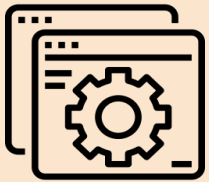
Janus: Risk based Planning of Network Changes in Data Centers

Omid Alipourfard, **Jiaqi Gao**, Jeremie Koenig,
Chris Harshaw, Amin Vahdat, Minlan Yu

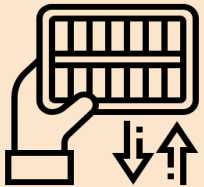
Yale HARVARD Google

Data centers are constantly evolving

- **Software changes:**
  - Bug fixes & feature releases



- **Hardware changes:**
  - Device repairs & upgrades

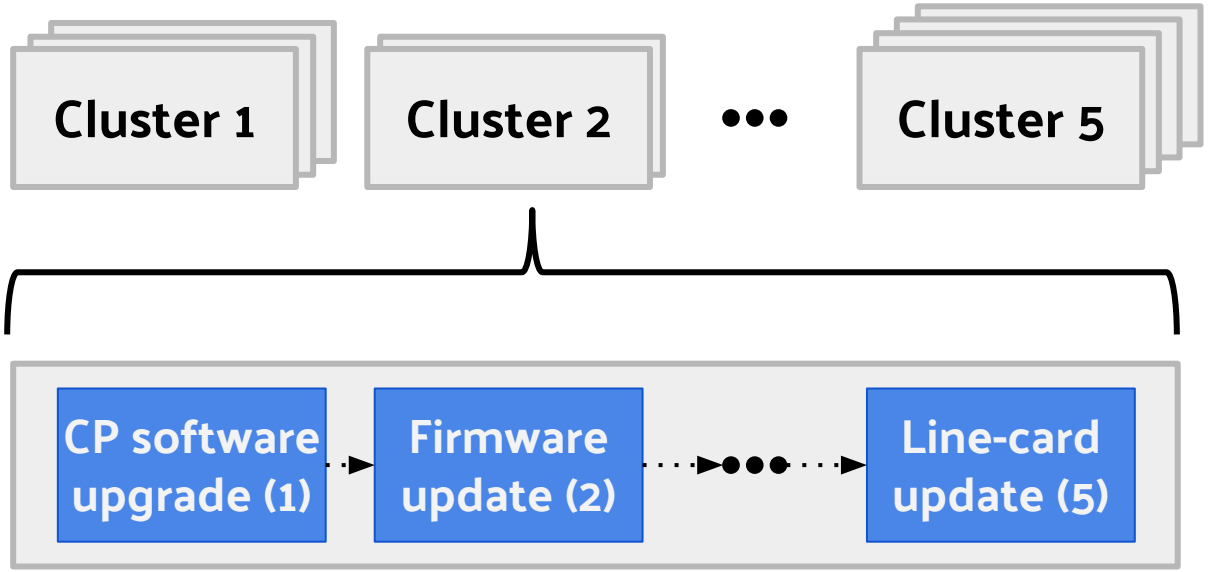


Changes are frequent:

- Apps need new features all the time.

- Exponential growth of traffic needs new design

Example: Transitioning to a Lossless Fabric



Deadline:
Finish all in 2 months!

Google:
58 changes per week

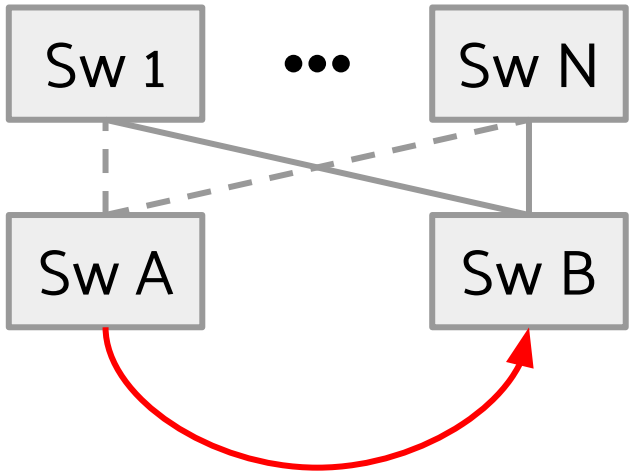
Bug fixes

Security patches

Other upgrades

Applying a change is risky.

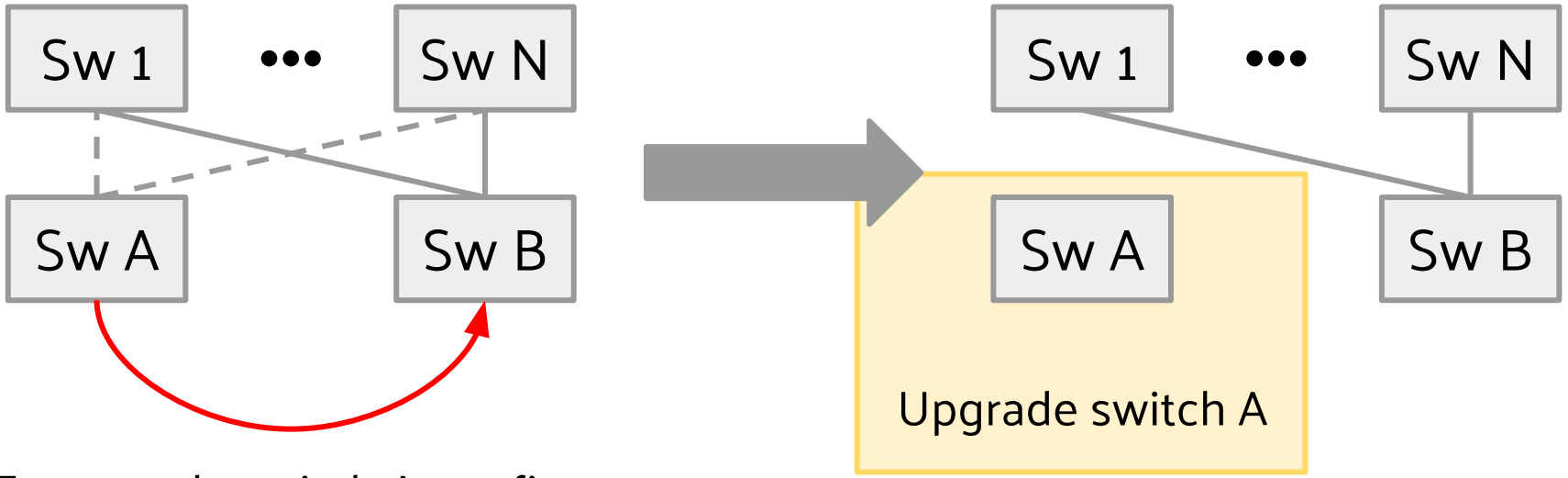
To apply a change, we (typically) drain the network devices.



