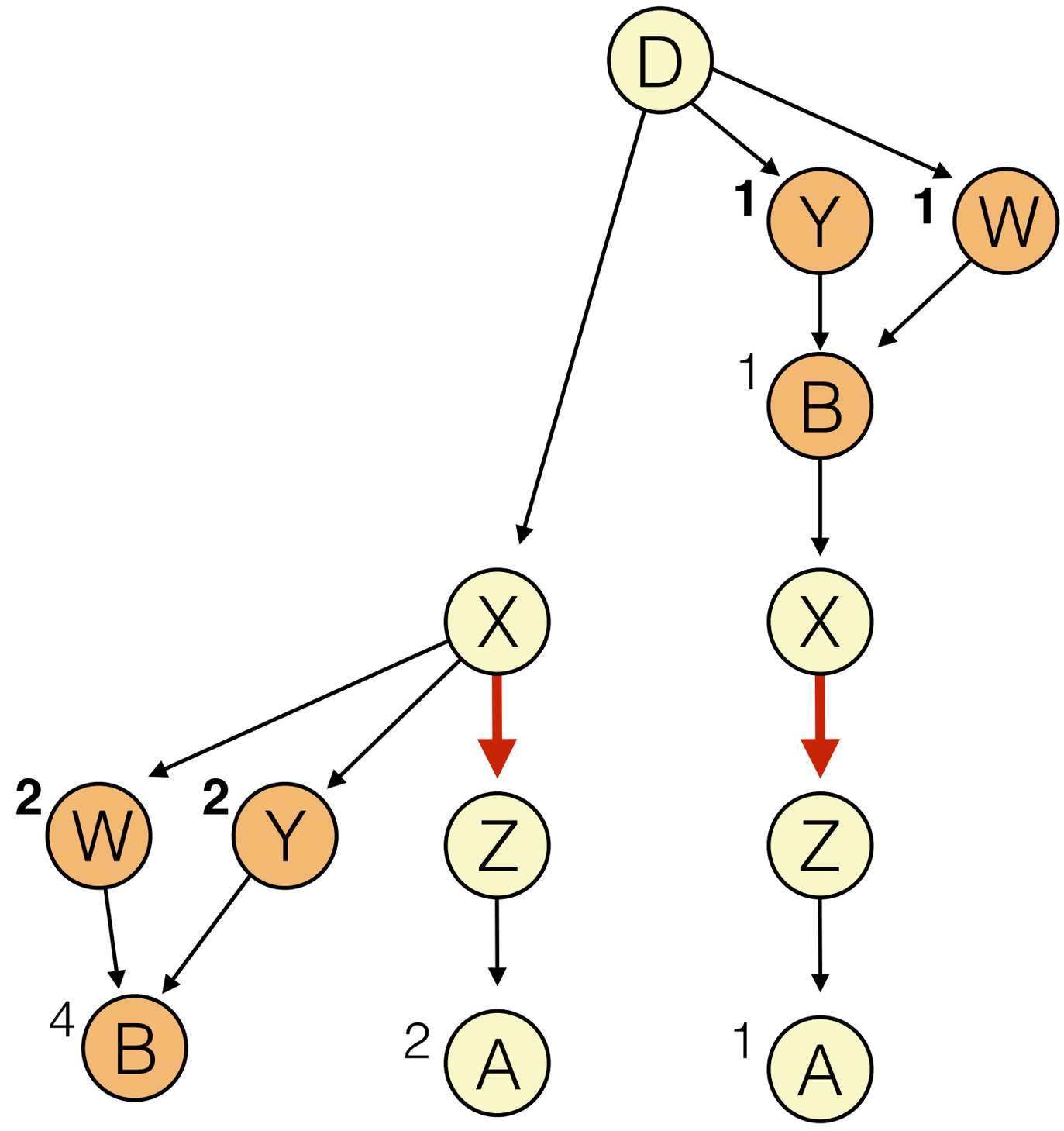
Conservative Analysis Limits

X can prefer right context —
B must have a better path

Works because we
don’t care about intermediates
Y and W

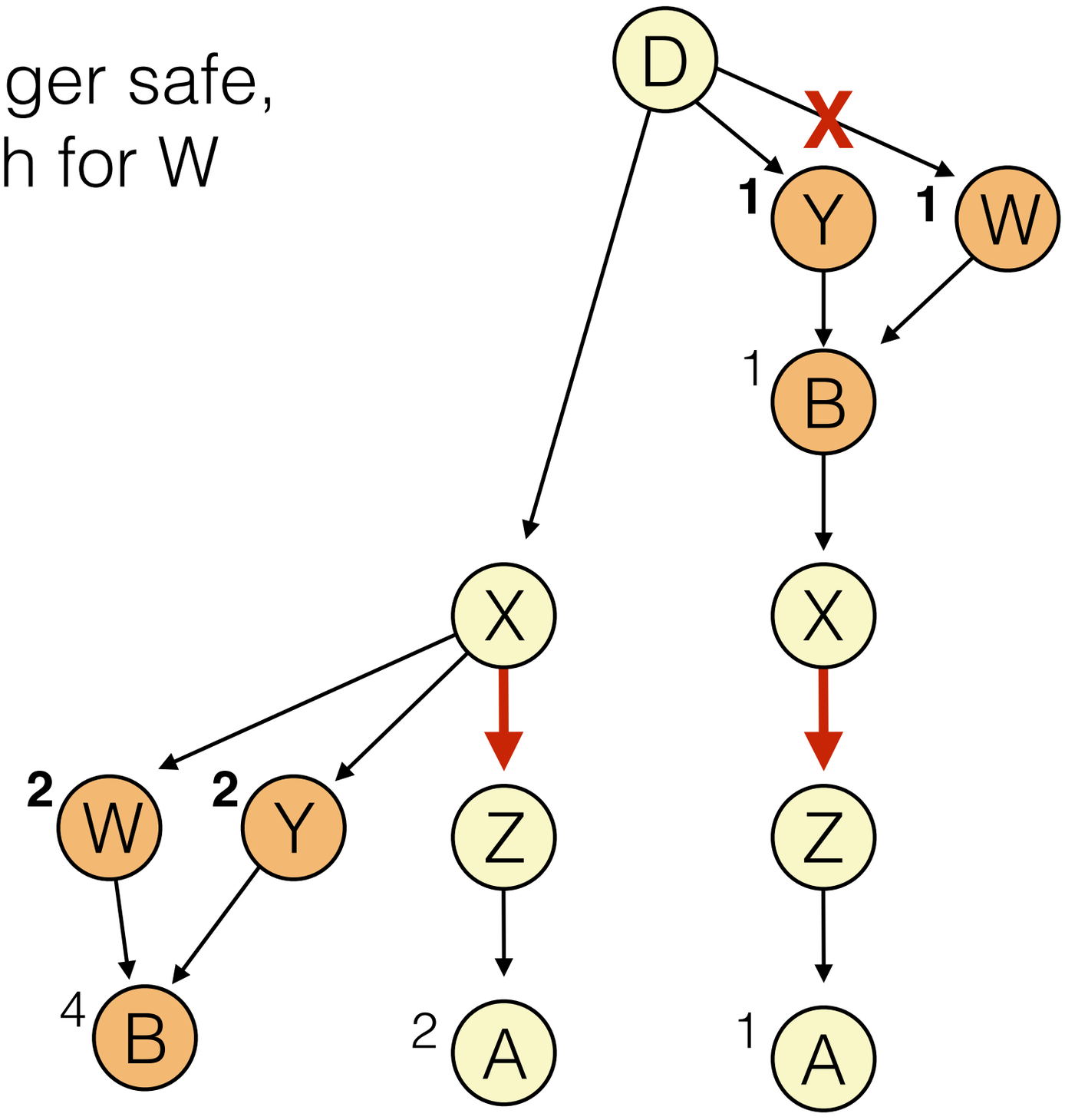


Conservative Analysis Limits



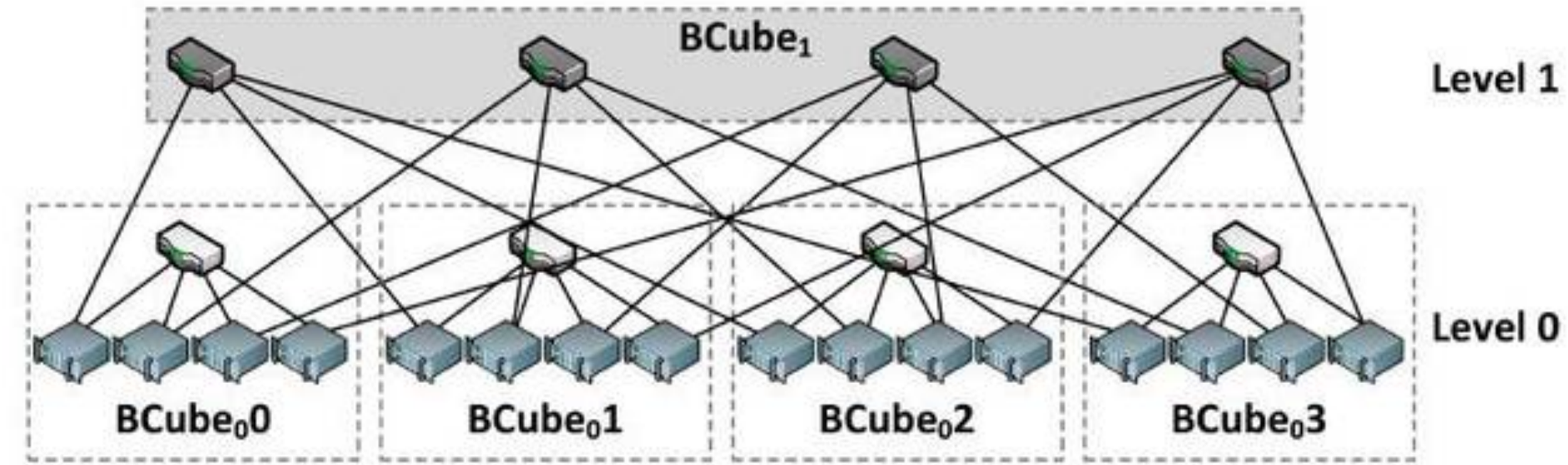
Conservative Analysis Limits

No longer safe,
no path for W

