Efficient Distributed Workload (Re-)Embedding

Monika Henzinger

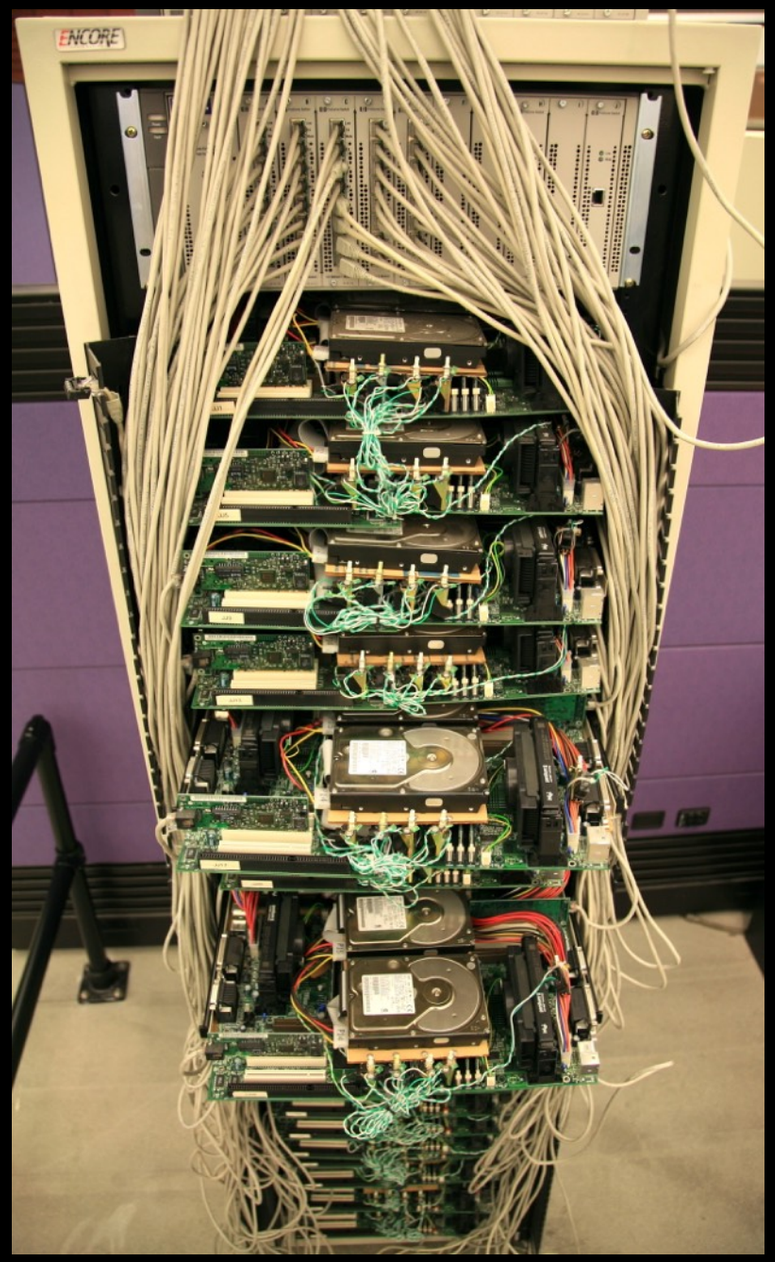
Stefan Neumann

Stefan Schmid

universität wien

Many Years Ago

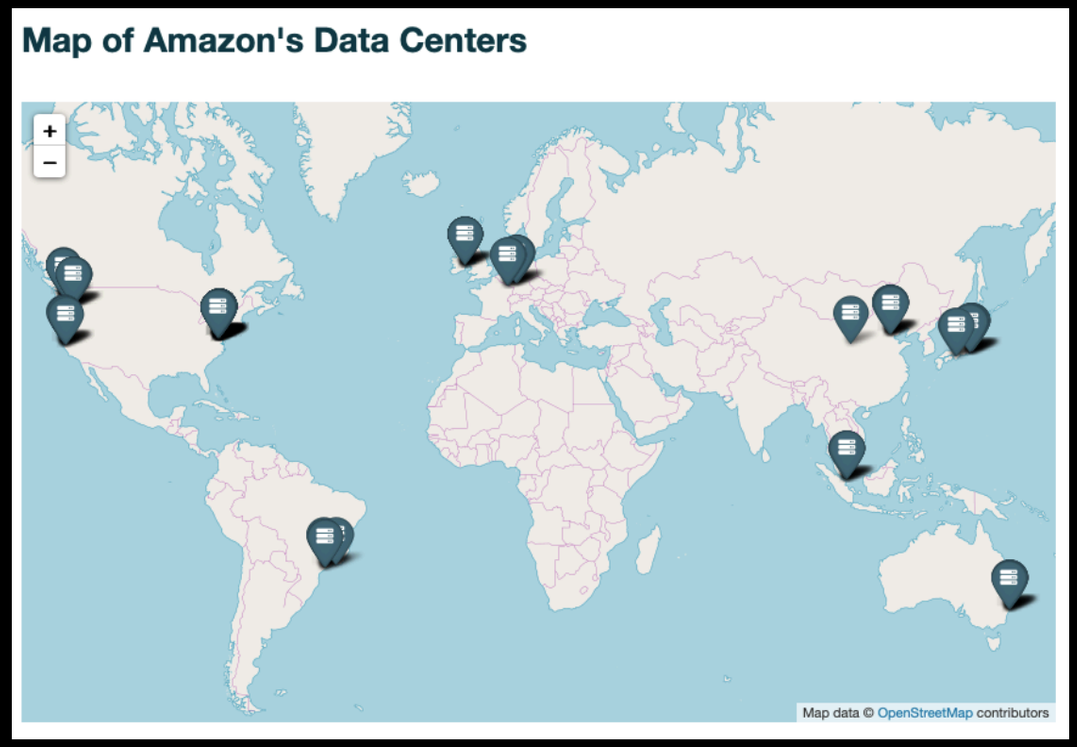
- Single server
- Systems were fixed and workload-agnostic
- Simple communication patterns (if at all), endpoints fixed



https://www.flickr.com/photos/jurvetson/157722937

Nowadays

- Large distributed systems (even geographically distributed): communication over network
- Virtualization technologies enable workload-aware operations that improve system efficiency
- Communicating processes can be far away and re-locating them is costly

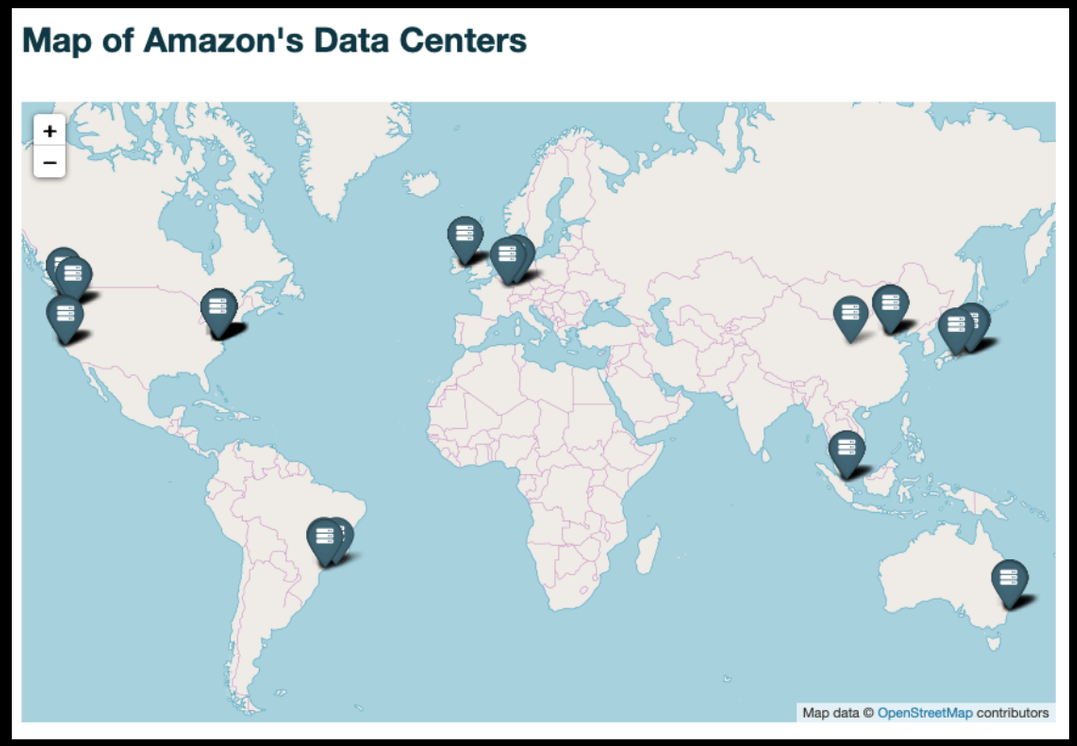


https://wikileaks.org/amazon-atlas/map/

https://commons.wikimedia.org/wiki/File:Bacloud.com_data_center.JPG

Nowadays

- Large distributed systems (even geographically distributed): communication over network
- Virtualization technologies enable workload-aware operations that improve system efficiency
- Communicating processes can be far away and re-locating them is costly
- Communication requests contain patterns