To upgrade switch A, we first move traffic away from it

Applying a change is risky.

To apply a change, we (typically) drain the network devices.



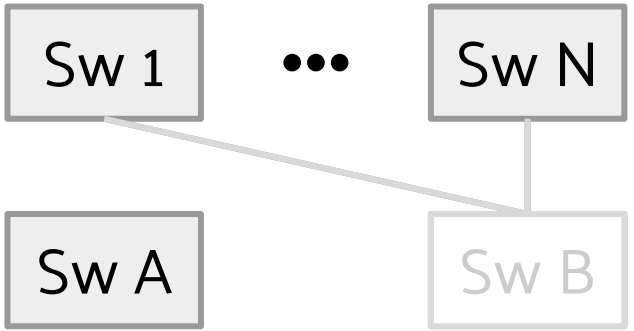
To upgrade switch A, we first move traffic away from it

Applying a change is risky.

Ongoing change can impact network customers.

Less capacity headroom
to absorb failures and traffic variations.

If Sw B fails
Sw 1, Sw2 and Sw N lose connectivity.



Even with warm reboots, we risk impacting customer traffic (e.g., buggy upgrades).

State-of-the-art: Operators follow rules of thumb

Today, Operators follow **rules of thumb.**

- Microsoft reserves 1/4th of capacity for changes (power-of-4)
- Google sets an upper bound for capacity reduction [SIGCOMM' 15]

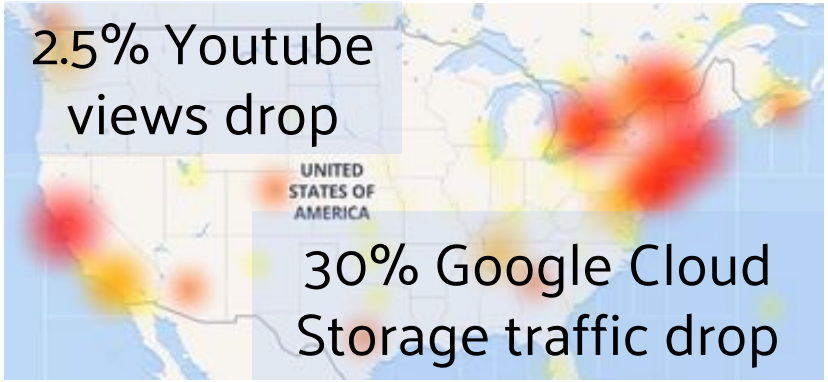
No estimation or control of **risks.**

- Slow upgrades: a long backlog of changes
- Fast upgrades: susceptible to impacting customers due to risks

Ignoring risks costs revenue and reputation

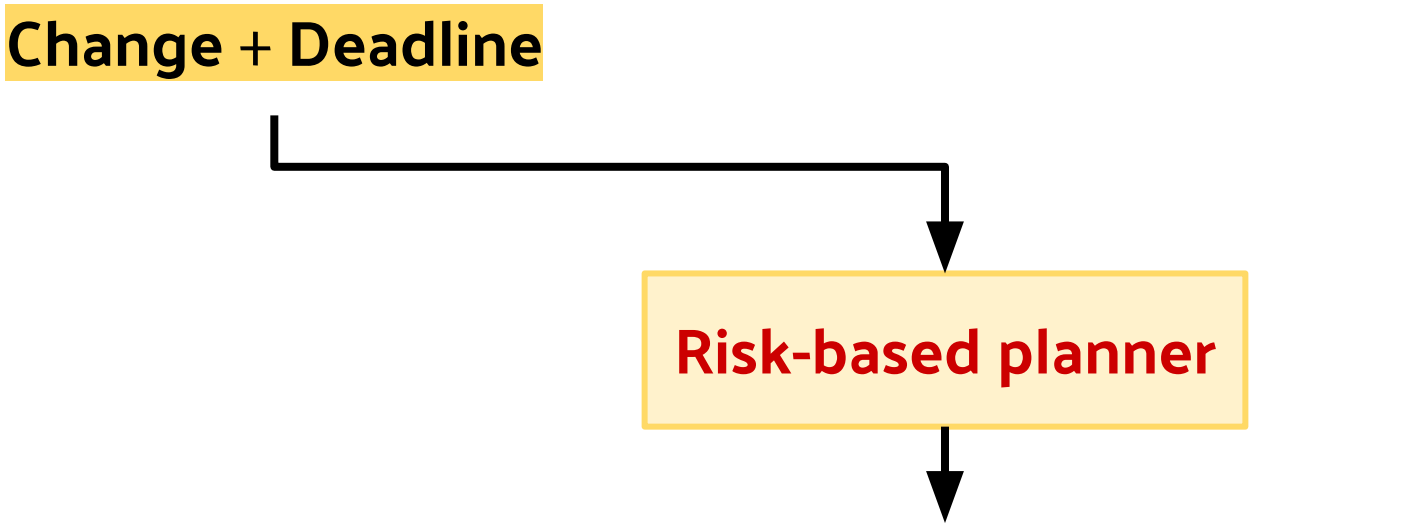
Service Level Agreement of Amazon compute service