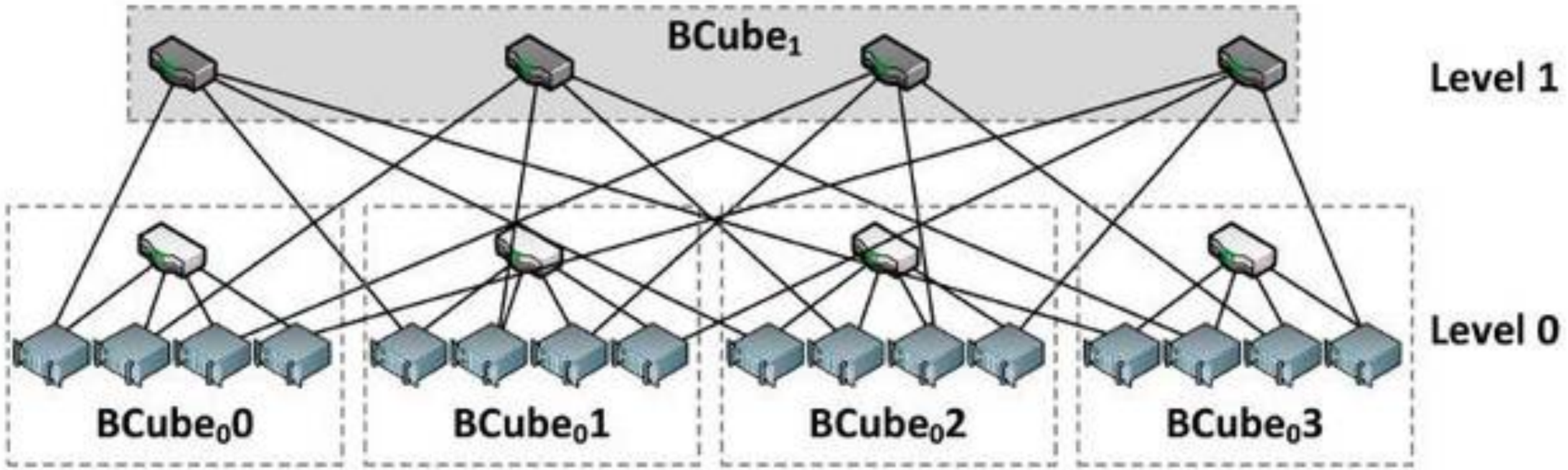


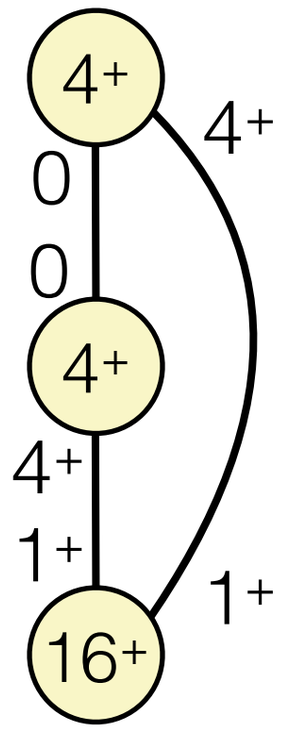
Other Abstract Topologies

Other Topologies - BCube

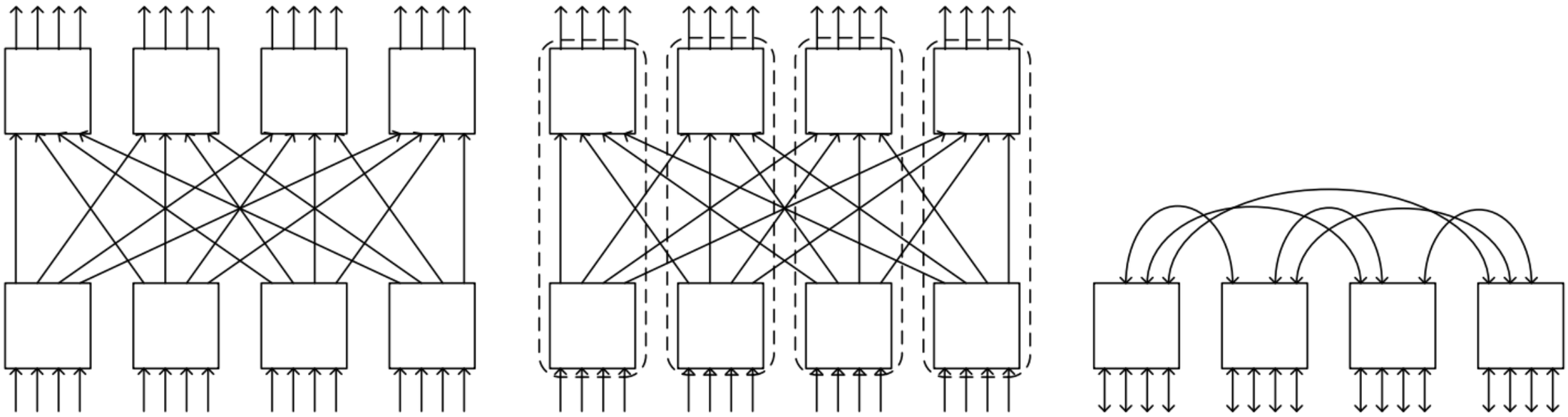


Other Topologies - BCube



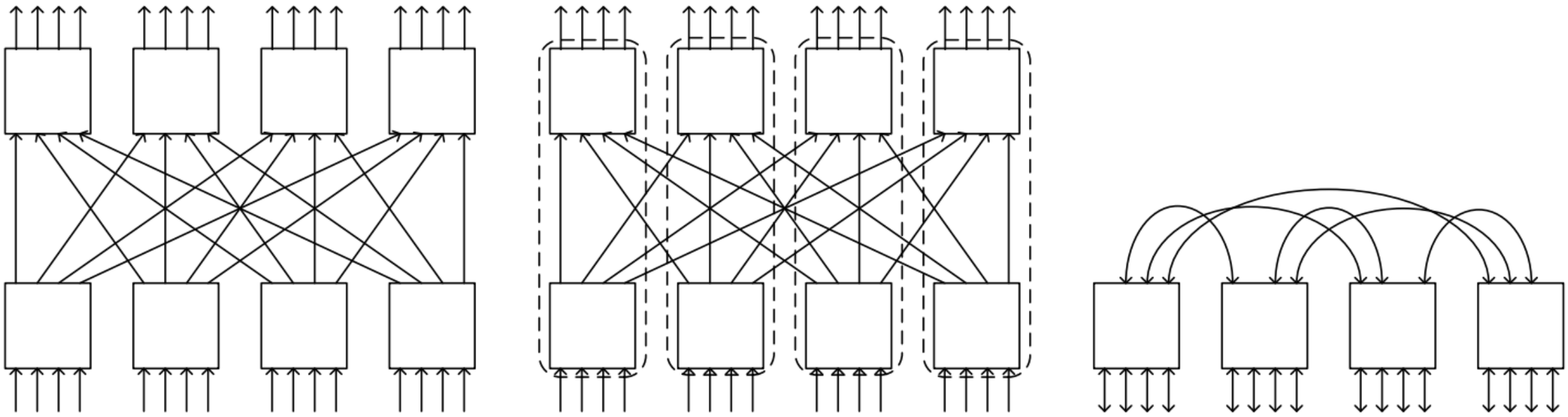


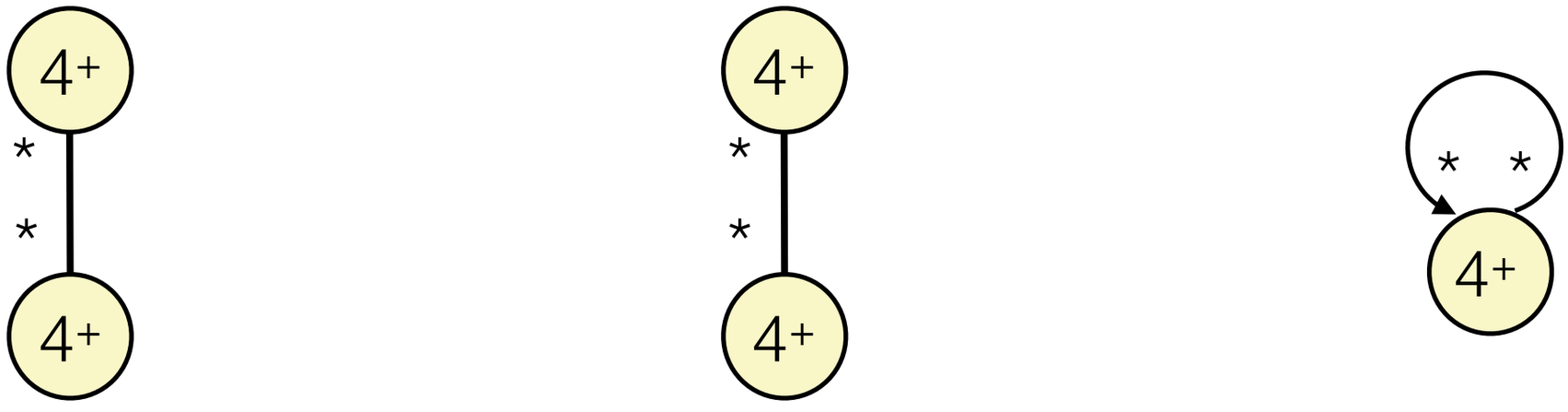
Other Topologies - Butterfly



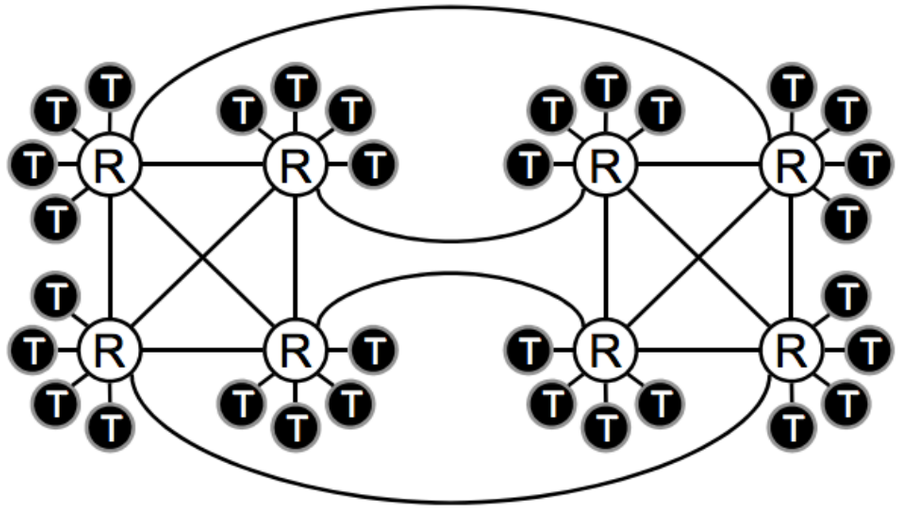
(a) (b) (c)