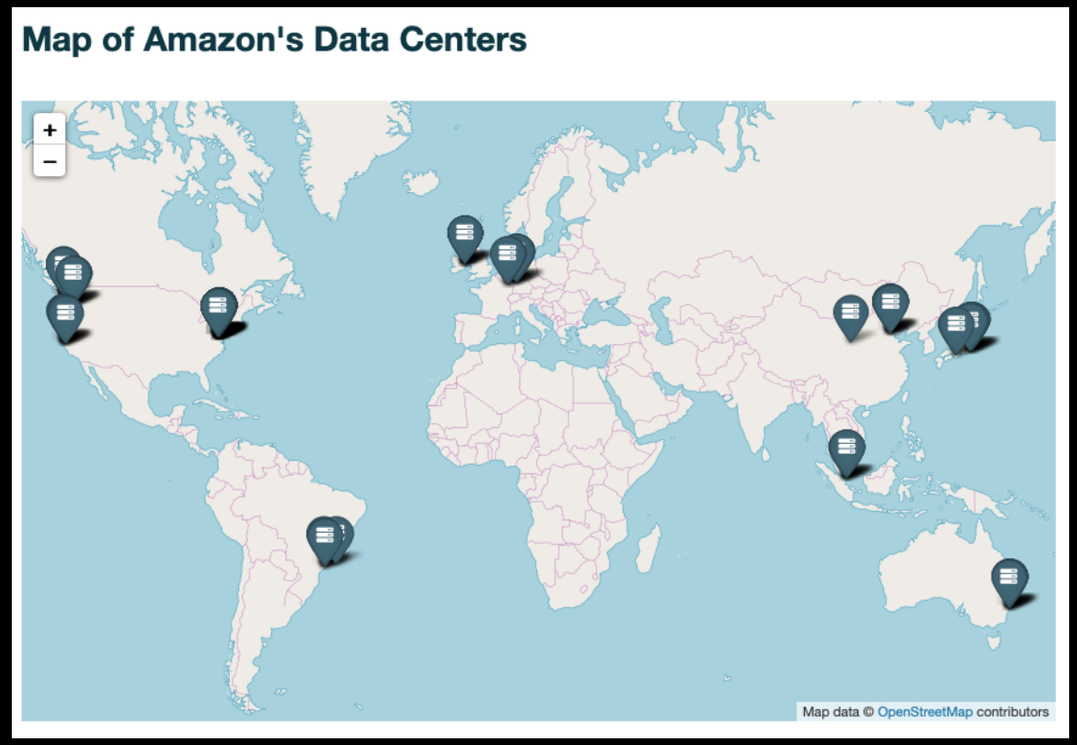


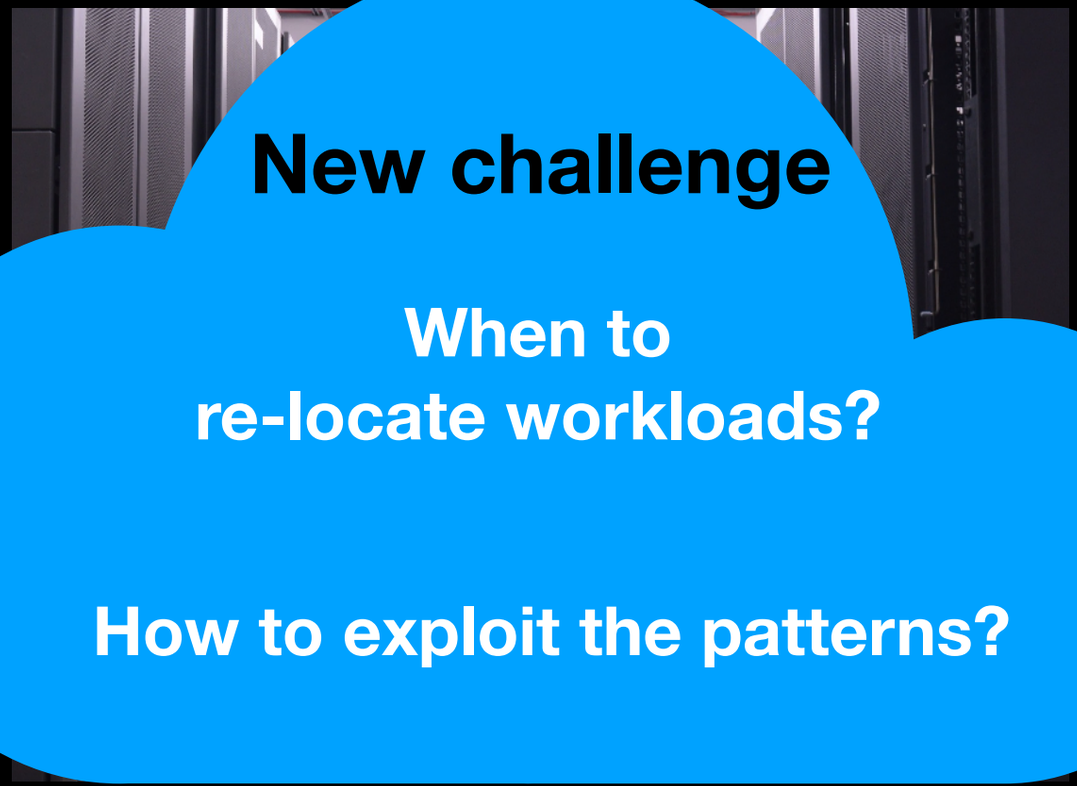
https://wikileaks.org/amazon-atlas/map/

https://commons.wikimedia.org/wiki/File:Bacloud.com_data_center.JPG

Nowadays

- Large distributed systems (even geographically distributed): communication over network
- Virtualization technologies enable workload-aware operations that improve system efficiency
- Communicating processes can be far away and re-locating them is costly
- Communication requests contain patterns





New challenge

When to re-locate workloads?

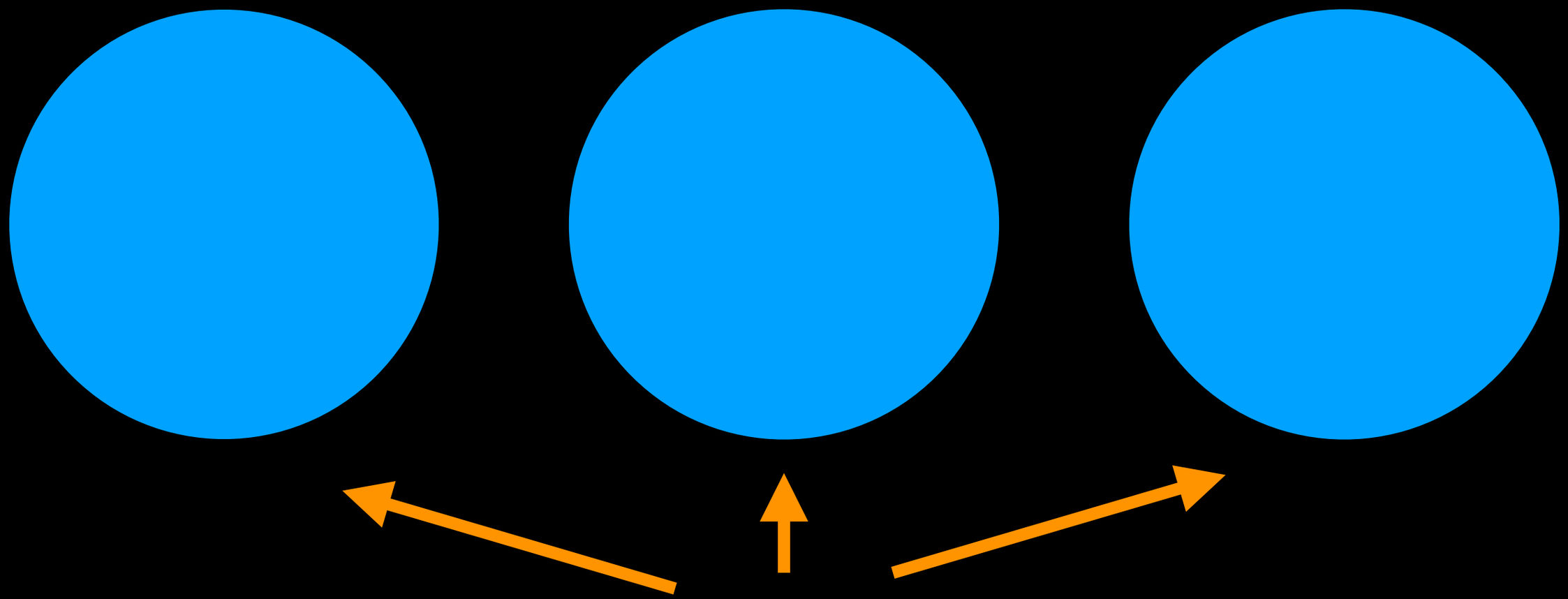
How to exploit the patterns?