Uptime	Refund
< 95%	**100%**
< 99%	30%
< 99.99%	10%

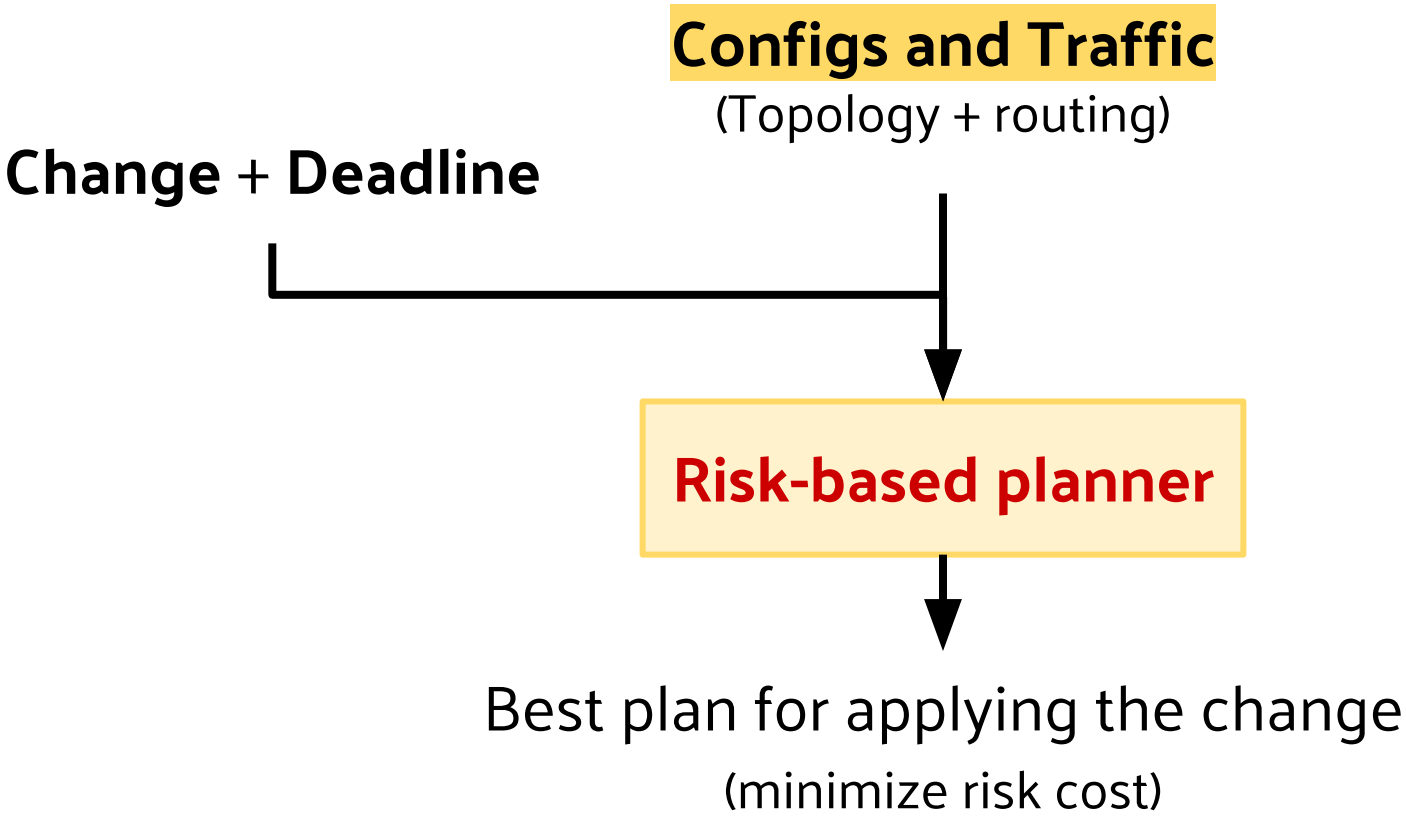


How to plan network changes while minimizing the cost of risks?