Other Topologies - Butterfly

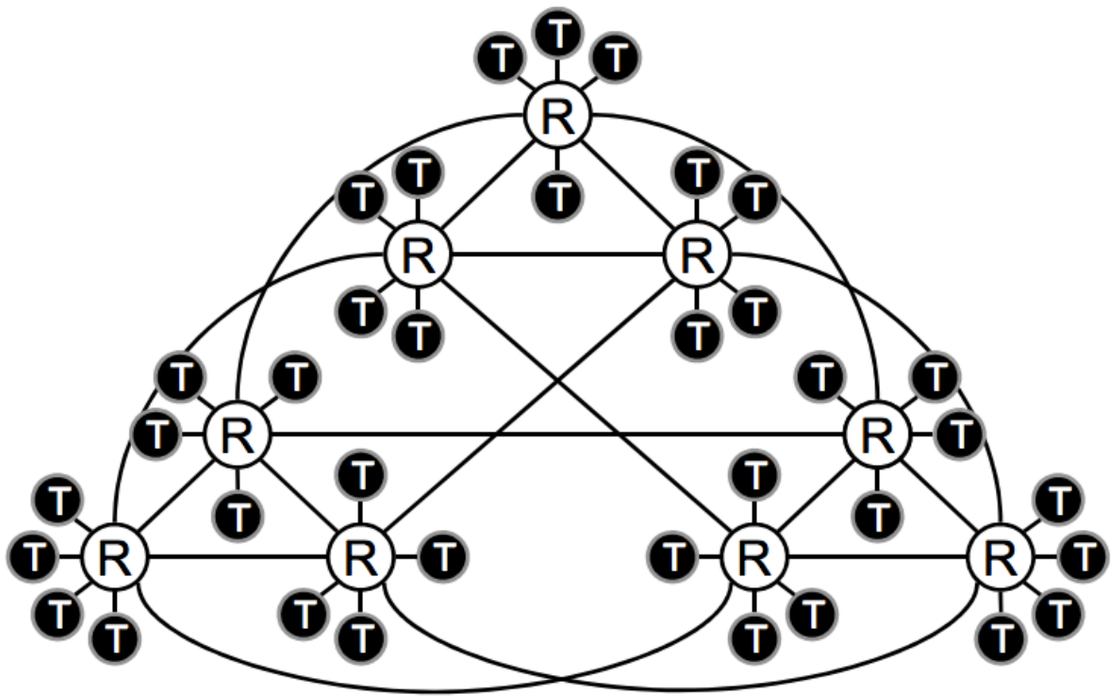




Other Topologies - HyperX

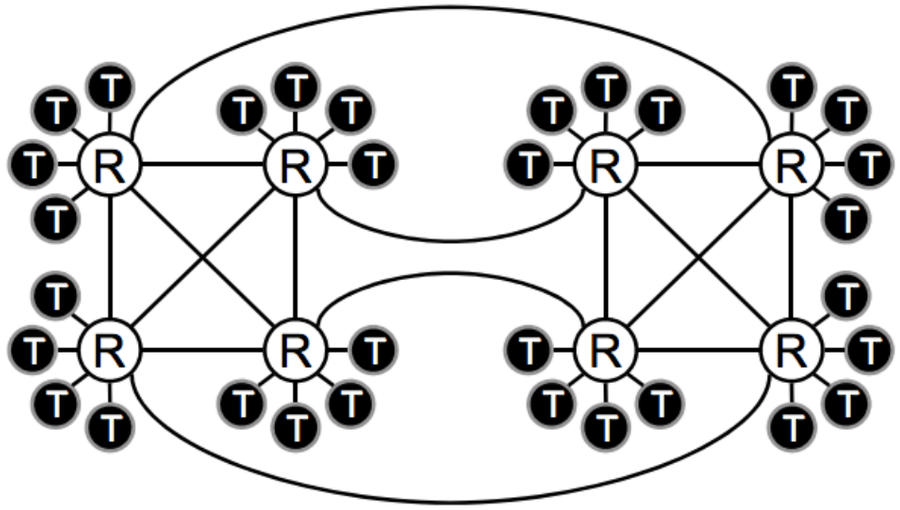


(a) $L = 2, S_1 = 2, S_2 = 4, K = 1, T = 4$

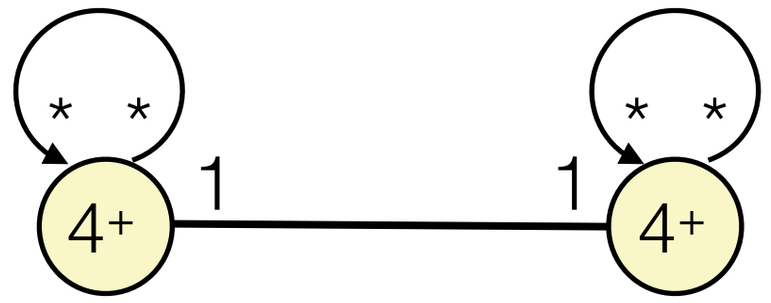


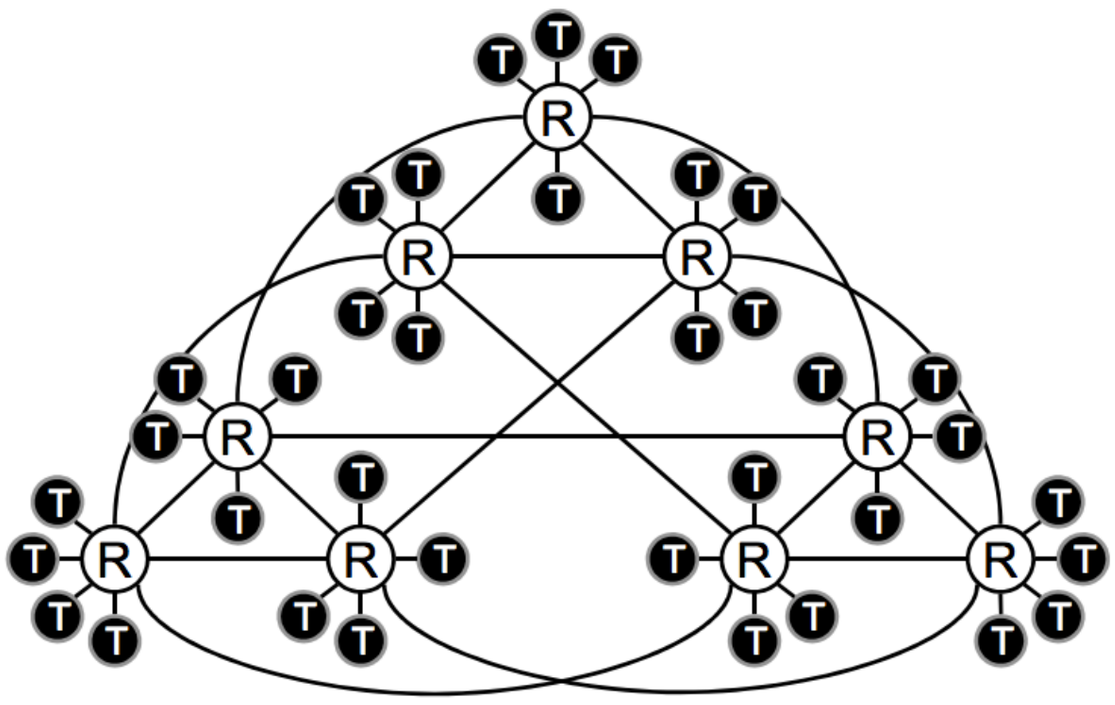
(b) $L = 2, S_1 = 3, S_2 = 3, K = 1, T = 4$