https://wikileaks.org/amazon-atlas/map/

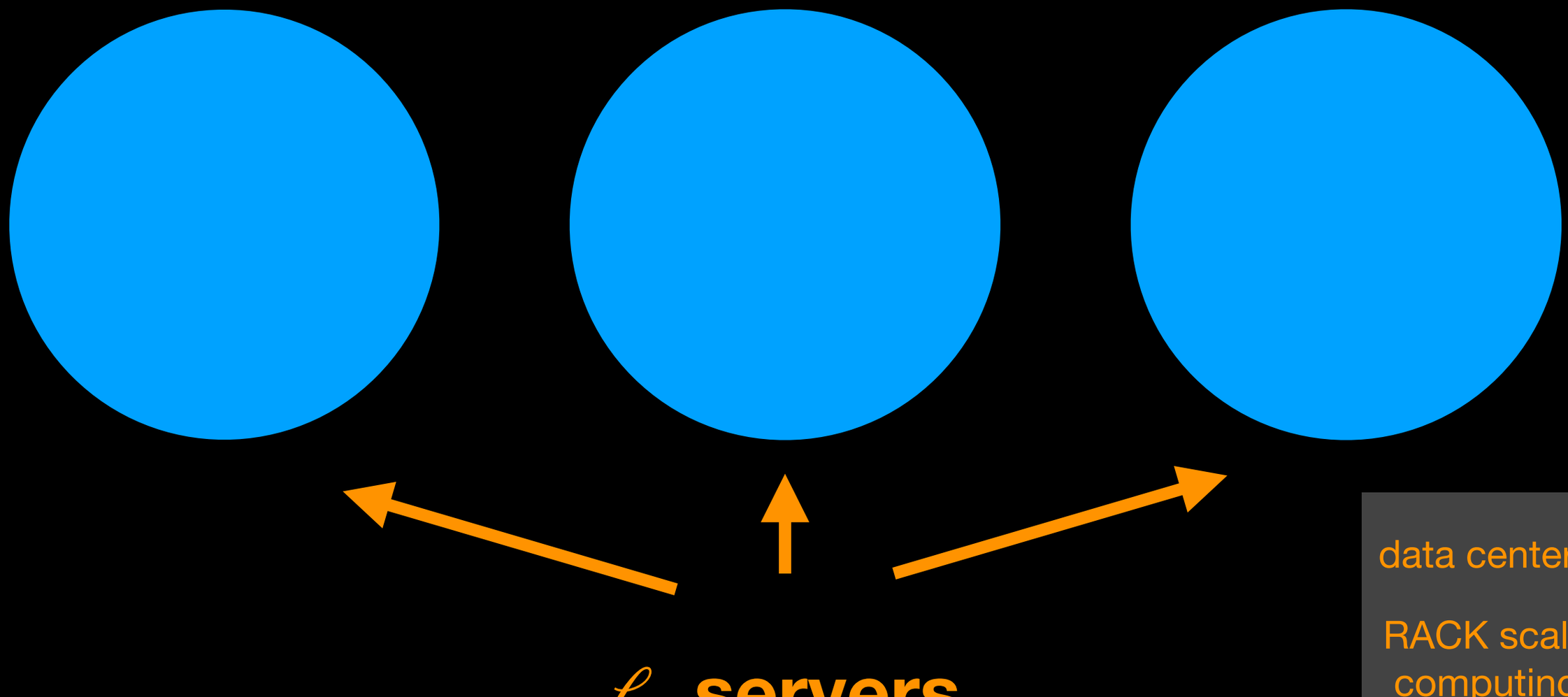
https://commons.wikimedia.org/wiki/File:Bacloud.com_data_center.JPG