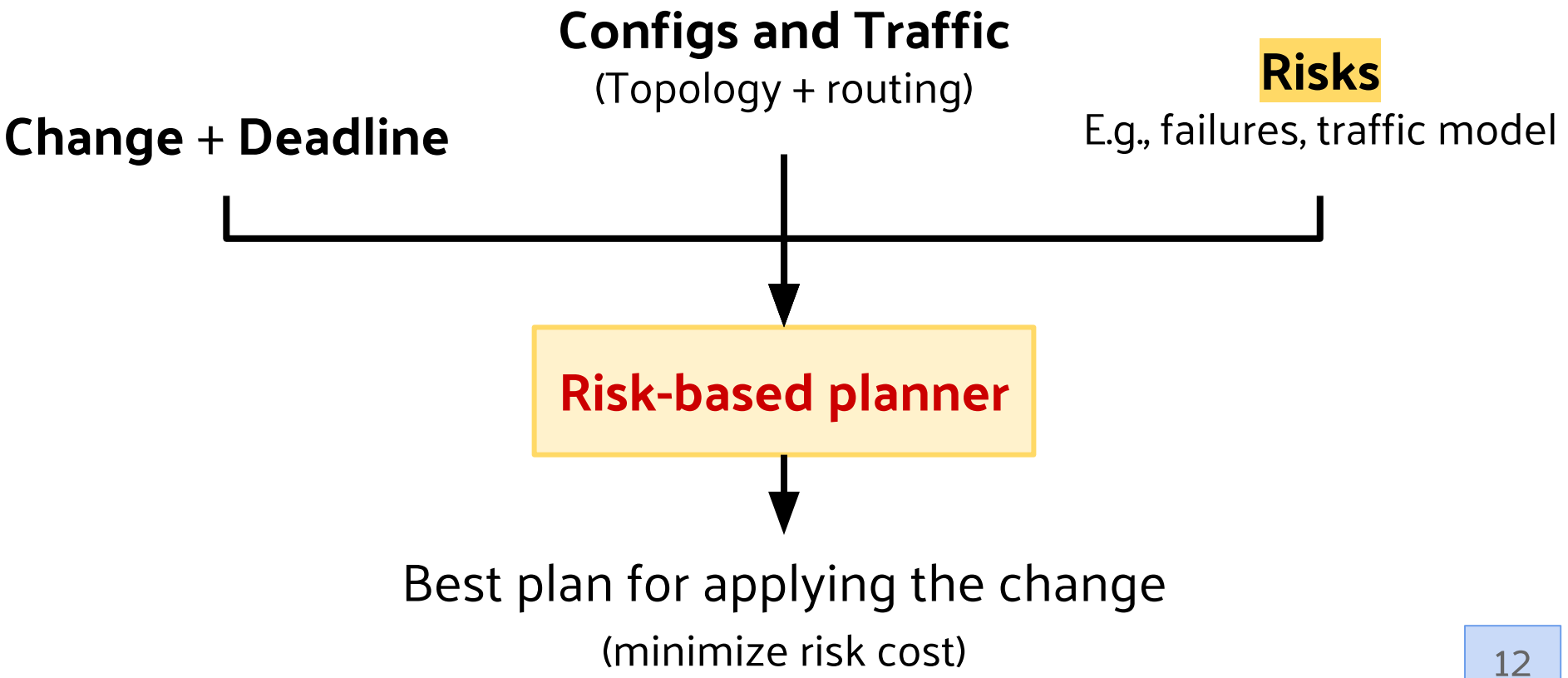
Risk-based planning



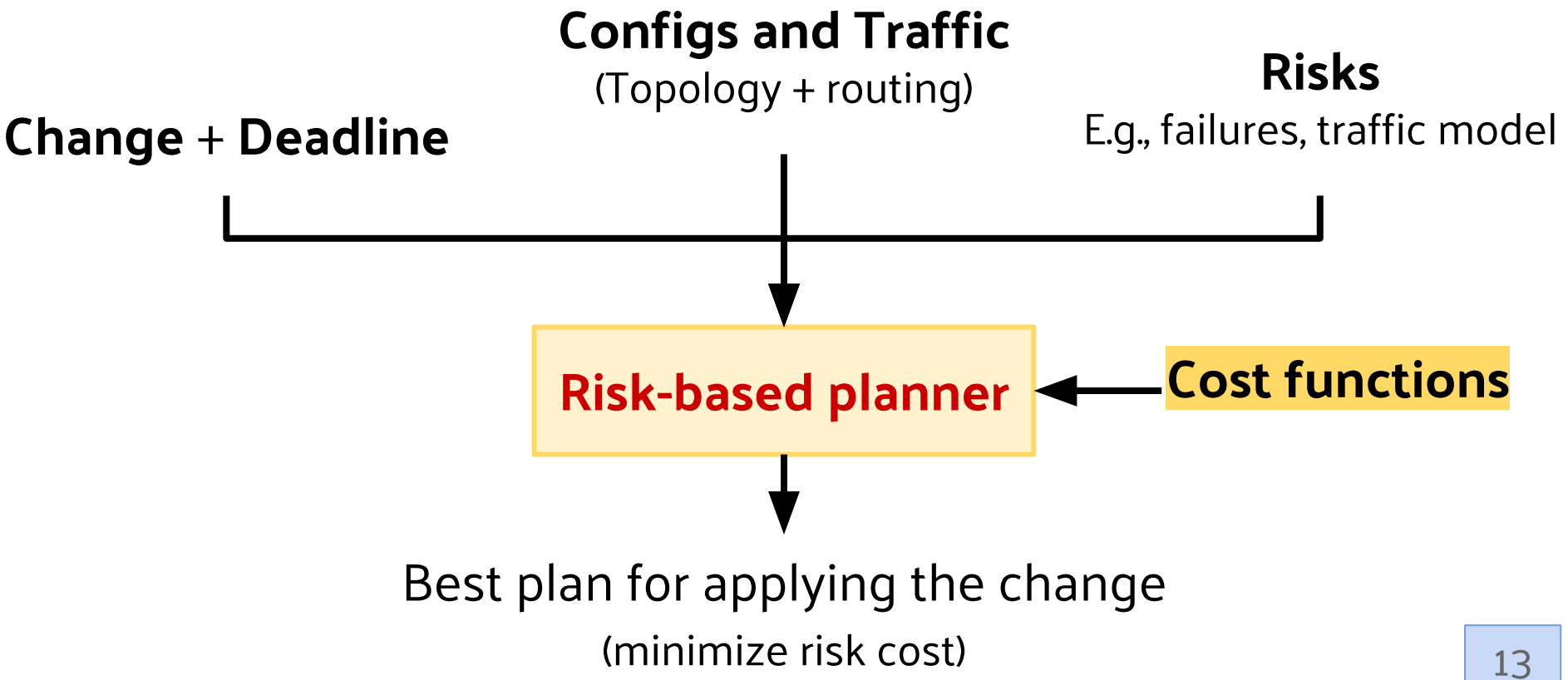
Risk-based planning



Risk-based planning



Risk-based planning



Planner should be adaptive

Change + Deadline

DC Configs and Traffic

Risks

Cost functions

10% vs 90% utilization

private vs public workloads

different failure patterns

We need to be adaptive to all the settings in different data centers

Planner should be adaptive: Modeling?

Change + Deadline

DC Configs and Traffic

Risks

Cost functions

~~**Modeling**~~

We need to be adaptive to all the settings in different data centers

Planner should be adaptive: Searching!

Change + Deadline

DC Configs and Traffic

Risks

Cost functions

~~**Modeling**~~

Searching independent of all settings

We need to be adaptive to all the settings in different data centers

Planner should be scalable

In how many ways, can you schedule a change with 3 operations?

Example: change-set = {A, B, C}: 13 possible plans

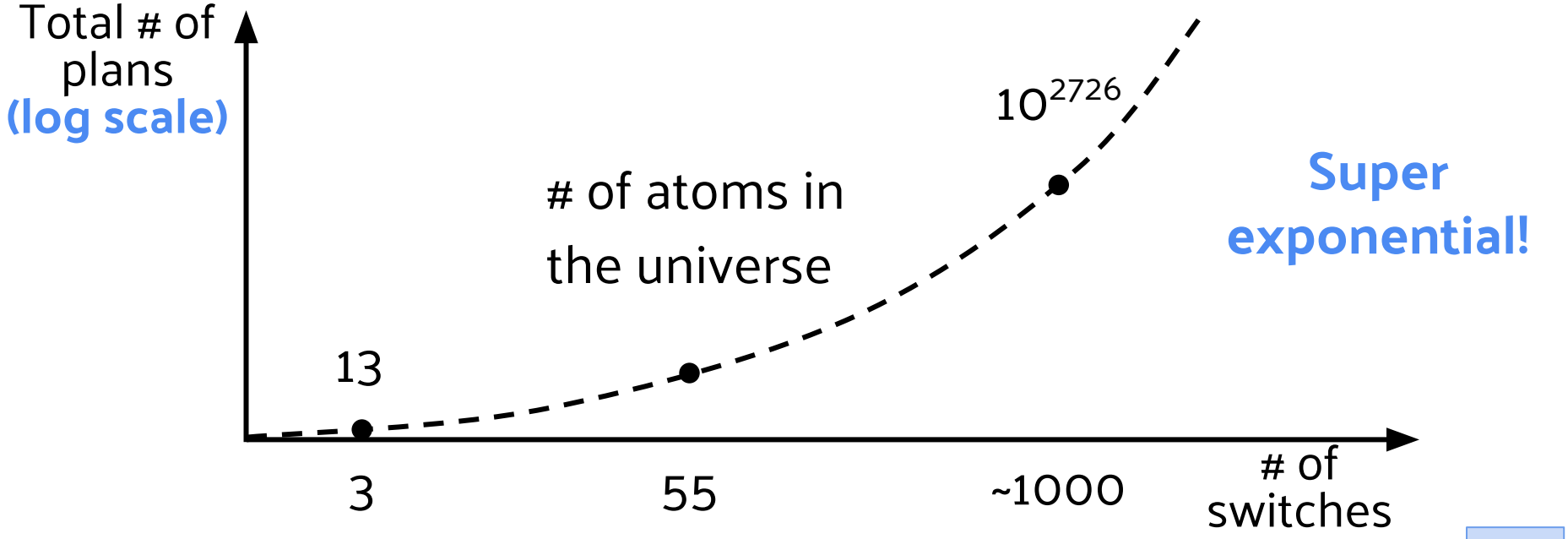
{A}, {B}, {C}	{A, B}, {C}	{A, B, C}
{B}, {C}, {A}	{B}, {A, C}	
…	…	