Other Topologies - HyperX

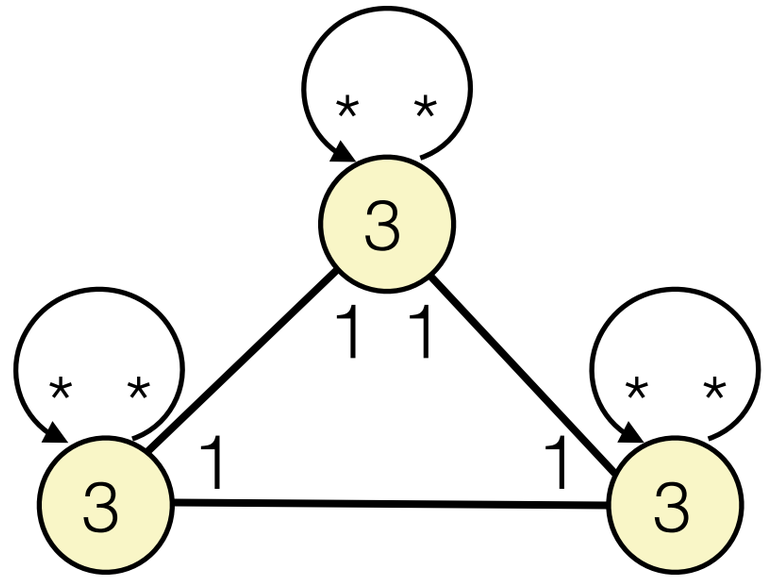


(a) $L = 2, S_1 = 2, S_2 = 4, K = 1, T = 4$

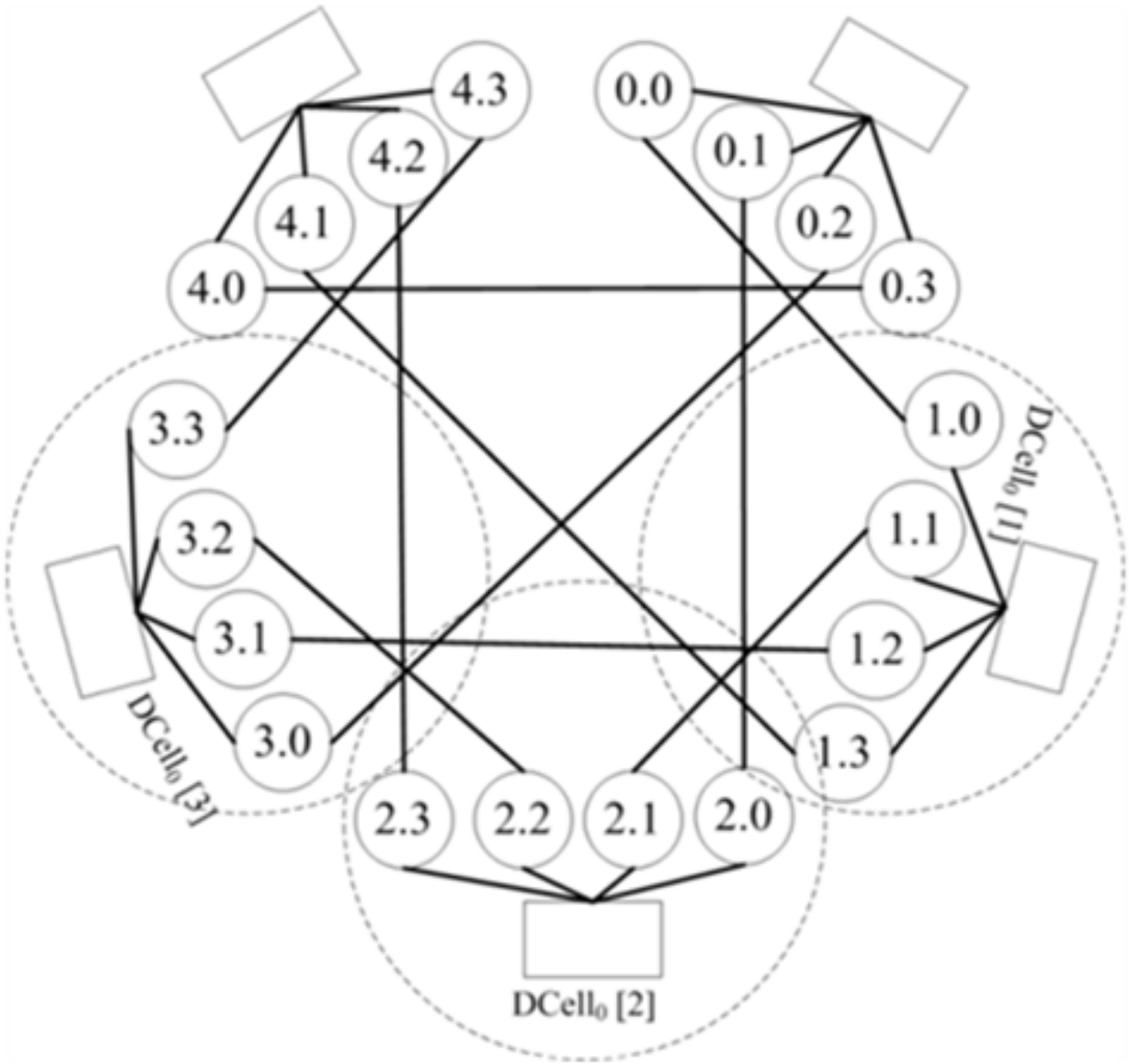




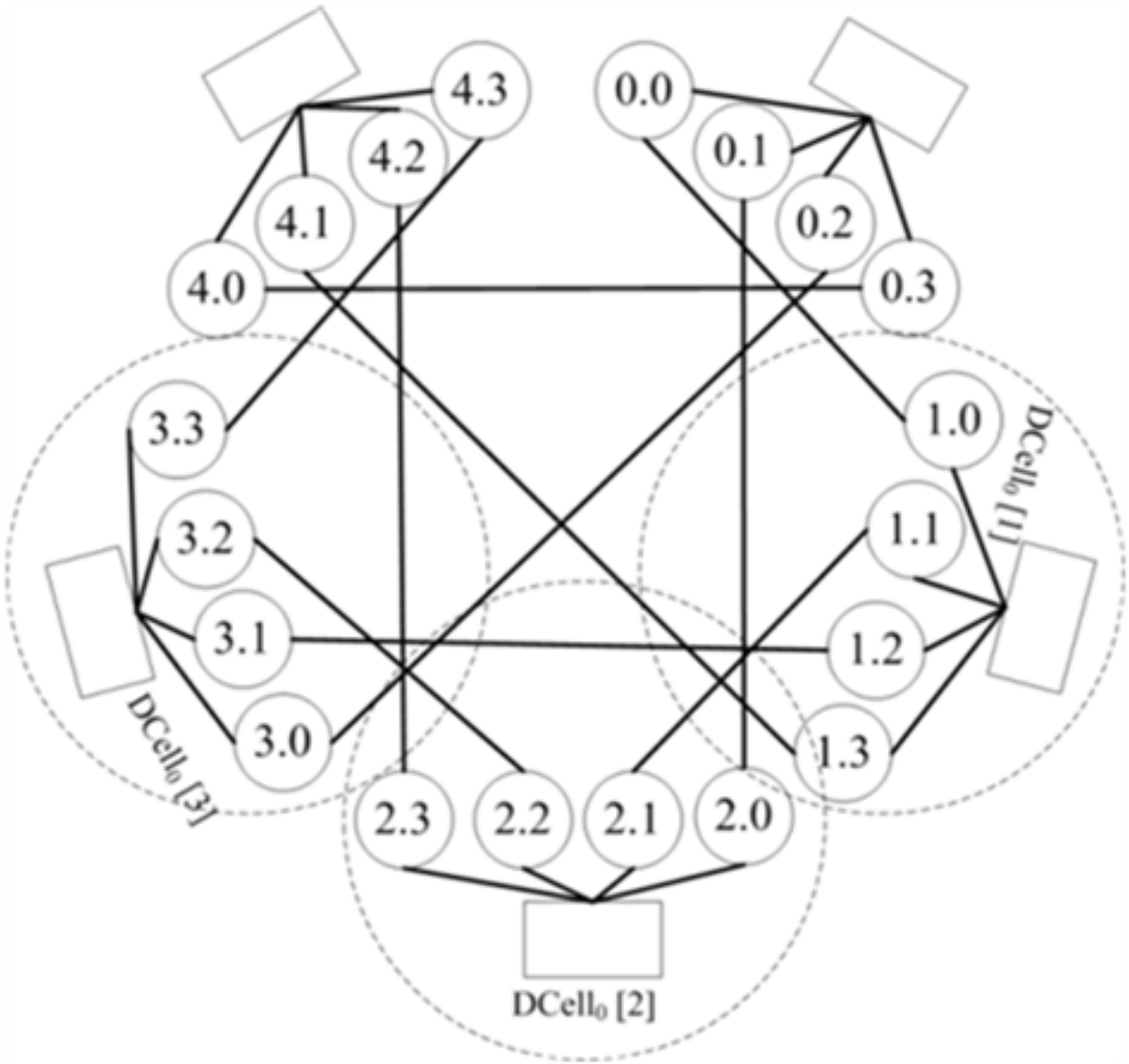
(b) $L = 2, S_1 = 3, S_2 = 3, K = 1, T = 4$



Other Topologies - DCell



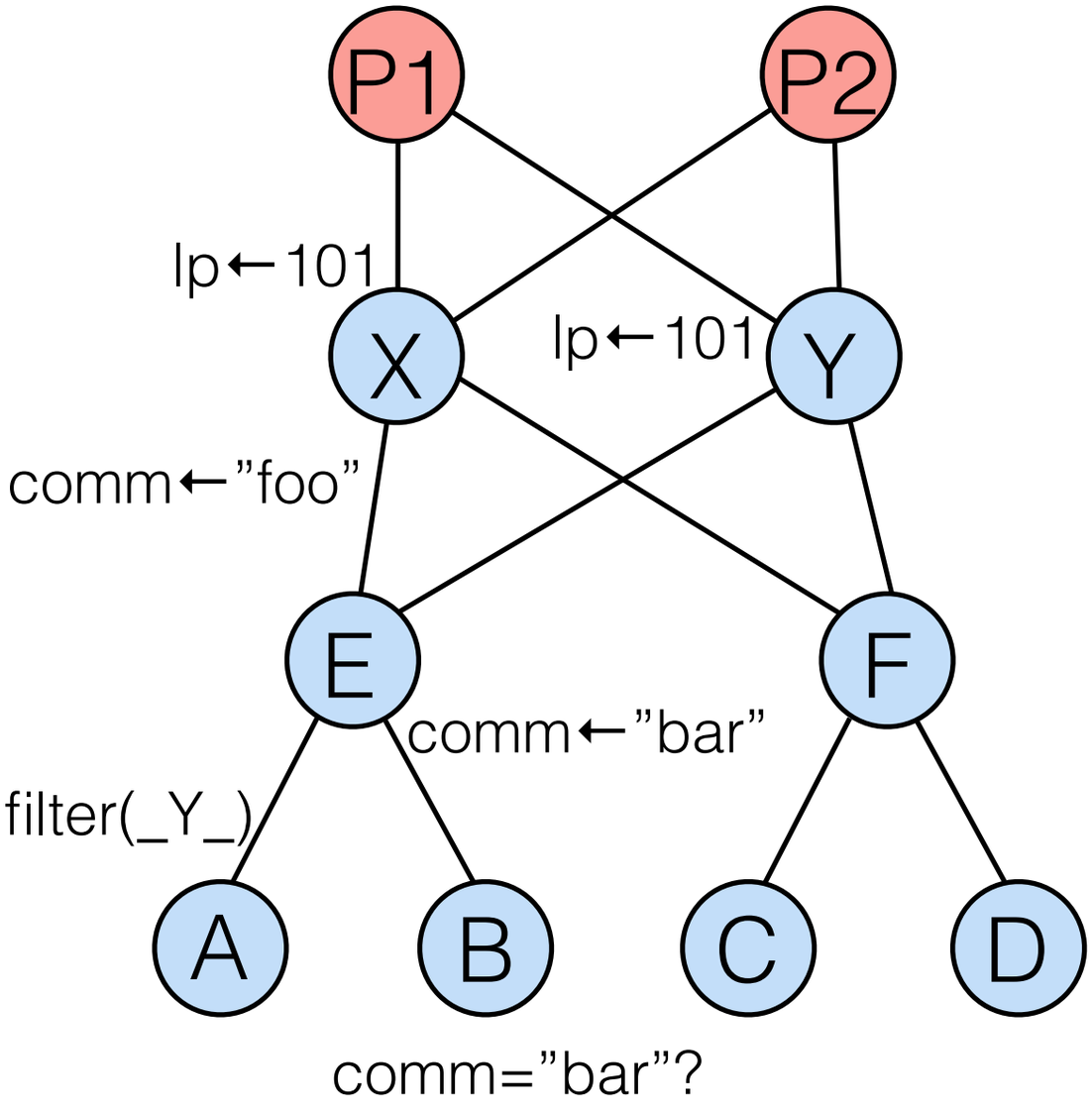
Other Topologies - DCell

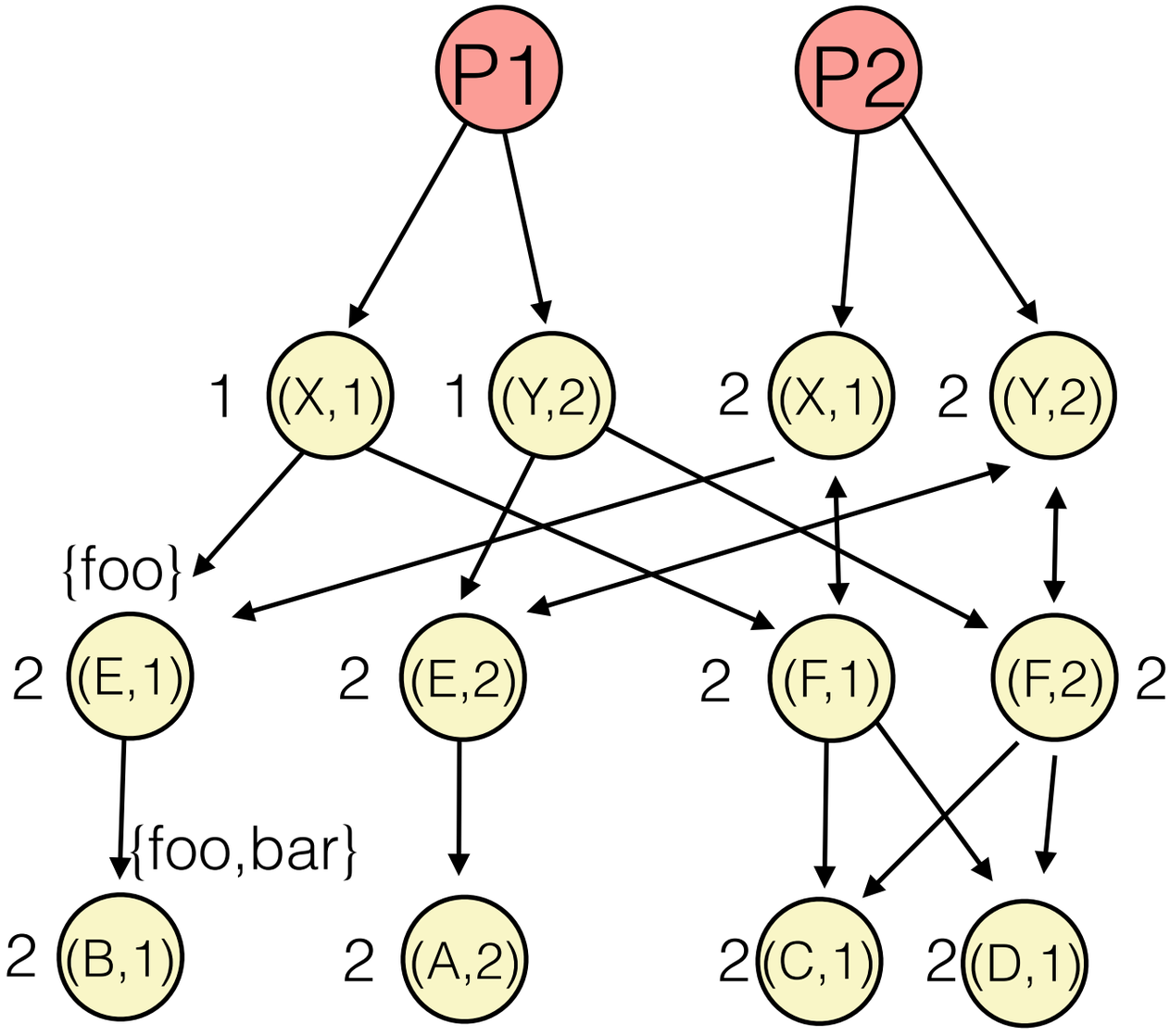


No **useful** abstraction possible?

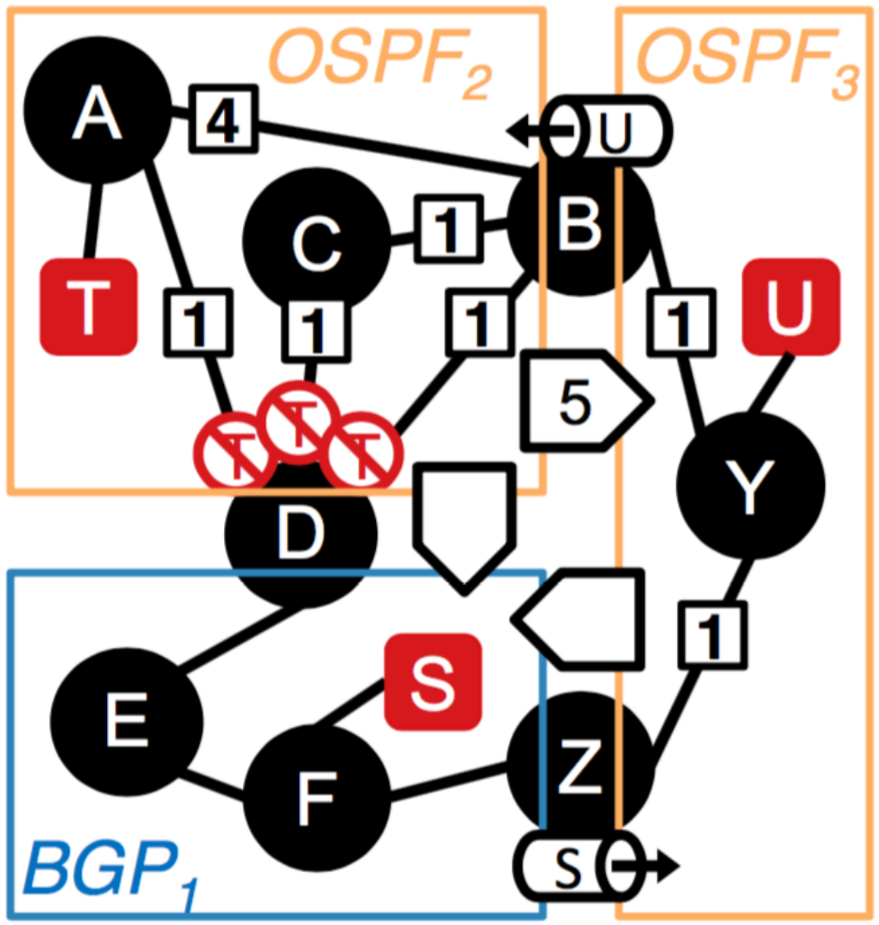
Config Verification (Thoughts)