The Model

The Model



The Model

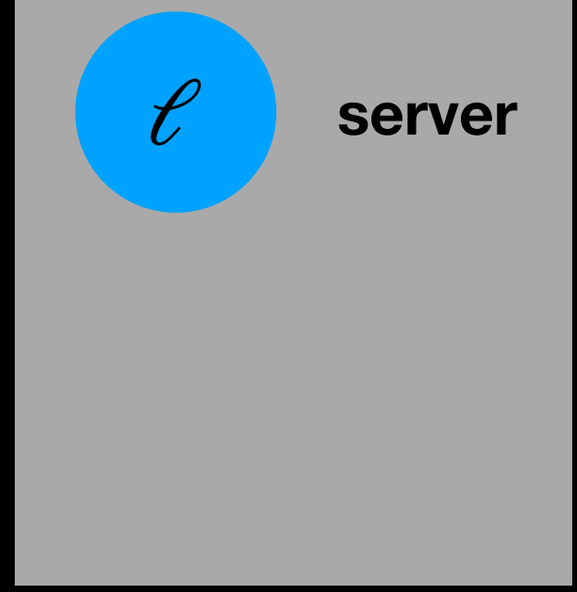


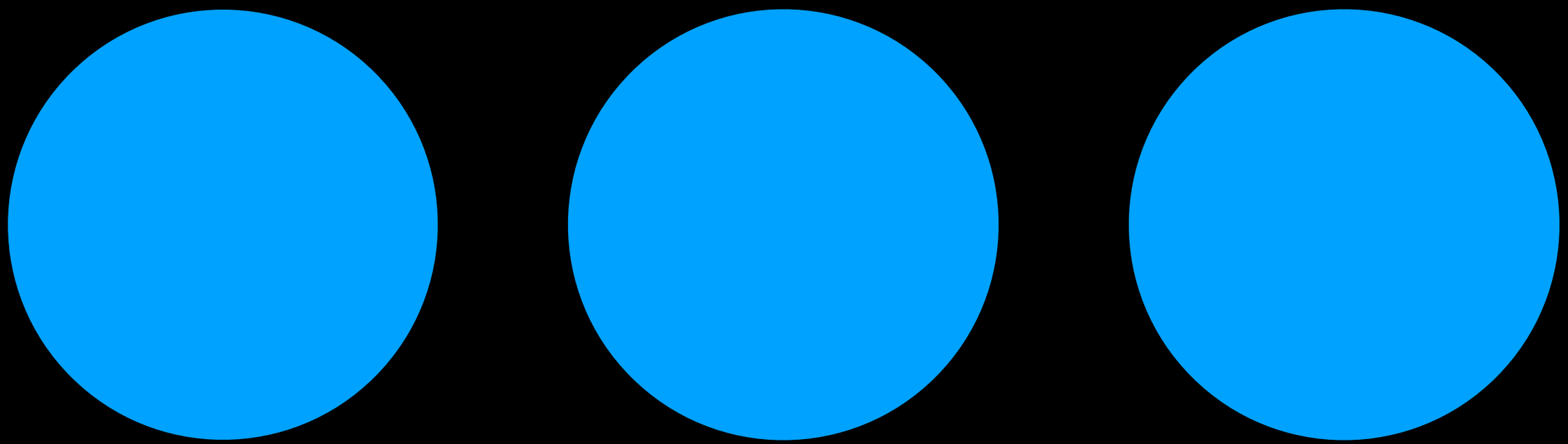
ℓ **servers**

data centers

RACK scale computing

The Model



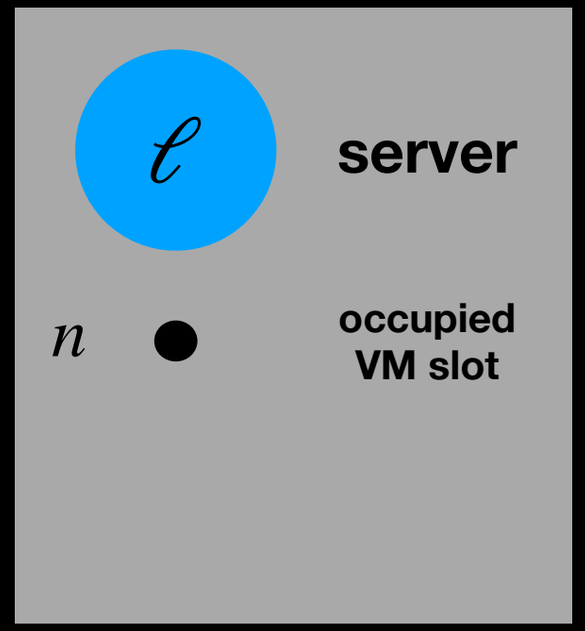


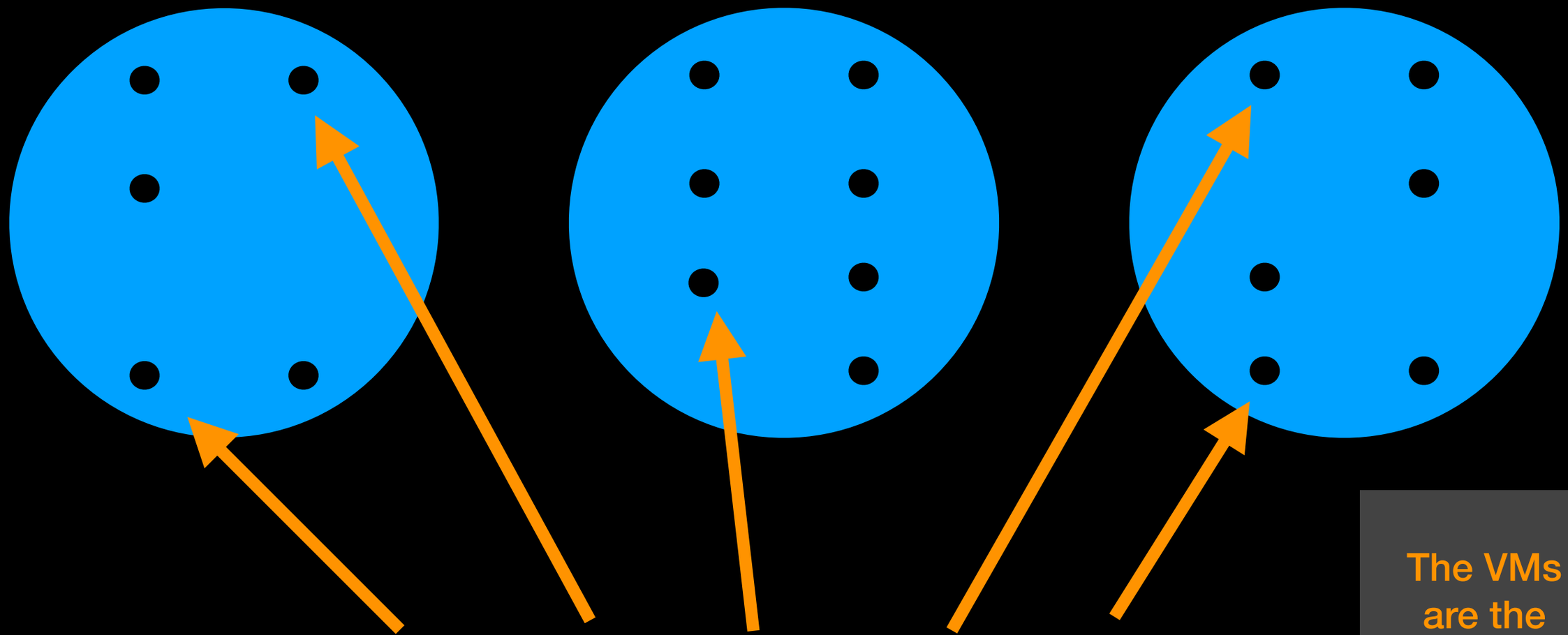
ℓ **servers**

data centers

RACK scale computing

The Model

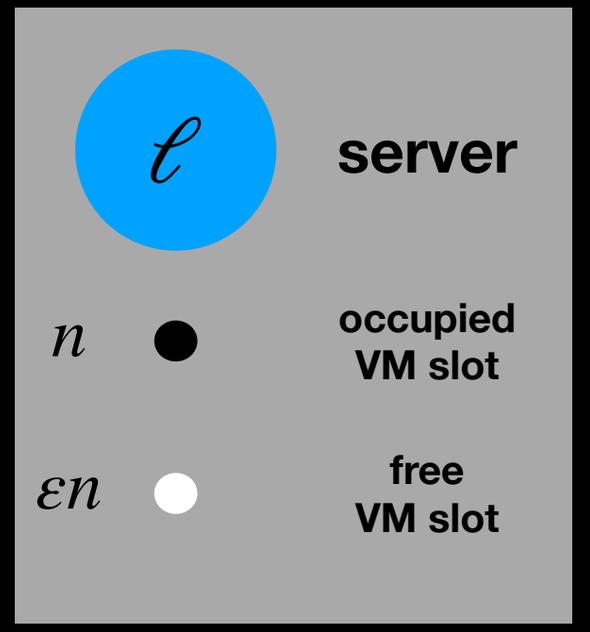


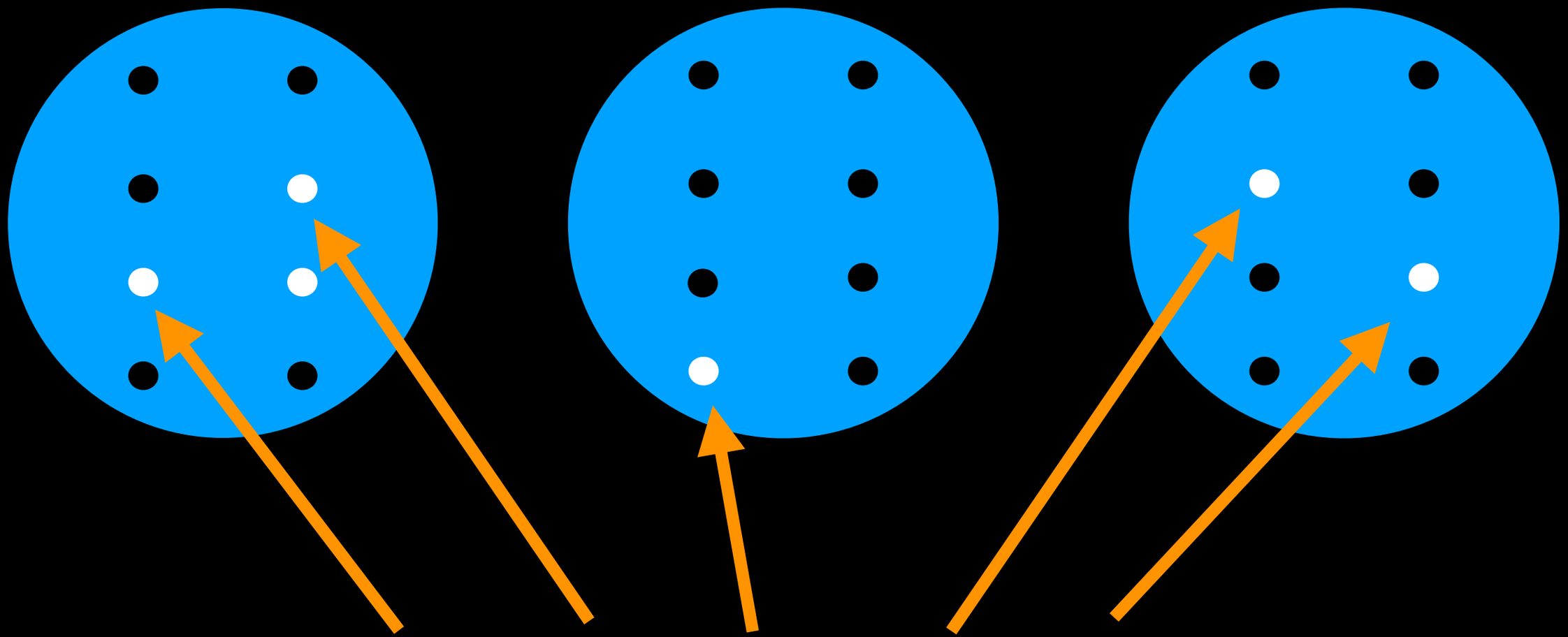


n **virtual machines (VMs)**

The VMs are the workloads.