subplan

# of plans for a change is the # of ordered partitions

Planner should be scalable in super exponential space

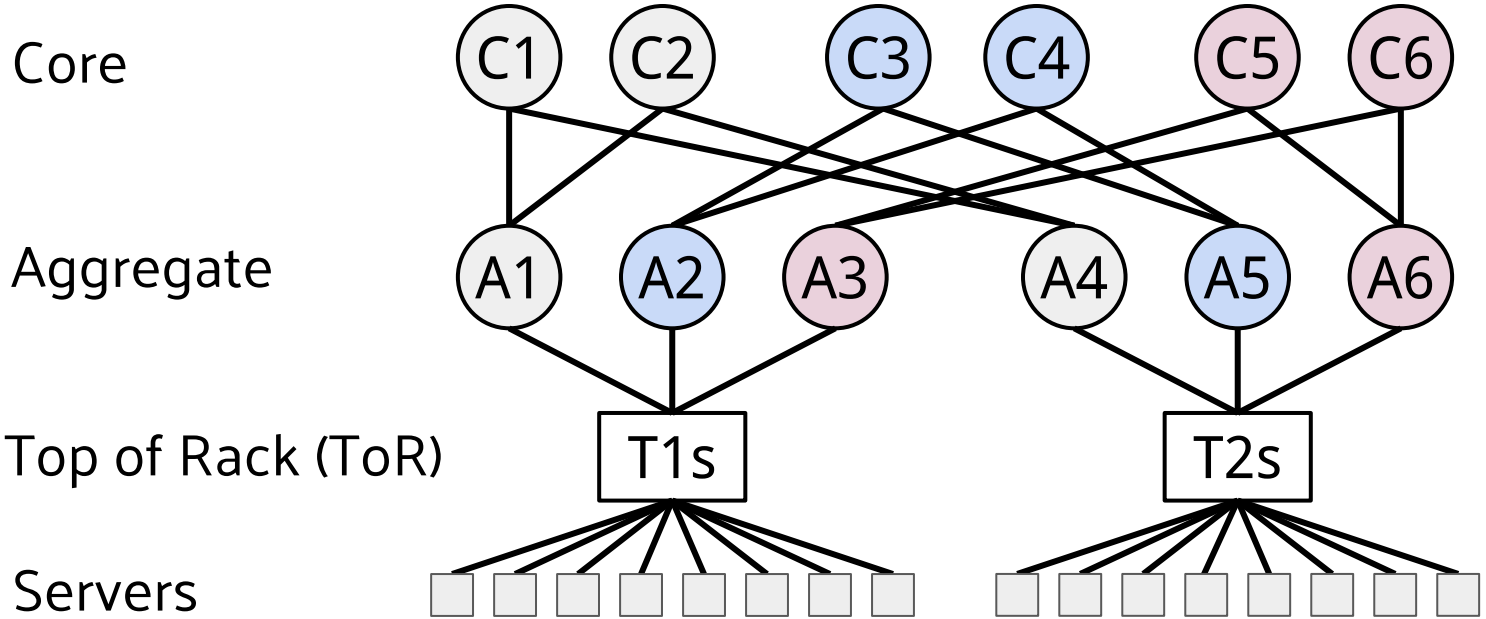
of possible plans for a change is the # of ordered partitions !!!



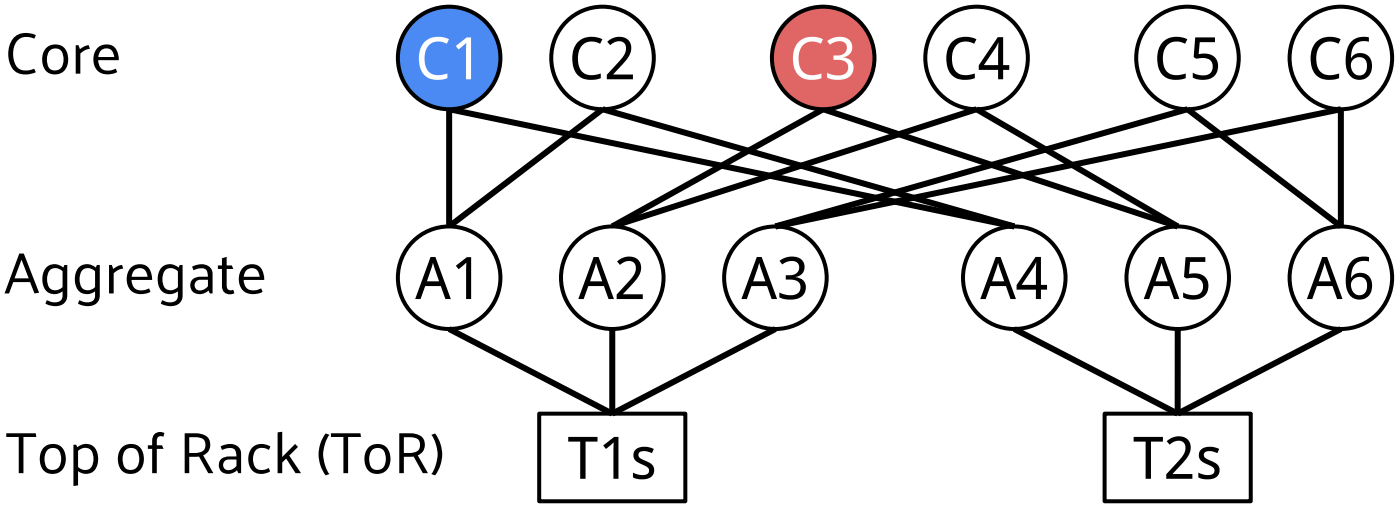
Leverage properties in large-scale DC networks