Verification

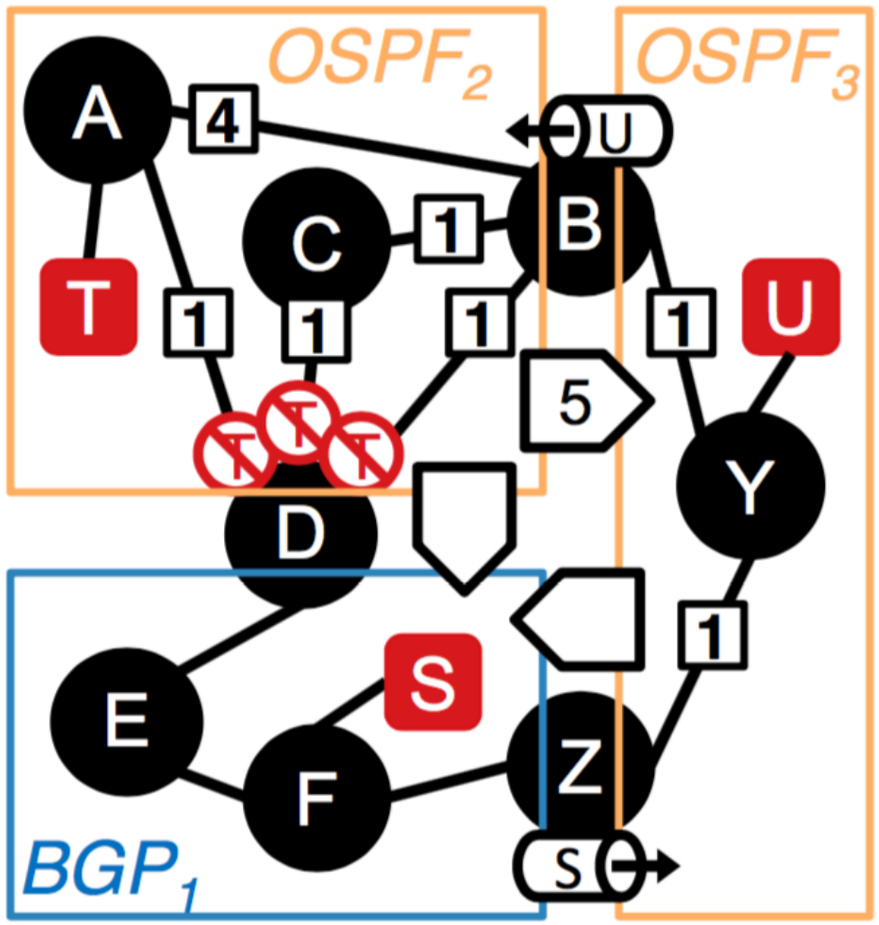




OSPF & Route Redistribution

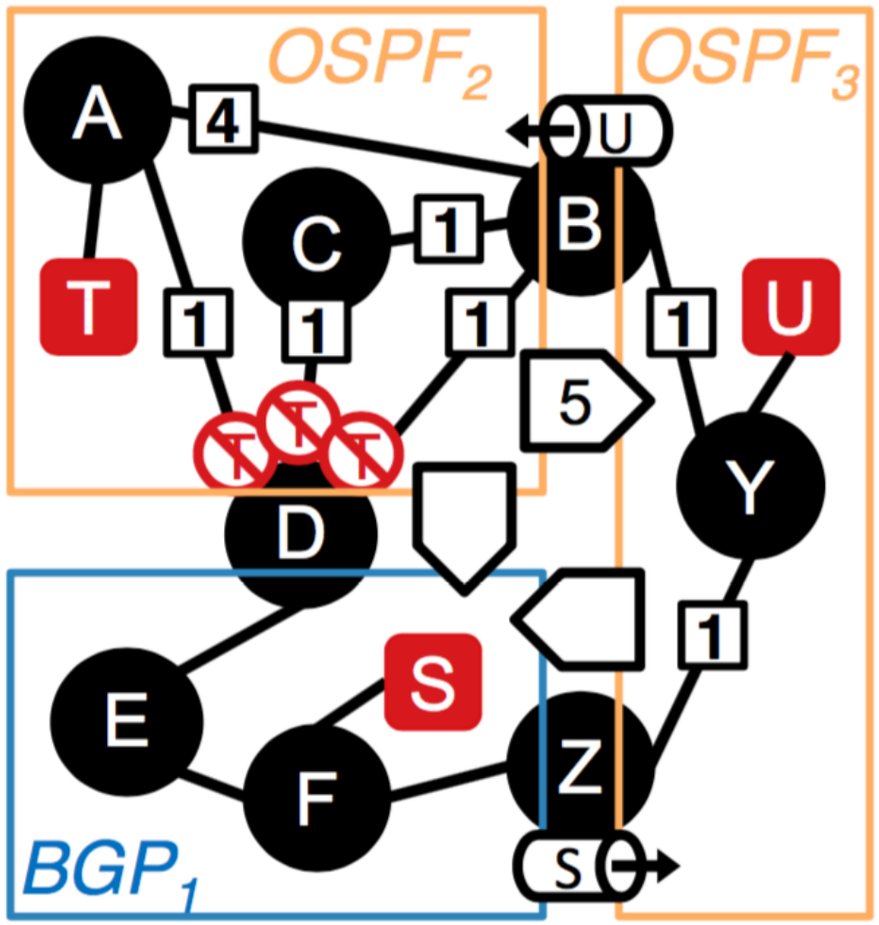


OSPF & Route Redistribution



Equivalence Classes:
Traffic classes that will experience the exact same forwarding behavior after the control plane stabilizes

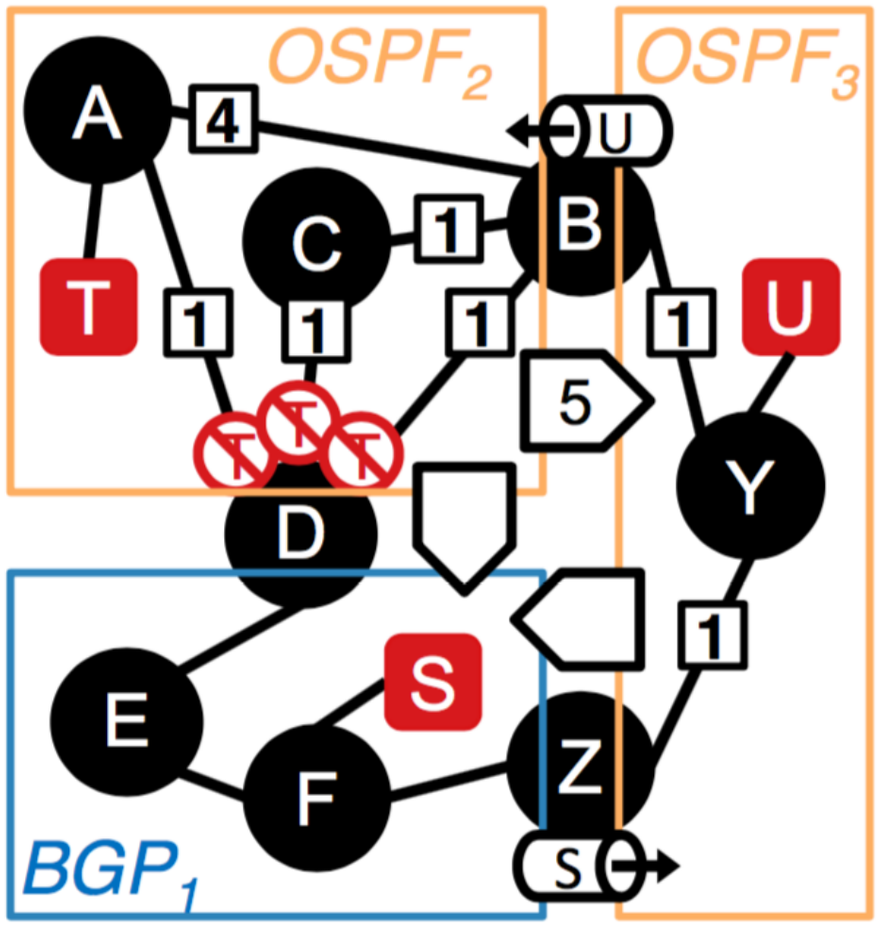
OSPF & Route Redistribution



Equivalence Classes:
Traffic classes that will experience the exact same forwarding behavior after the control plane stabilizes

EC: {T}, {S}, {U}

OSPF & Route Redistribution



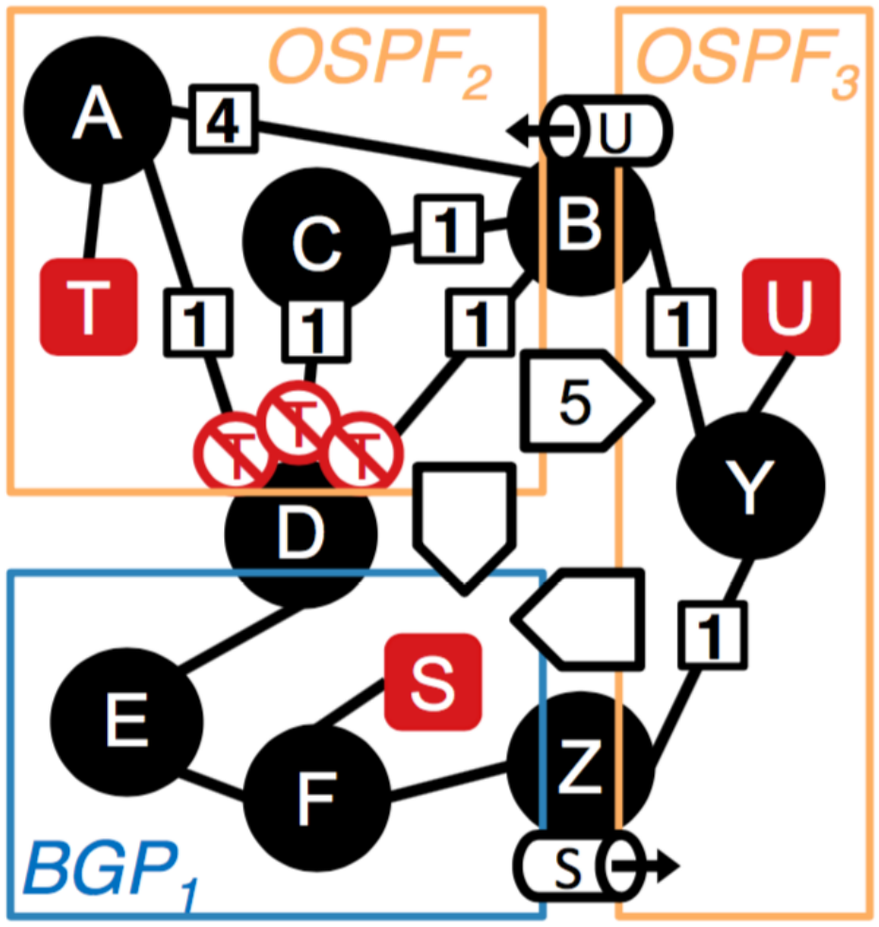
EC: {T}, {S}, {U}

Equivalence Classes:
Traffic classes that will experience the exact same forwarding behavior after the control plane stabilizes

High-level Idea
PG lets us represent a local preference among neighbors. We wish to prefer based on:

(1) Protocol (AD)
(2) Protocol-specific preference

OSPF & Route Redistribution



EC: {T}, {S}, {U}

AD: {**1**↦1, **5**↦2, **20**↦3, **110**↦4}

Static User eBGP OSPF

BGP (lp): {100 ↦ 1}

OSPF: {_ ↦ 1}

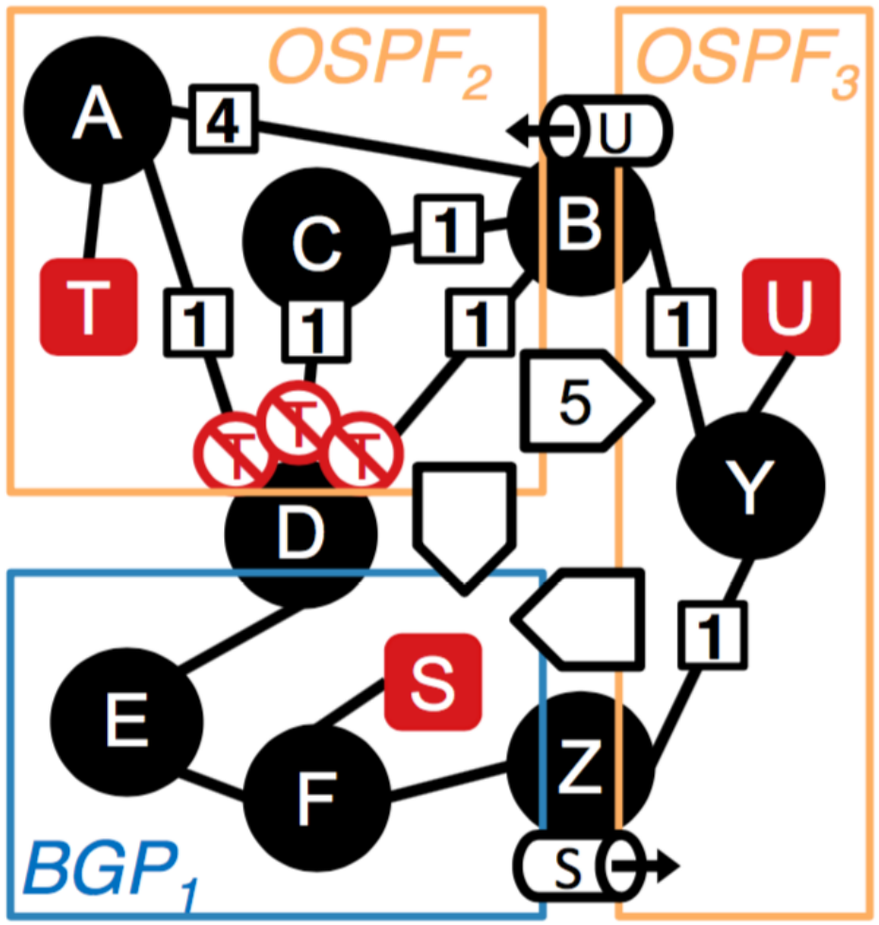
Preference: (AD x _)