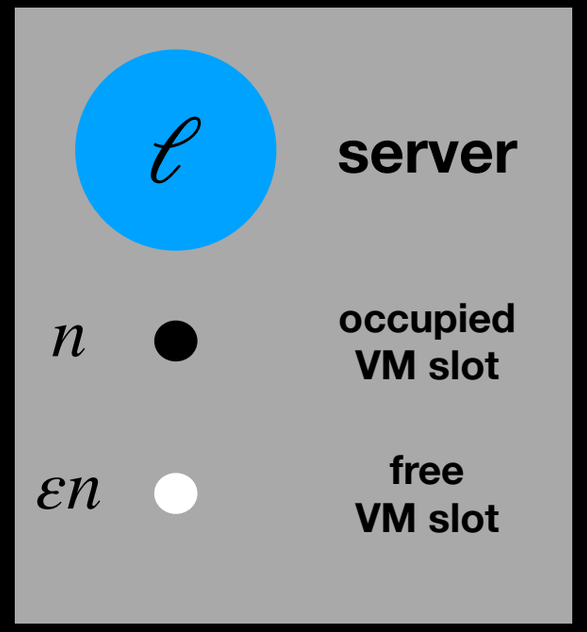
The Model

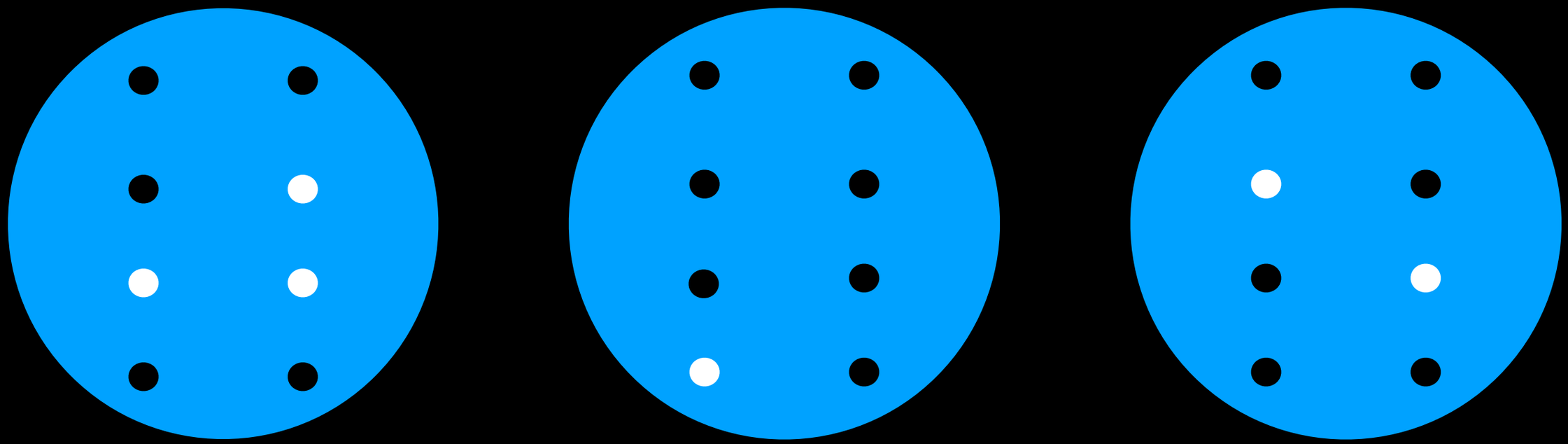




εn **additional slots for VMs**

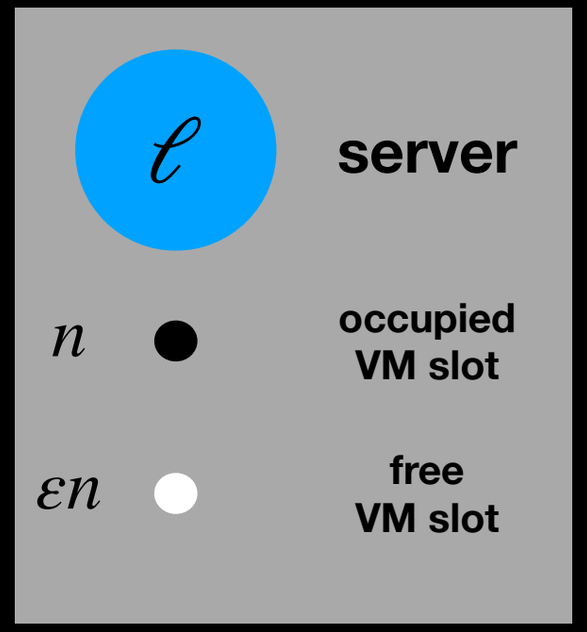
The Model

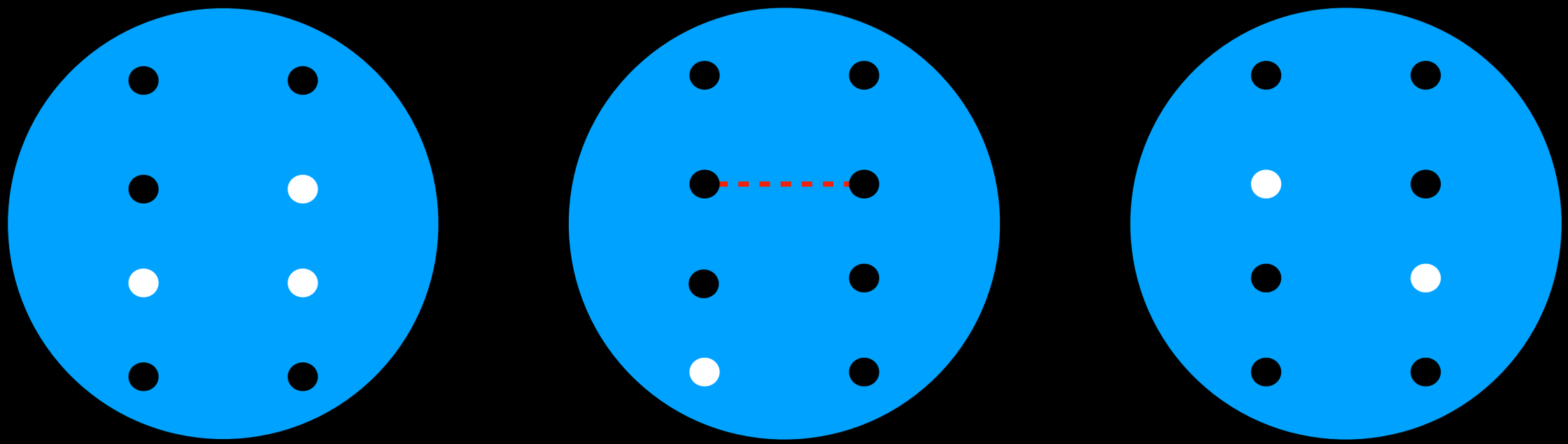




Communication requests arrive online

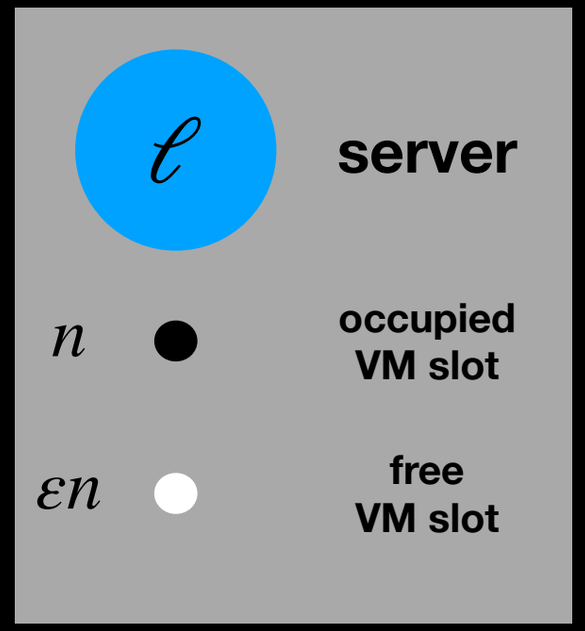
The Model