High degree of symmetry

Many path choices

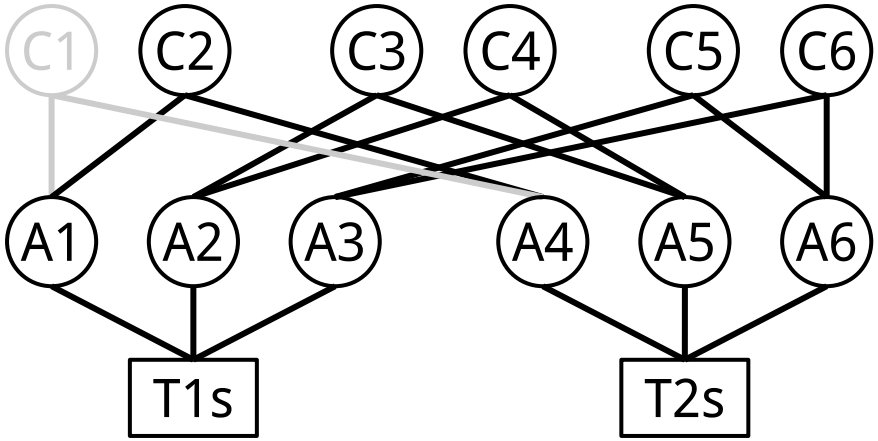


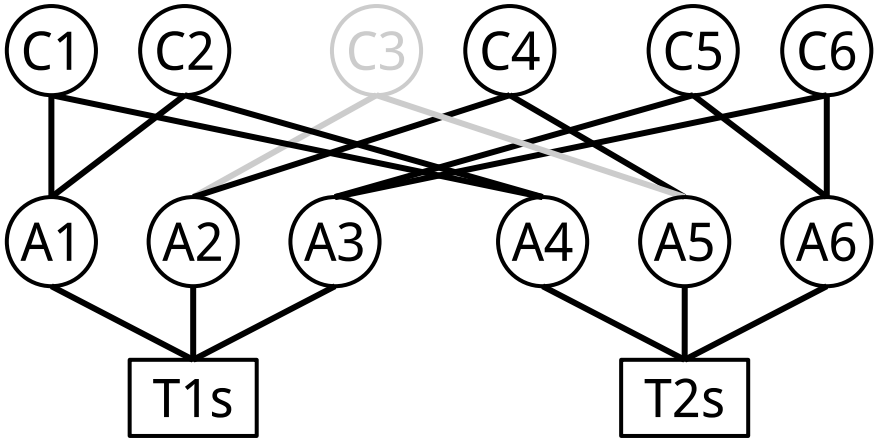
Leverage symmetry to find equivalent subplans



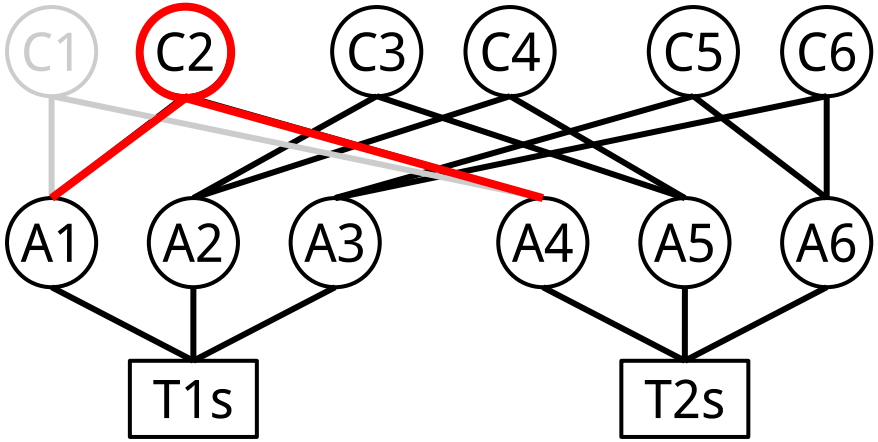
Two subplans are equivalent (with same cost) when they have equivalent traffic matrices, topology and routing