Protocol specific

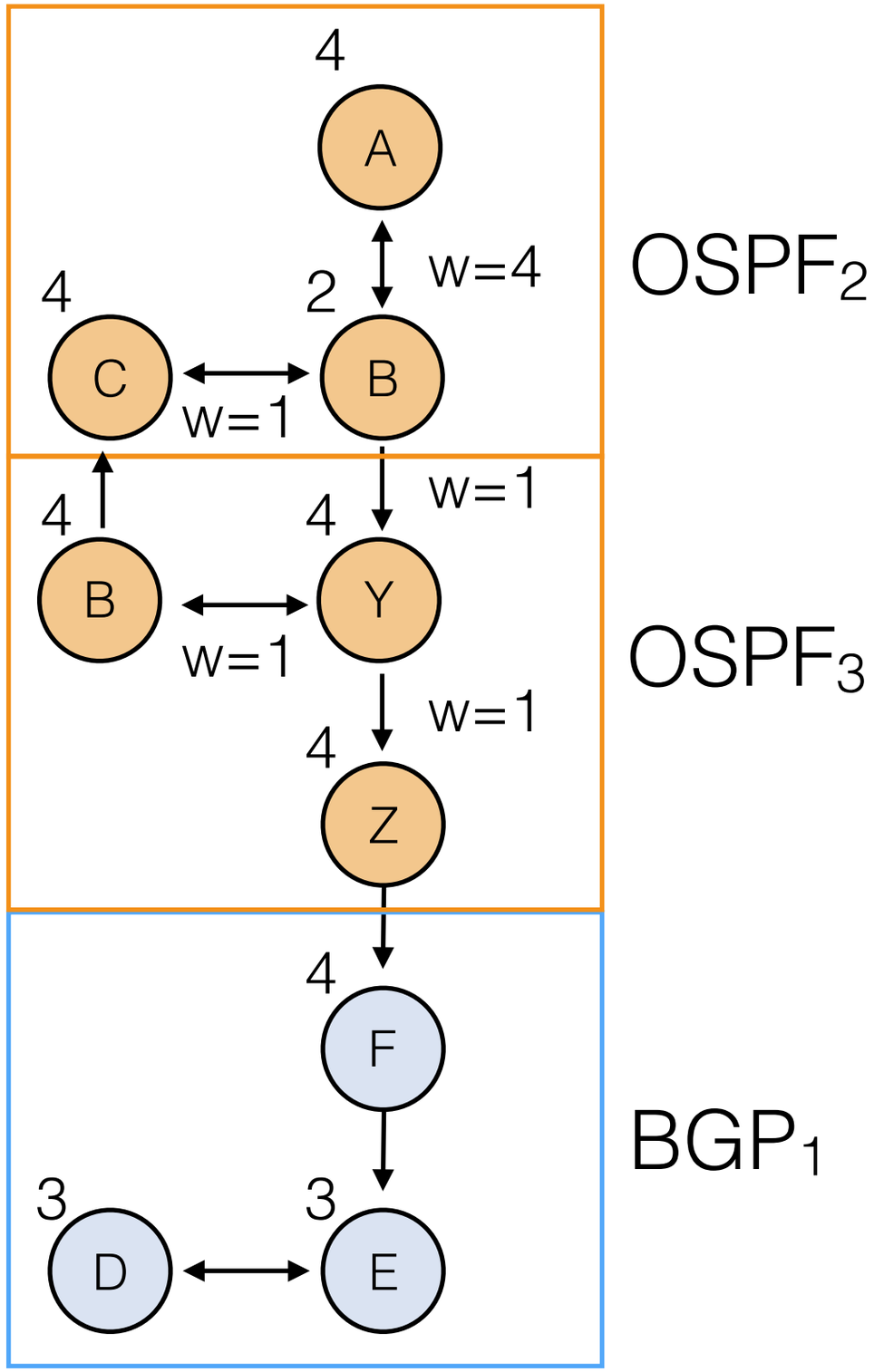
BGP: {1,2,3,4}

OSPF: {1,2,3,4}

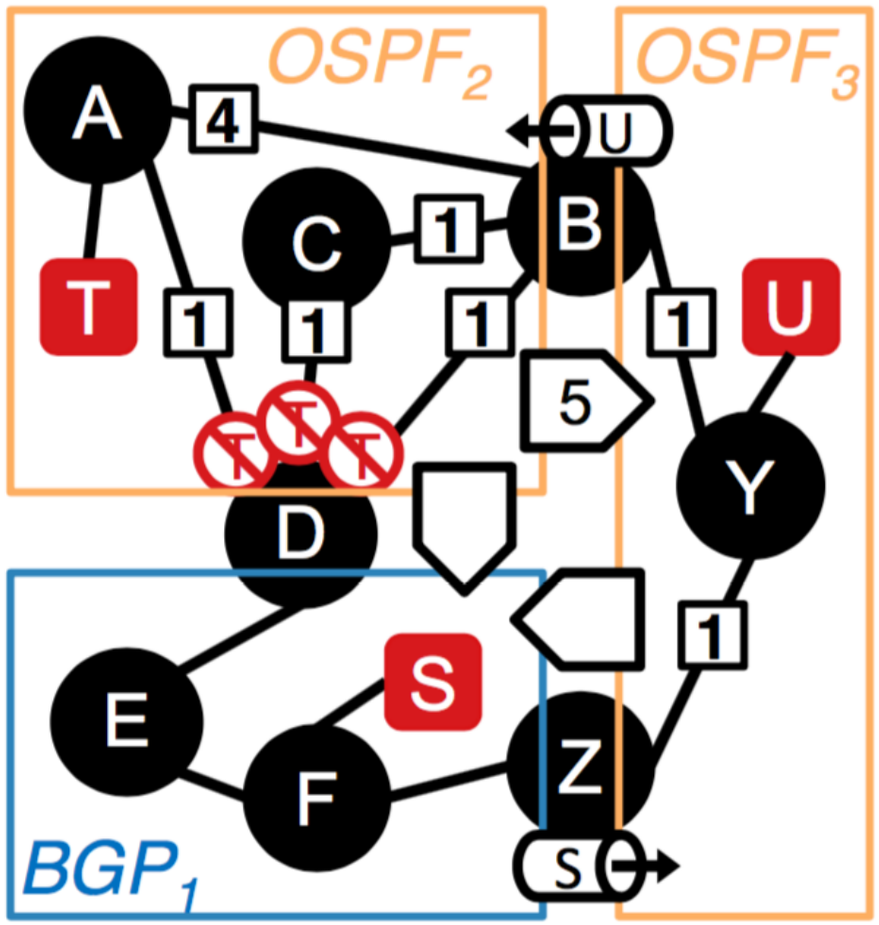
OSPF & Route Redistribution



EC: {T}, {S}, {U}

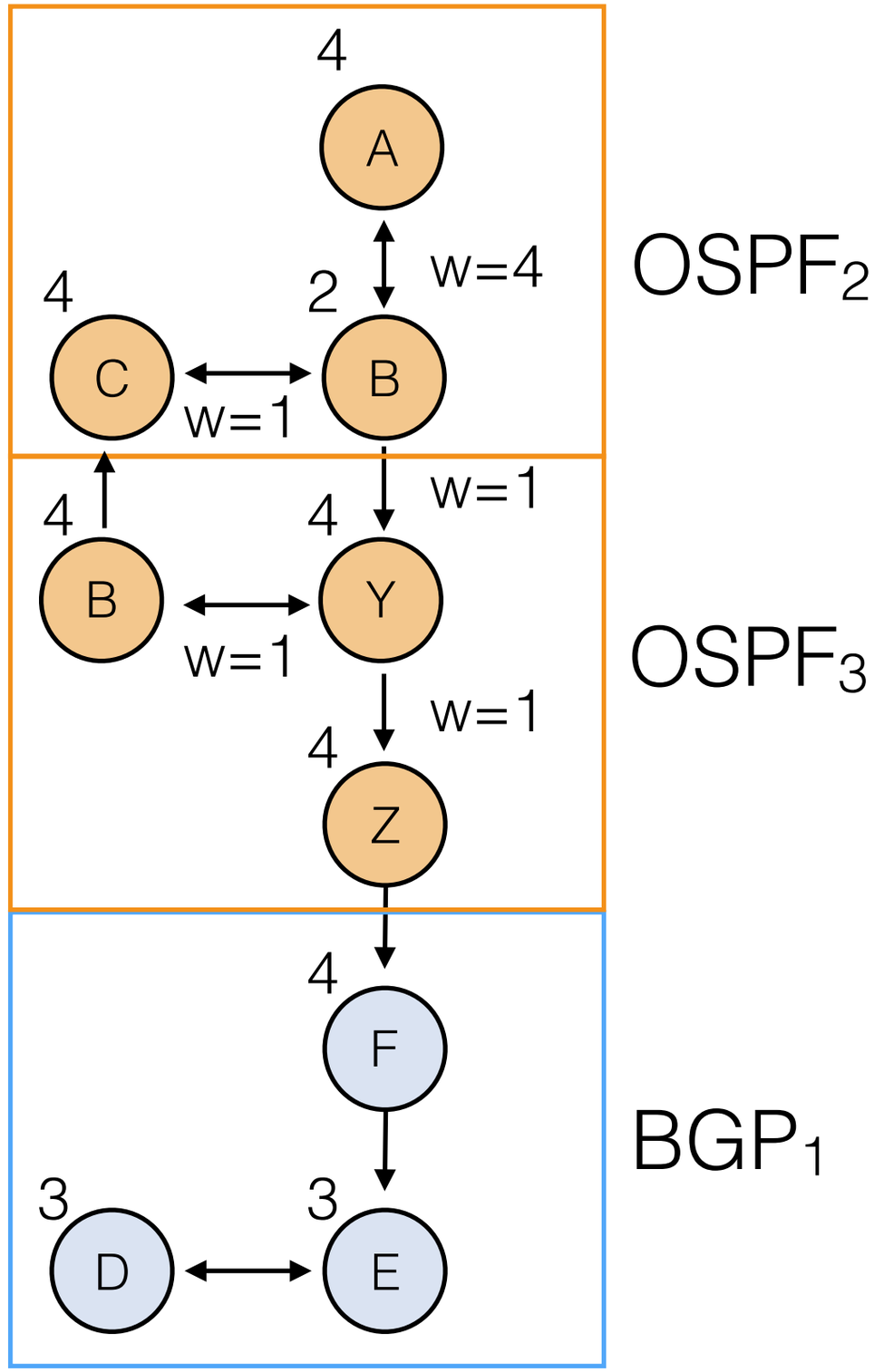


OSPF & Route Redistribution

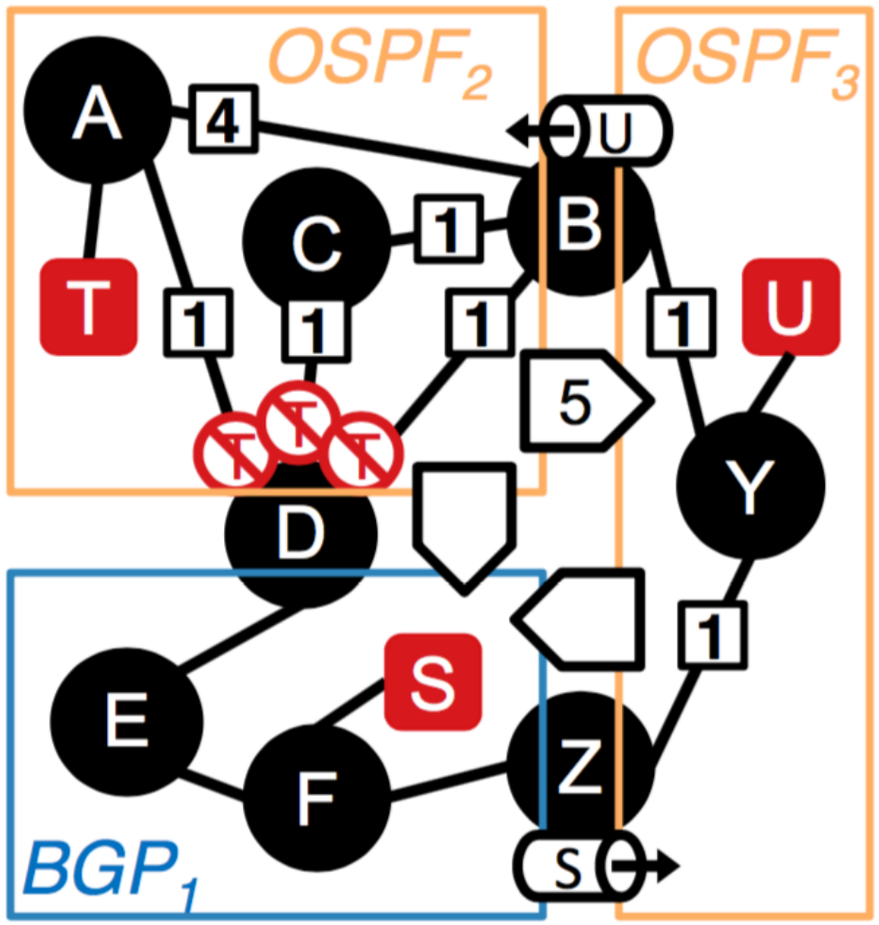


EC: {T}, {S}, {U}

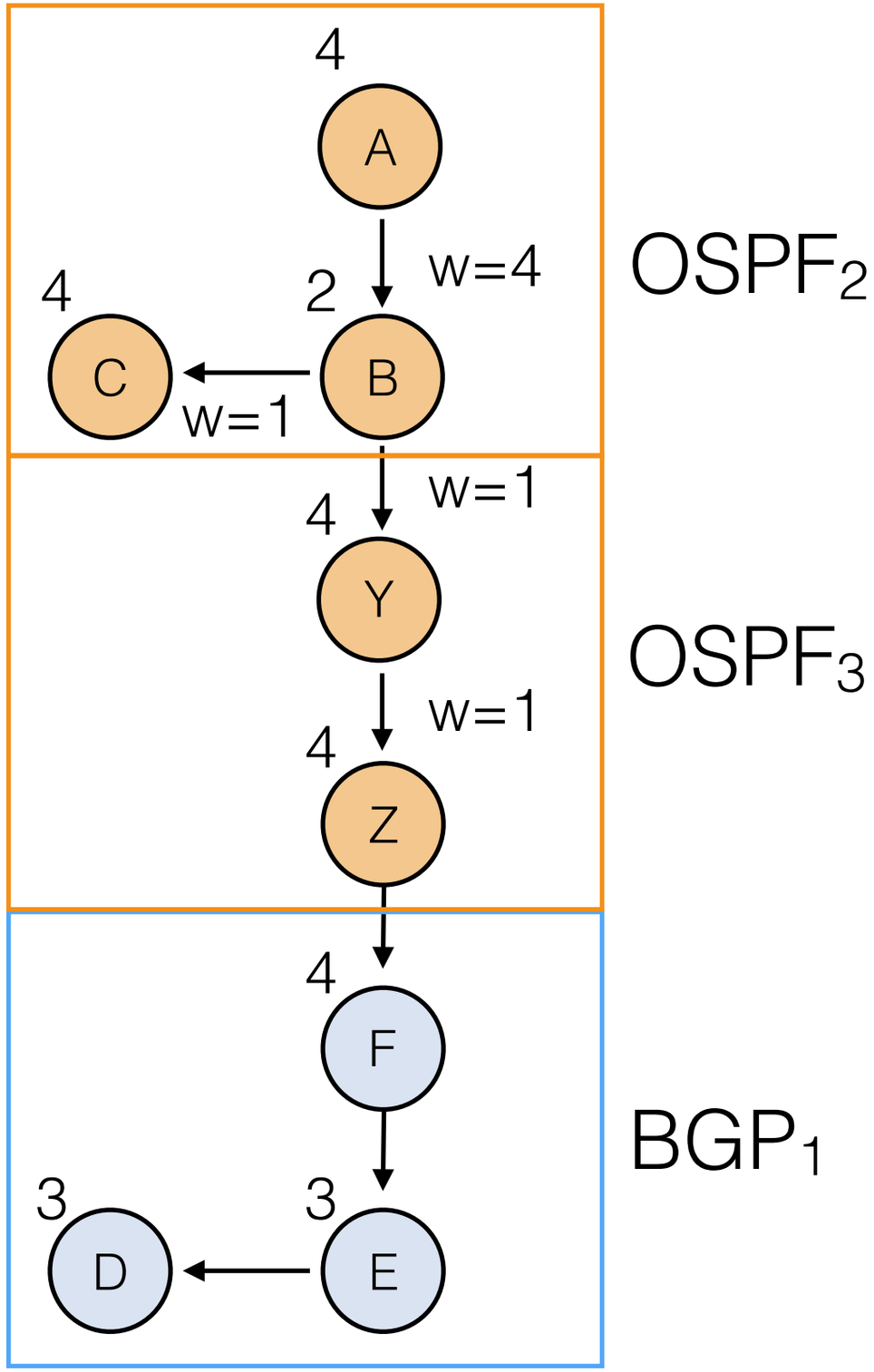
Dominated



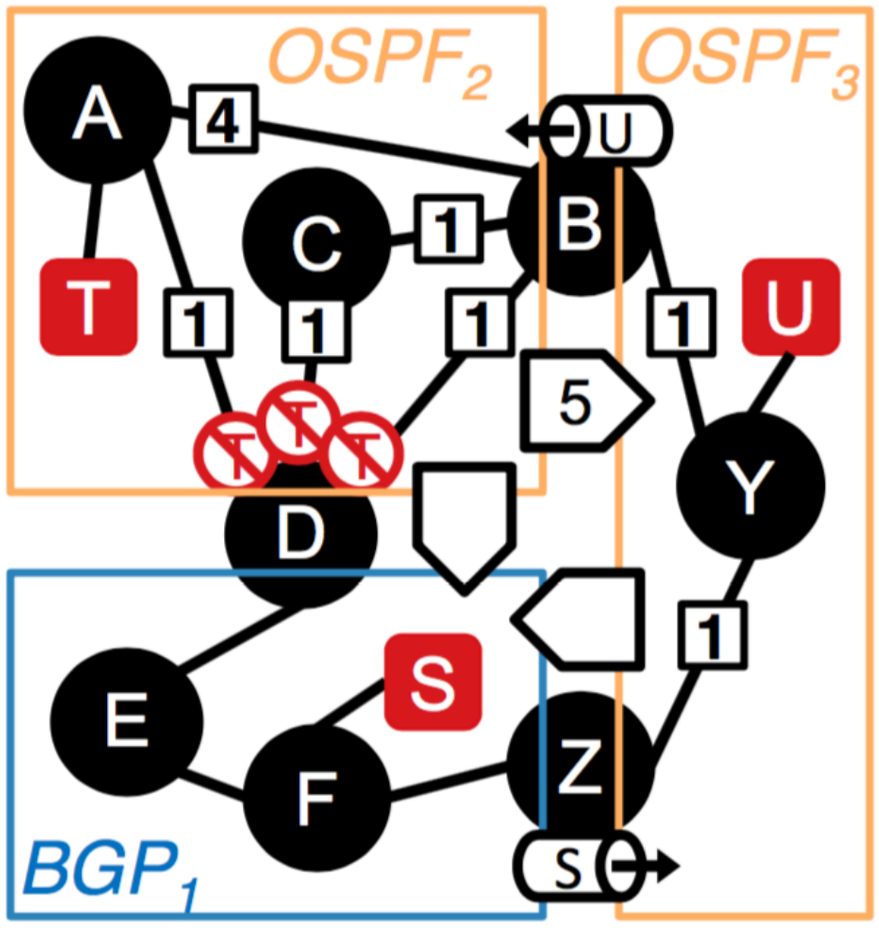
OSPF & Route Redistribution



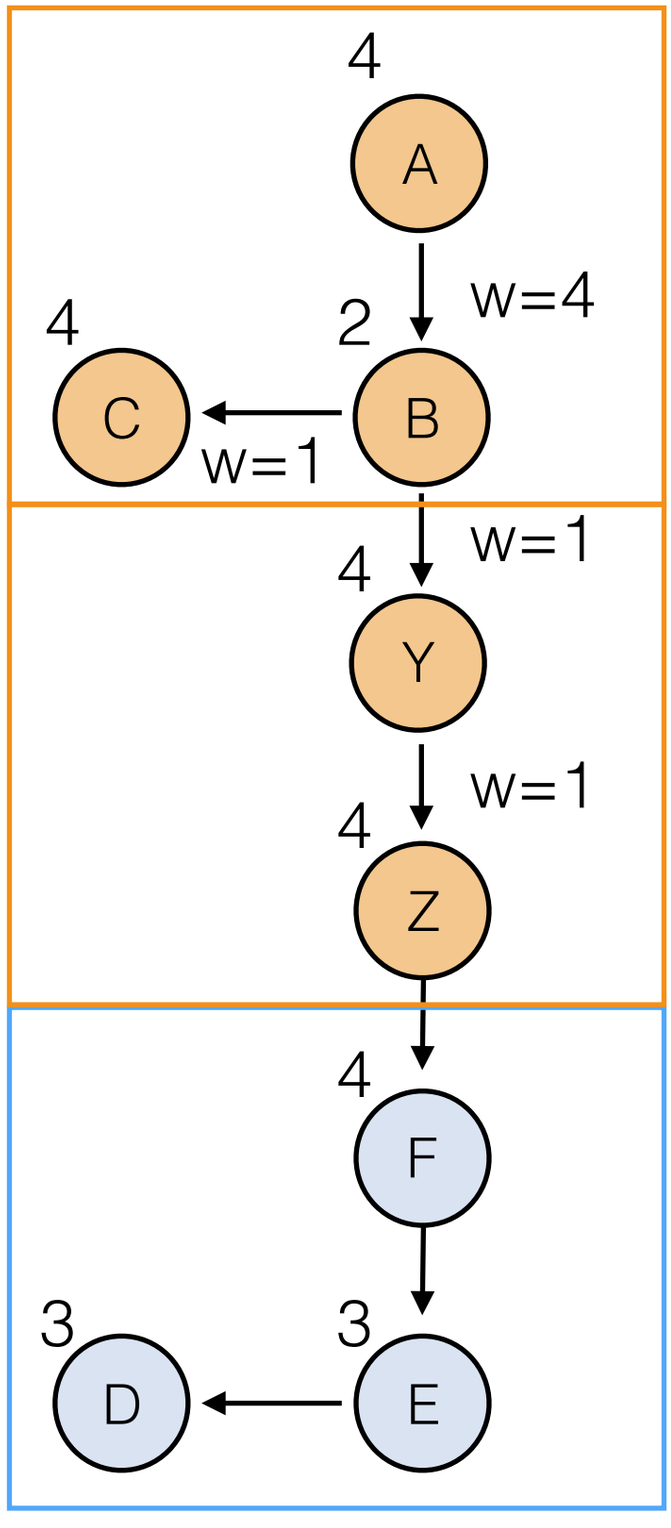
EC: {T}, {S}, {U}



OSPF & Route Redistribution

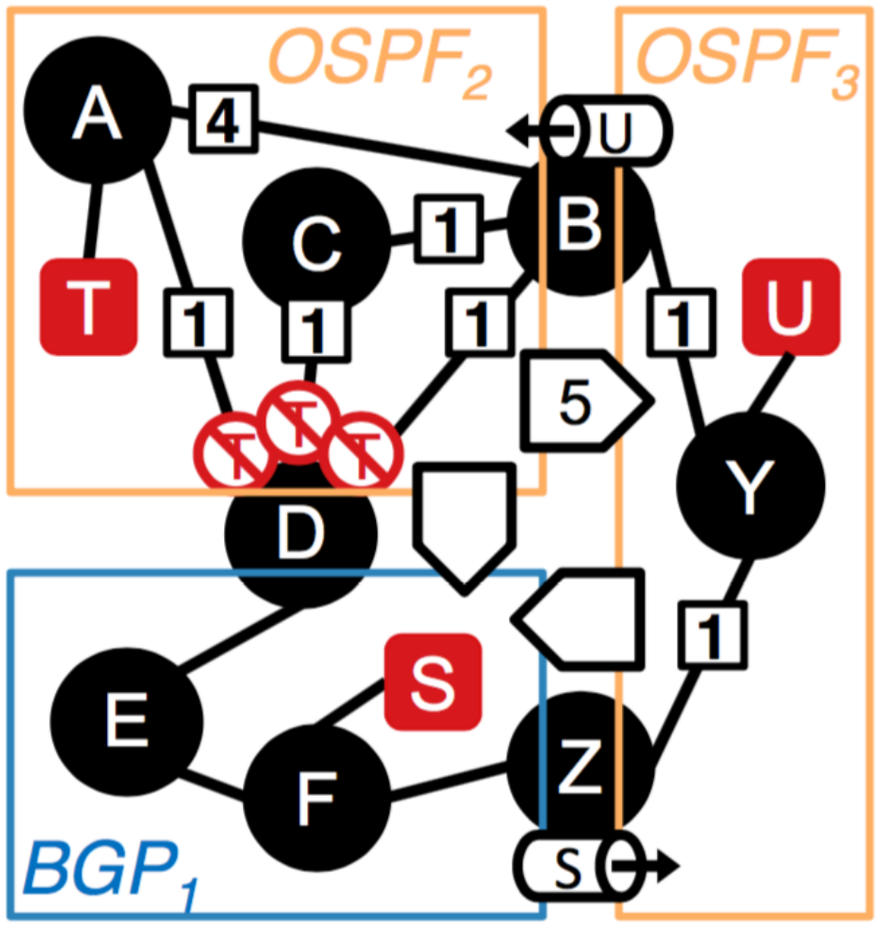


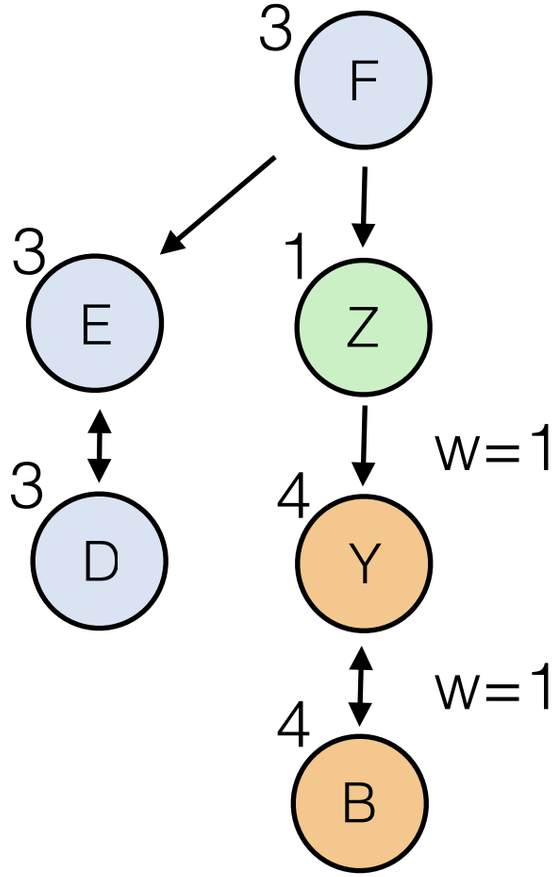
EC: {T}, {S}, {U}



No duplicate nodes means simple graph algorithms are enough!

OSPF & Route Redistribution





EC: {T}, {S}, {U}

Routing Protocol	Feature	ARC	PG
OSPF	Single area	Yes	Yes
	Standard area	No	Yes
	Stubby area	No	Yes
	Totally stubby area	No	Yes
	Not so stubby area	No	?
RIP		Yes	Yes
BGP	eBGP	Yes	Yes
	local-pref	No	Yes
	regex filters	No	Yes
	community tags	No	Yes
	iBGP	No	Yes
Static routes		Yes	Yes
ACLs		Yes	Yes
Route filters		Yes	Yes
Route redistribution	acyclic	Yes	Yes
	cyclic	No	Yes

Tradeoffs

Pros:

- Completeness, can handle (almost) all routing features