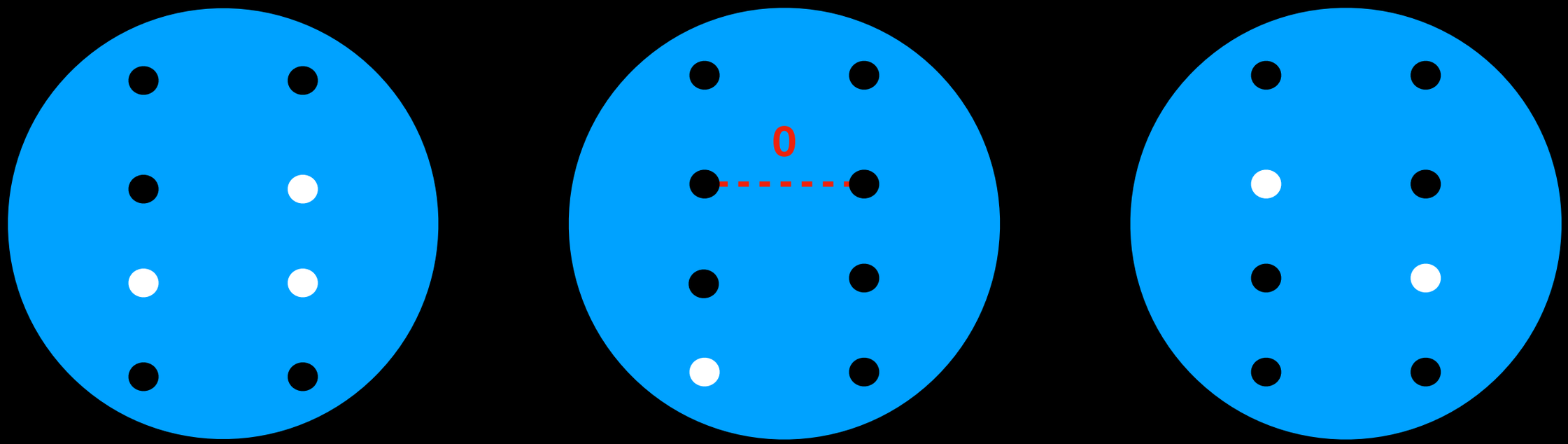




Communication requests arrive online

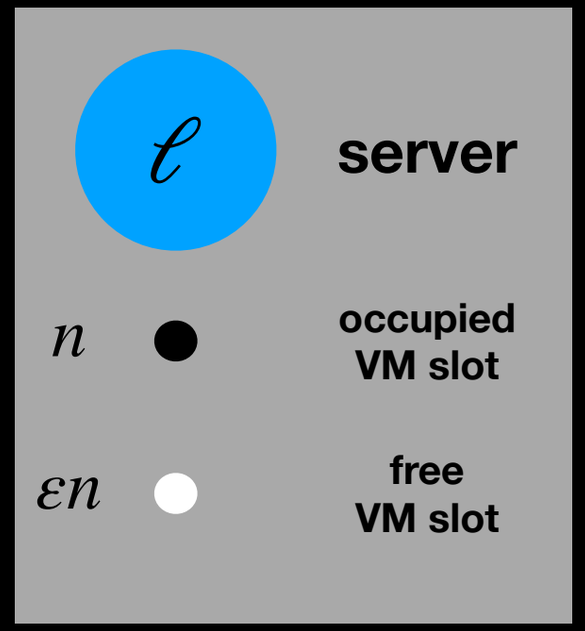
The Model

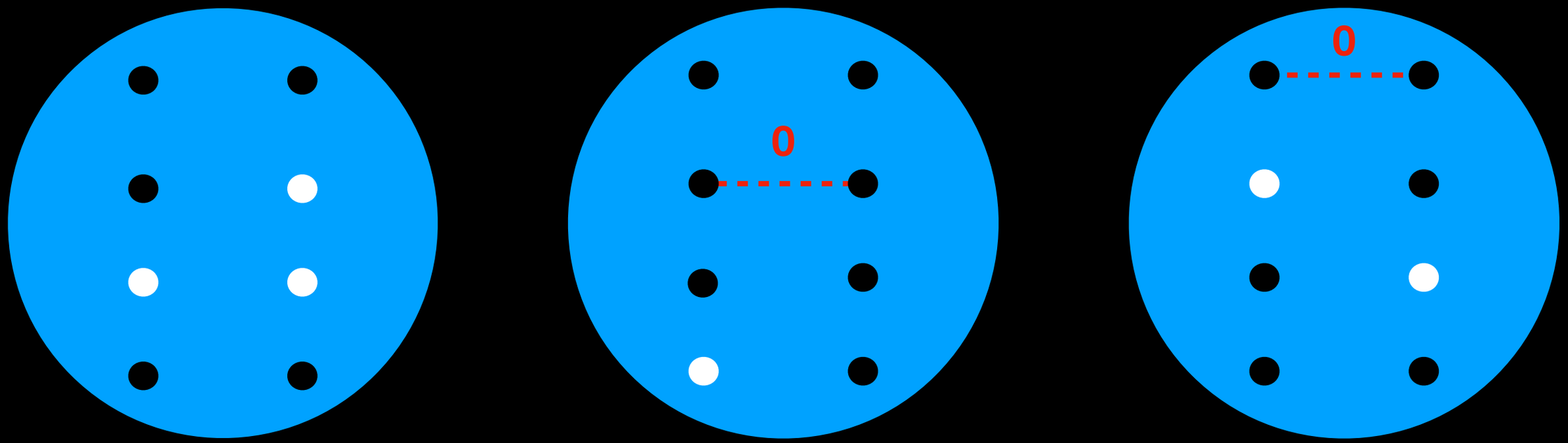




Communication requests arrive online

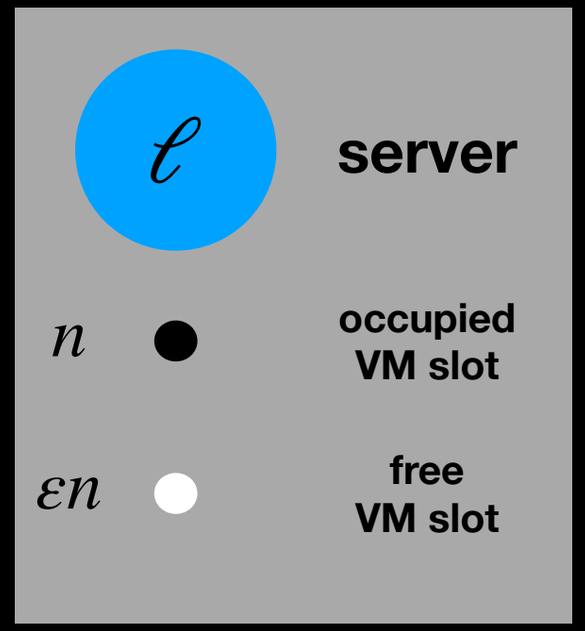
The Model

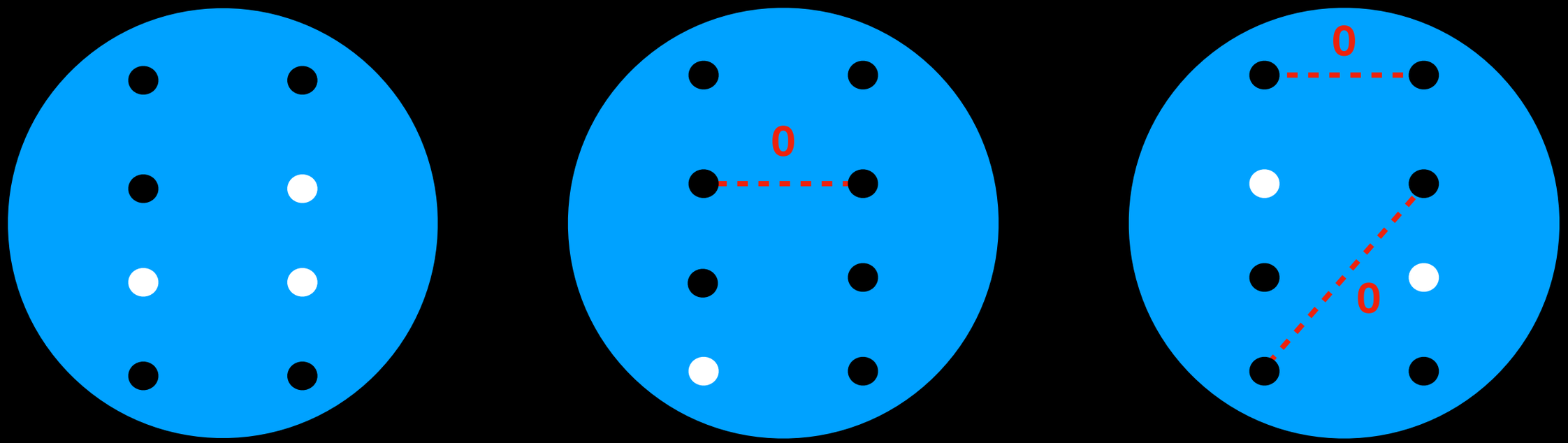




Communication requests arrive online