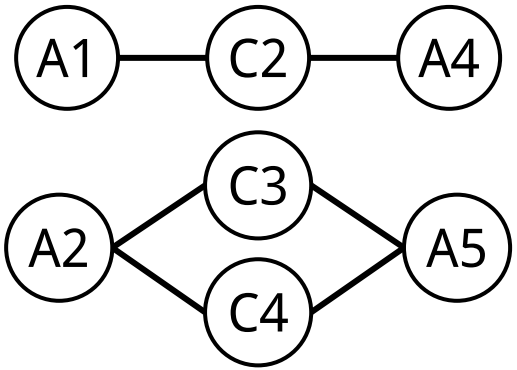
(1) Equivalent traffic matrices

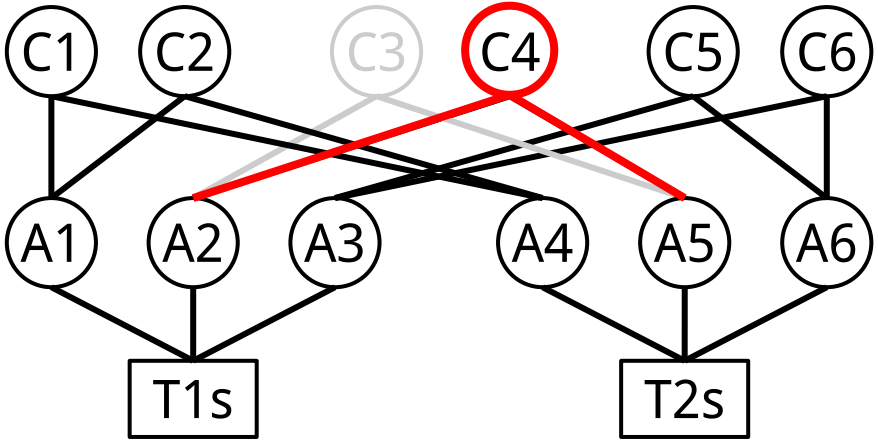


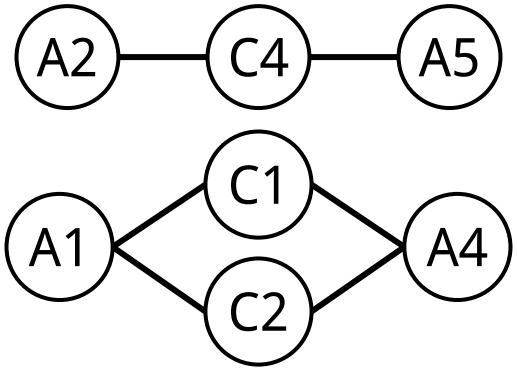


(2) Equivalent topology

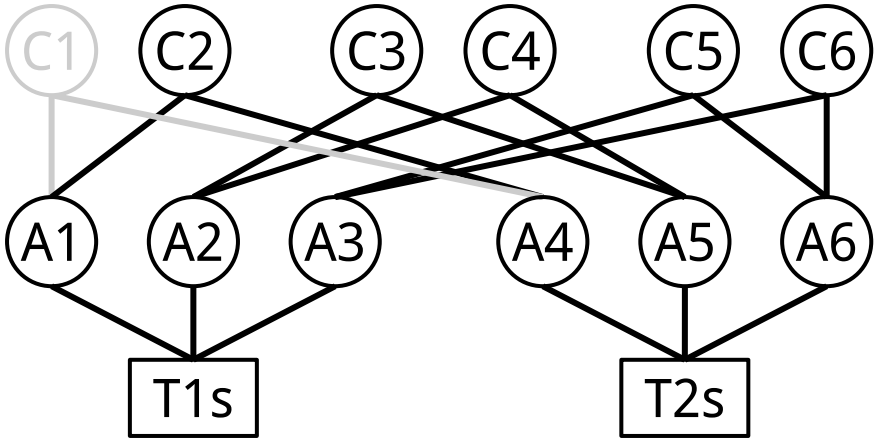


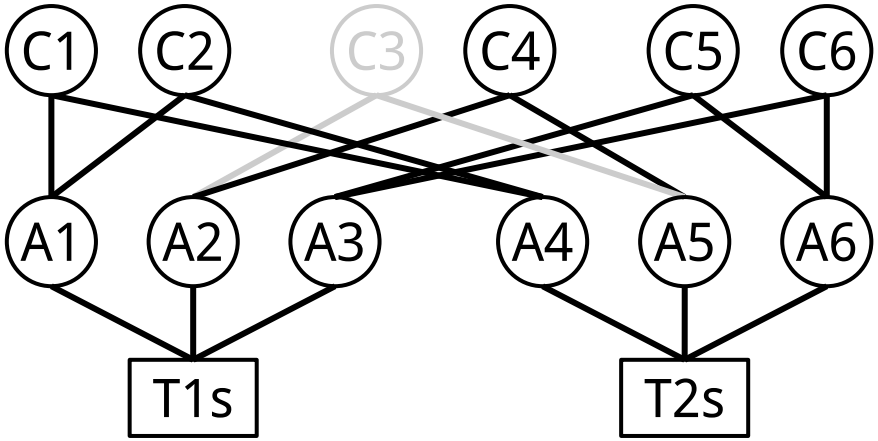


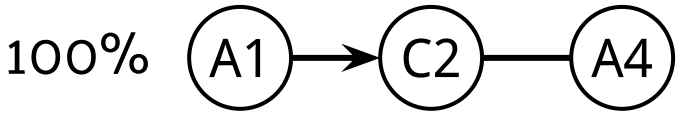


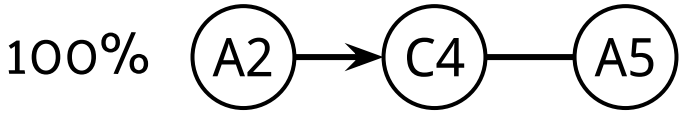


(3) Equivalent routing





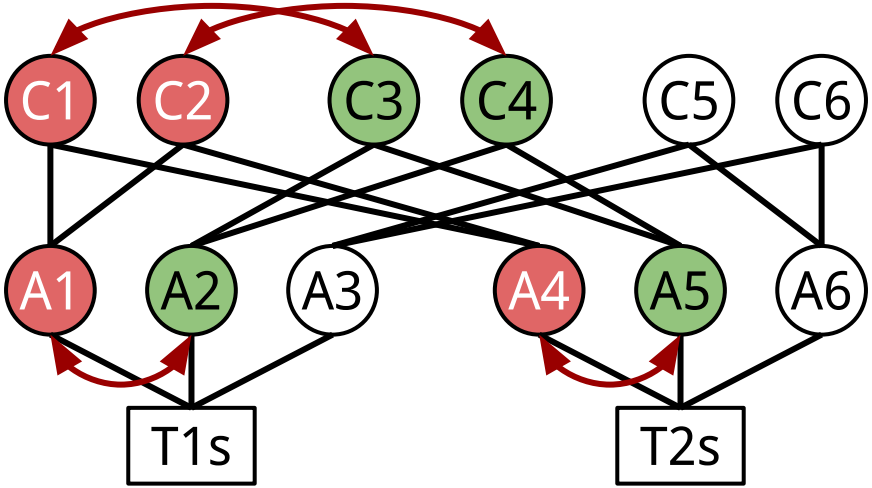




How to find equivalent subplans?
Enumerate all subplans and check pairwise? No!

Use network automorphism to find renaming function

- f preserves the three properties for the original network
  - Subplan A is equivalent to subplan f(A)



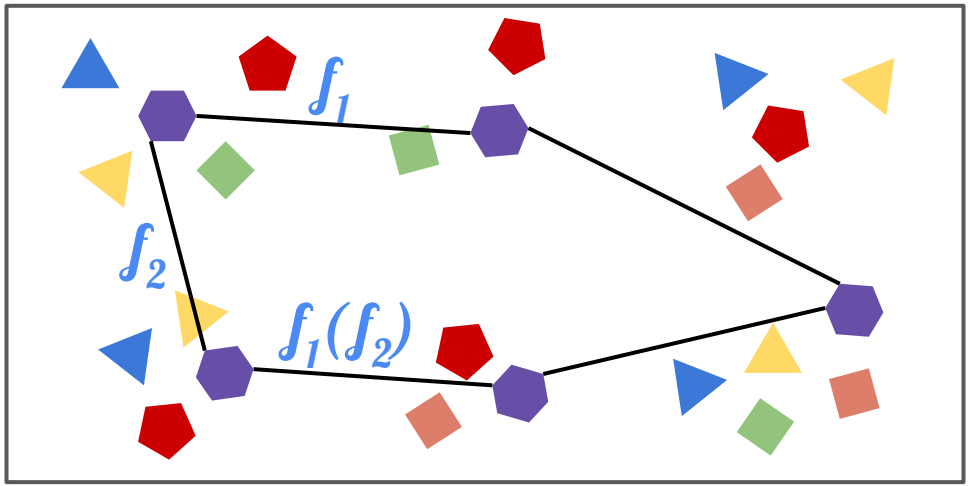
$\{C1\} = f(\{C1\}) = \{C3\}$

f: swap the red switches and green switches

Note: connectivity and routing are kept the same

Use network automorphism to find renaming function

- f preserves the three properties for the original network
  - Subplan A is equivalent to subplan f(A)
  - Link the plan with the same cost in the search space