- Route redistribution and eBGP modeled the same way
- Often the simpler cases are computationally simpler

Cons:

- Standard graph algorithms don’t work due to dup labels
- Most algorithms are either NP-hard or conservative

Properties to Check:

Local Properties:

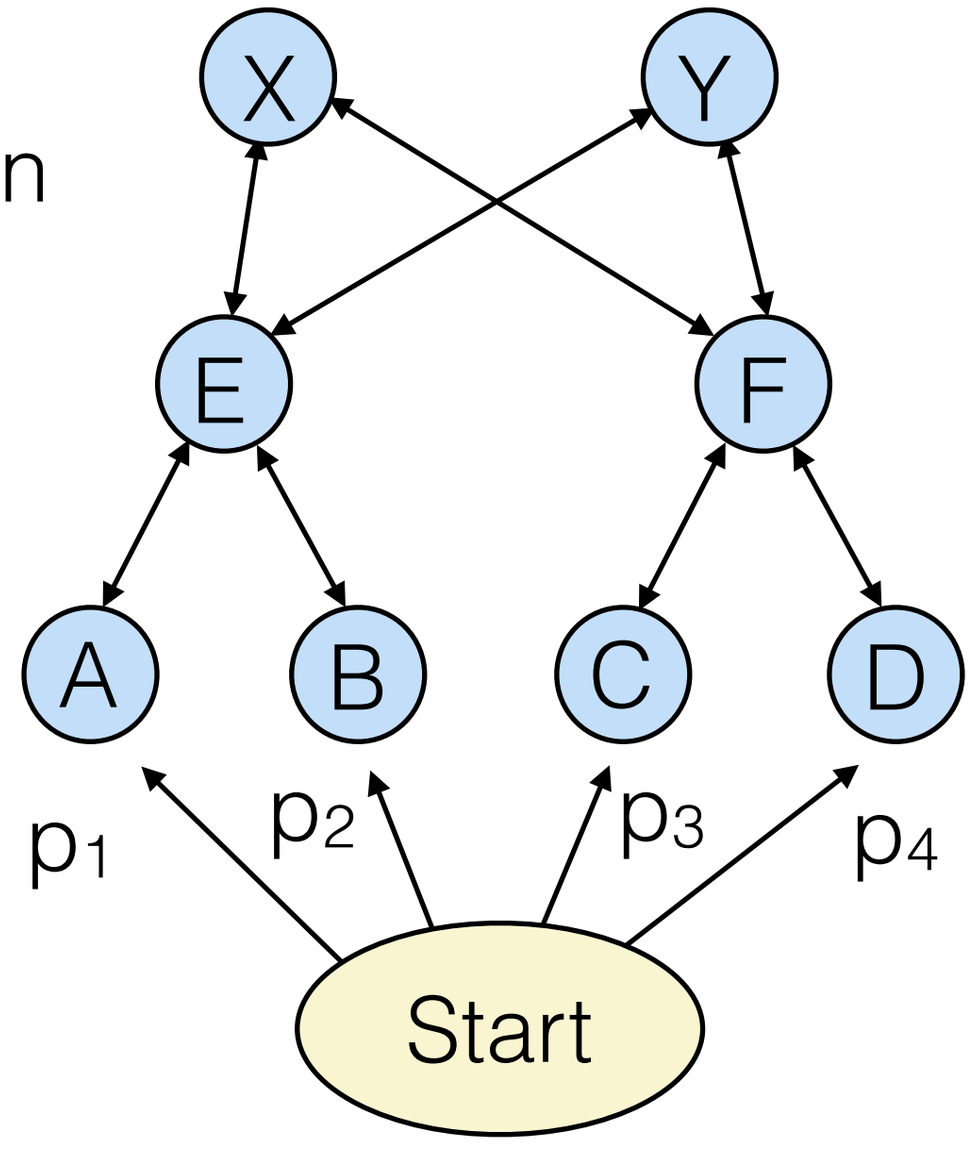
- Do all T2 routers locally prefer P1 over P2?
- Can dest. *d* ever be advertised from A to B?
- Can an eBGP adv. with comm *c* ever leak outside?

Path Properties:

- Is there a path for router A under all k failures?
- Is a valid path to the destination not being used?
- Does traffic for *d* always have a certain shape?
- Is path p1 always preferred to path p2?
- Will traffic for *d* ever take a path longer than 10?
- Policy equivalence (paths equivalent under all failures)

Equivalence Classes

If the only difference is
where traffic originates,
then we can sacrifice precision
for big memory/time savings



Equivalence Classes

If the only difference is
where traffic originates,
then we can sacrifice precision
for big memory/time savings

E.g., Does **F** ever advertise
any route to **Y**

Look for path from **start** to **F**