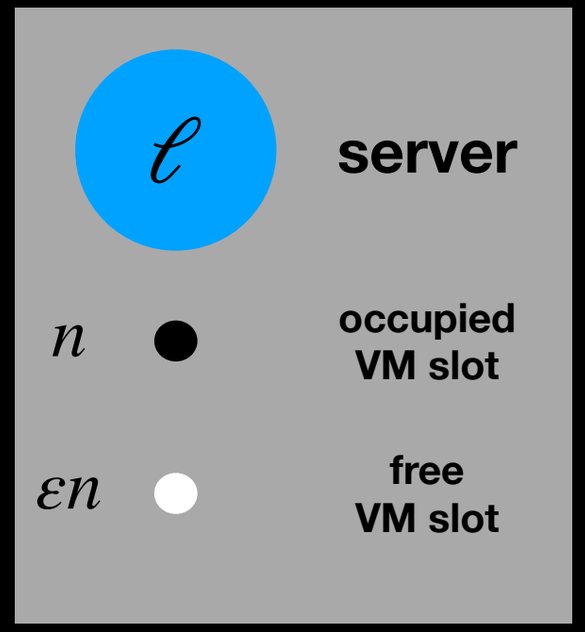
The Model

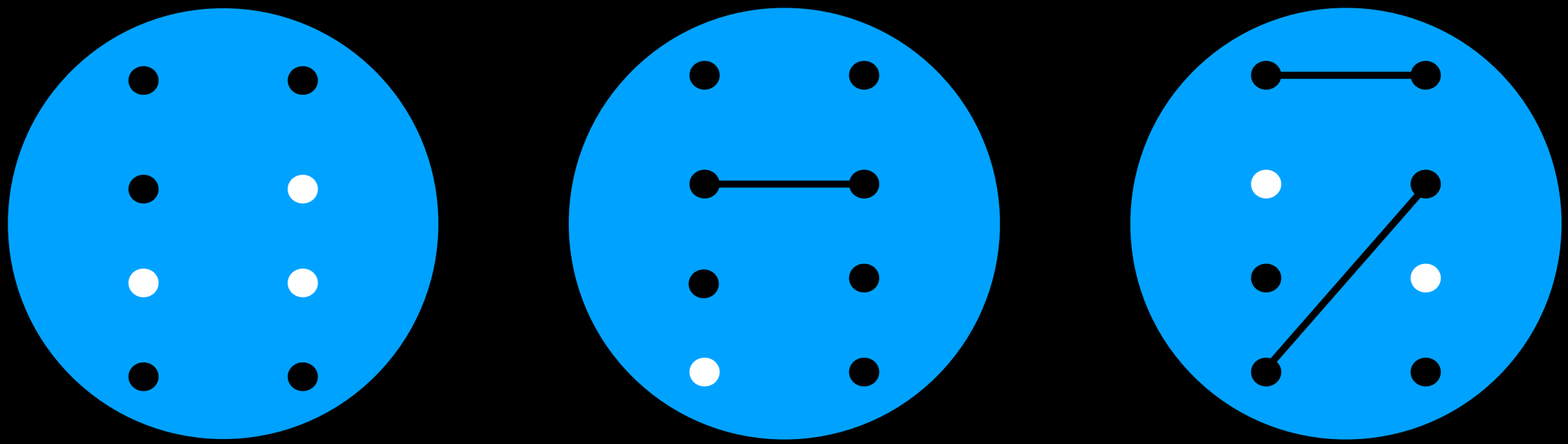




Communication requests arrive online

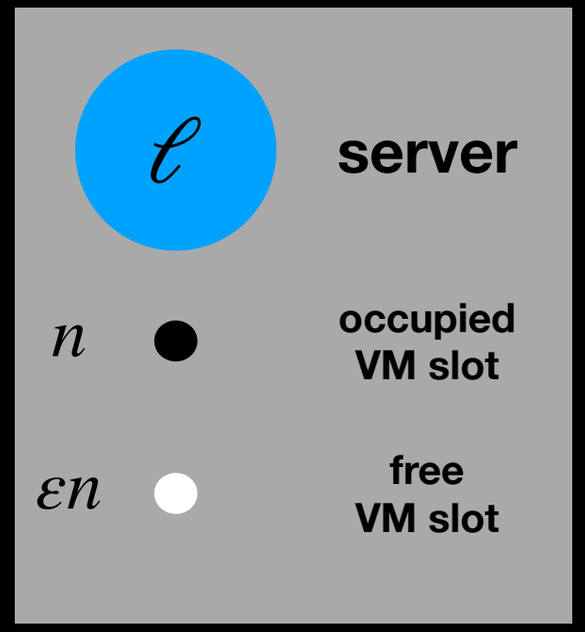
The Model

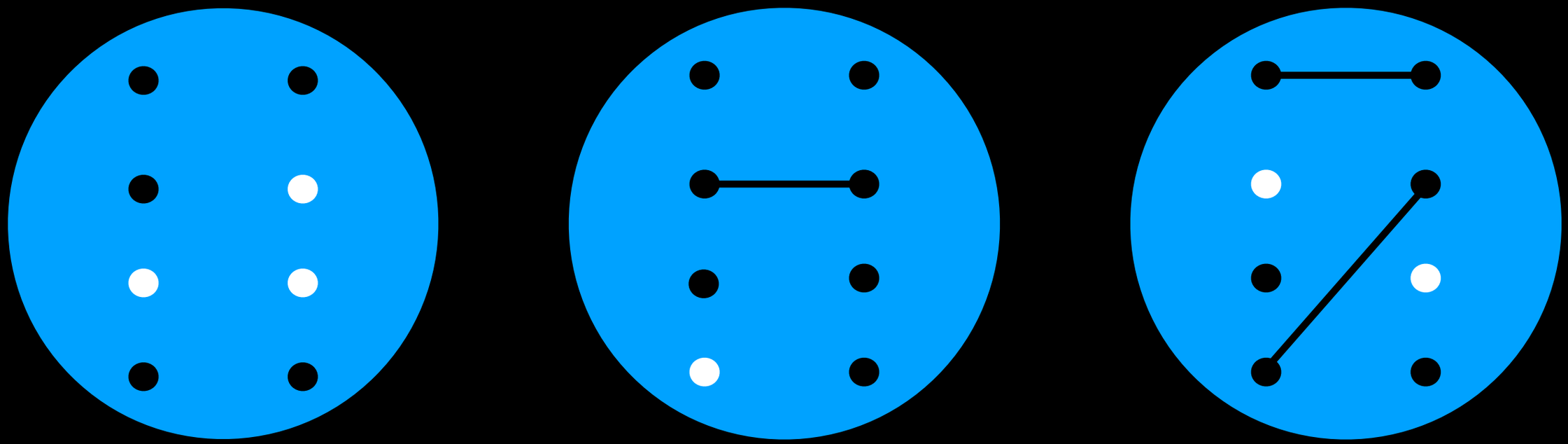




Old communication links stay *forever*

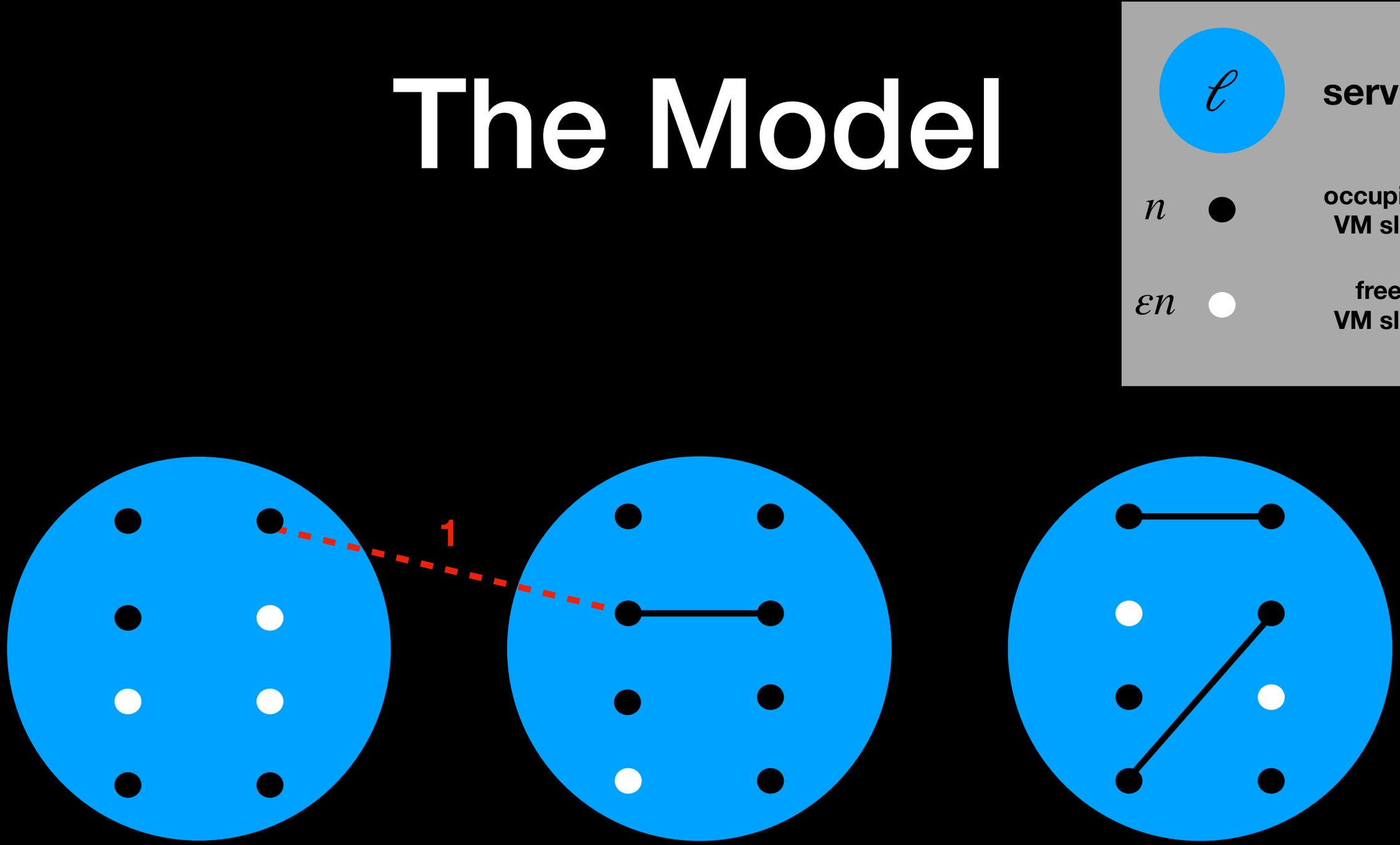