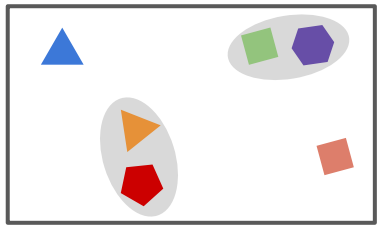


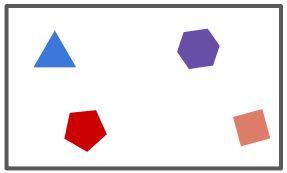
Path redundancy in data centers

Observation:

- Data center network has many path choices between servers
  - E.g., 10 aggregate, 60 core switches in clos topology has 600 paths
- Many subplans have similar cost
  - Taking down 5/60 vs. 6/60 core switches, the resulting cost difference is small

Solution: Discretize the steps (0, 4, 8, … 60)



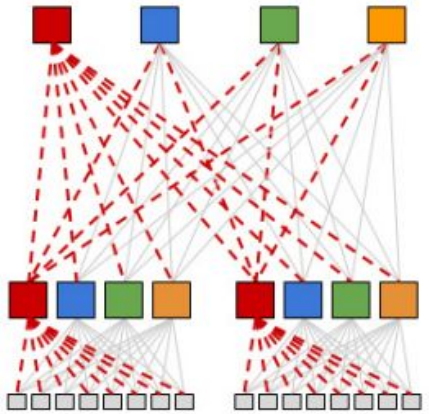


Janus solves other challenges

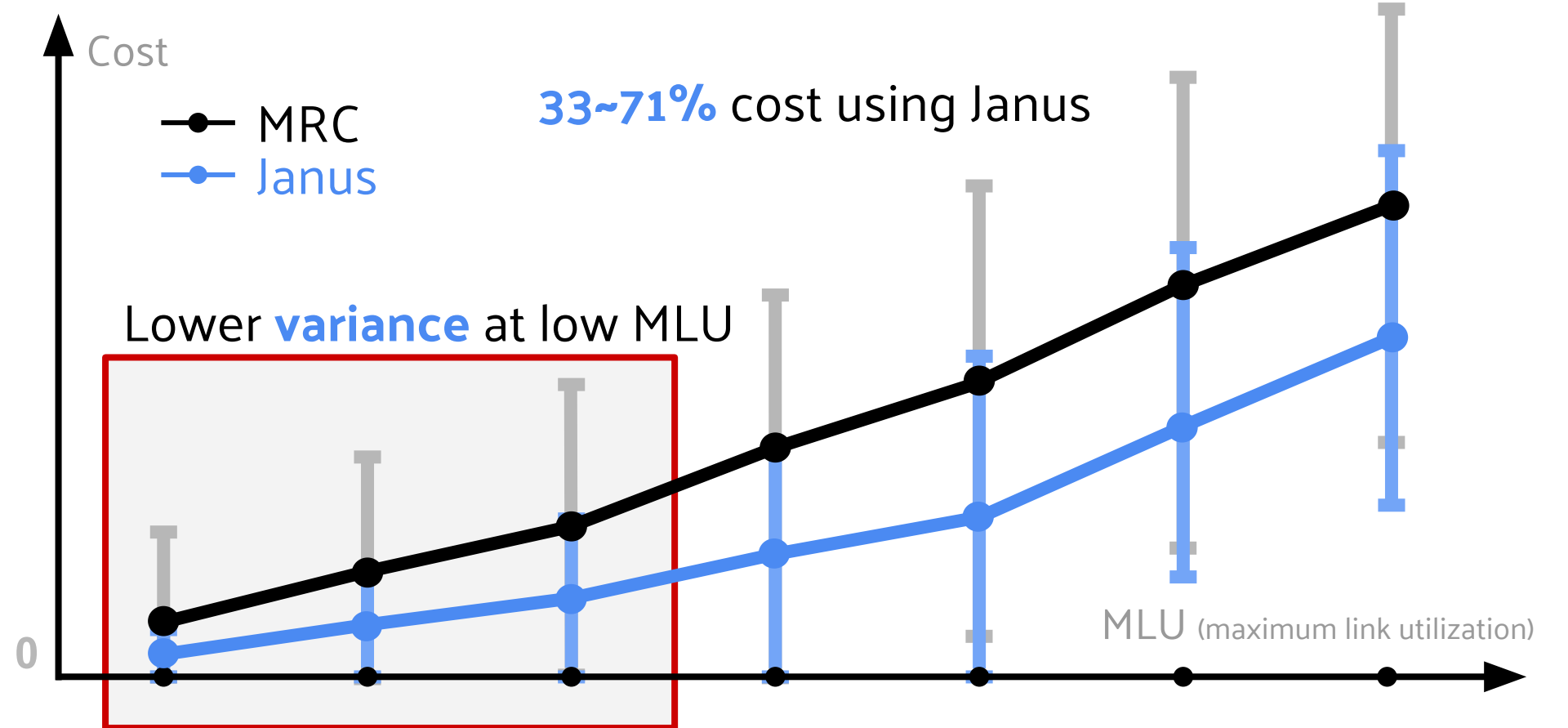
- Calculating the impact of subplans is slow at large scale
  - Build a quotient network graph
  - 4100x acceleration in planning time
- Failure
  - Map each failure scenario to a pre-compute scenario
- Roll-back
- Delayed changes

Evaluation Setup

- **Jupiter-like topology**
  - 168 switches, 3840 servers
- **Change**
  - Upgrade all aggregate and core switches (72 switches)
  - Staged cost function based on Amazon SLA
- **Traffic**
  - Google Borg tenant arrival times
  - FB traffic traces
- **Baseline**
  - MRC (Maximize residual capacity)



Evaluation: Benefits over MRC under different traffic



Evaluation: Adaptivity

Janus outperforms MRC under different network factors

- Cost functions
- Deadlines
- Data center scales
- Traffic settings
- Failures
- Rollback plans

Evaluation: scalability

Topology (# servers)	Topology / Upgrade (# switches)	Planning time (20 cores)
4,000 servers	168 / 72 switches	0.125 sec
16,000 servers	600 / 216 switches	0.503 sec
32,000 servers	1350 / 486 switches	1.795 sec
64,000 servers	2400 / 864 switches	8.75 sec

Conclusion

- Data centers are constantly changing
  - Fast network changes are critical for enabling quick evolution of data center
- Planning network changes should be
  - Adaptive to all kinds of network settings
  - Scalable to search the universe
- Janus leverages the high degree of symmetry and redundancy to plan network changes in real time

Thank you!