The Model





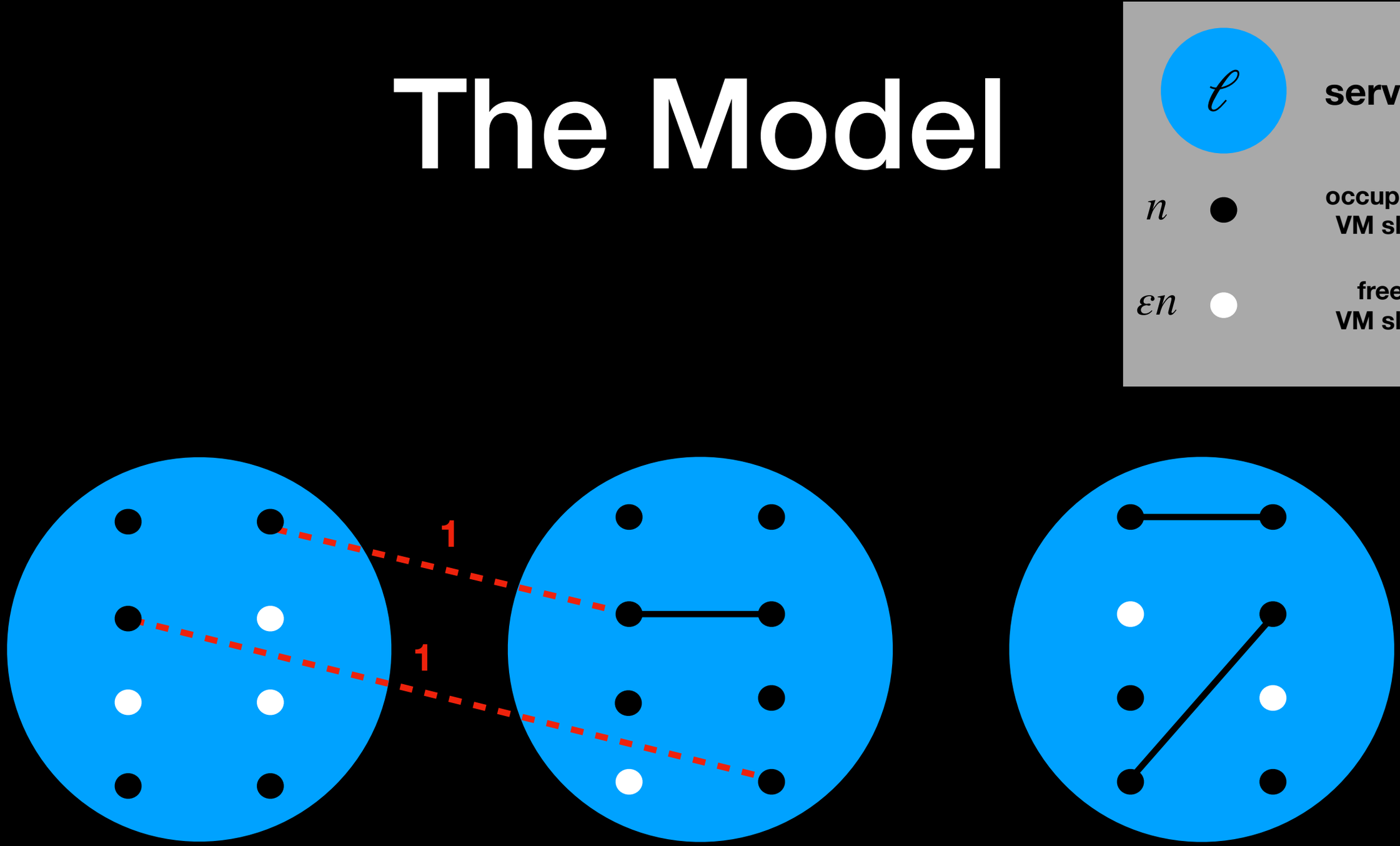
Communication requests arrive online

The Model



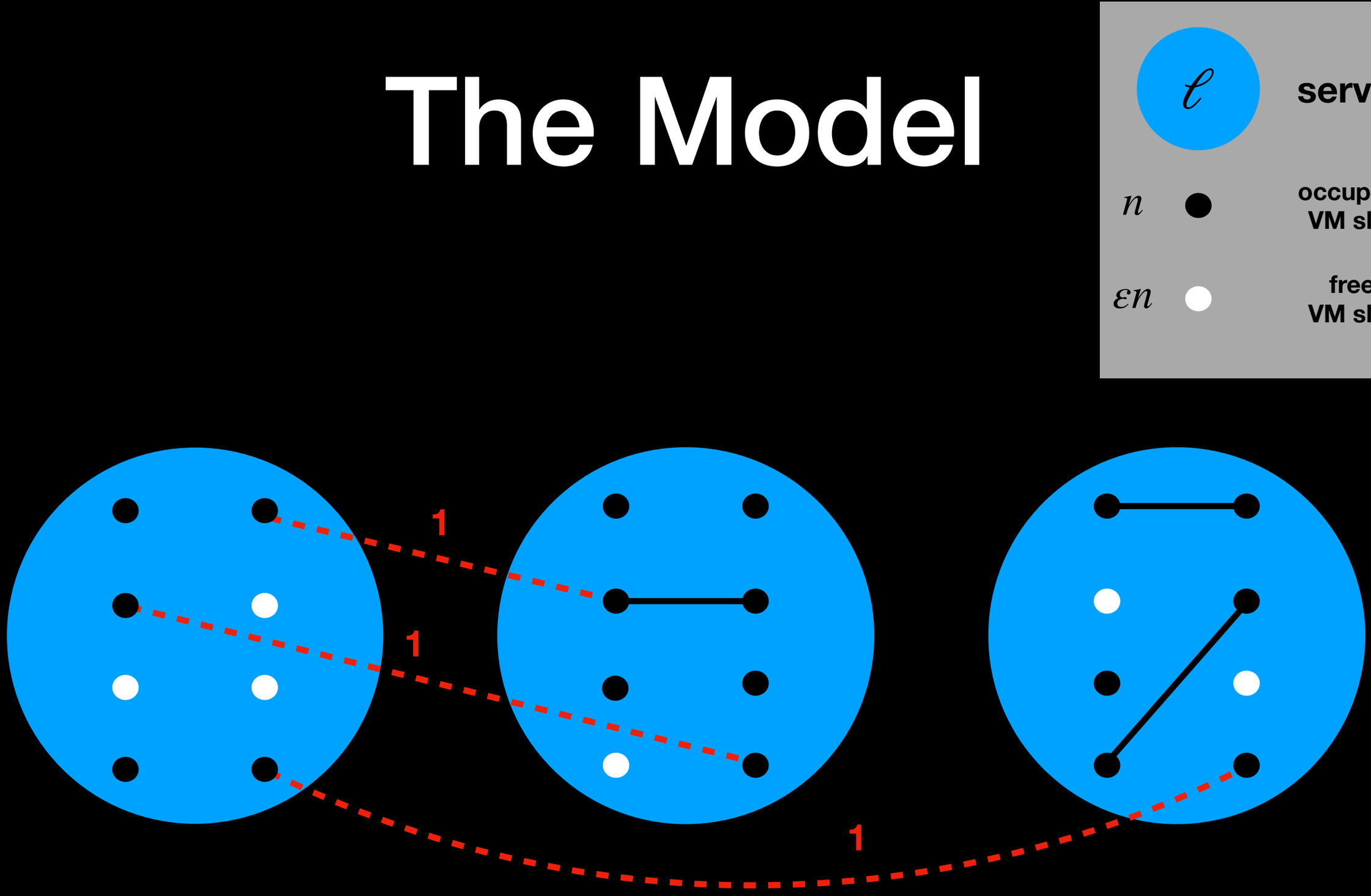
Communication requests arrive online

The Model



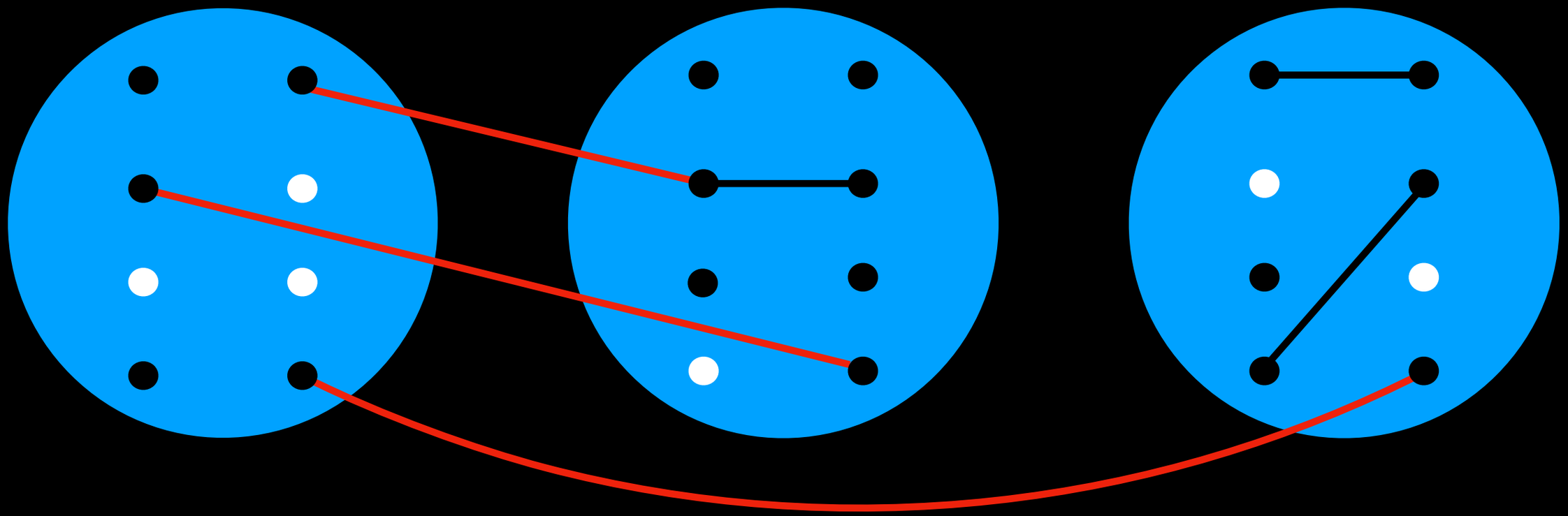
Communication requests arrive online

The Model



Communication requests arrive online

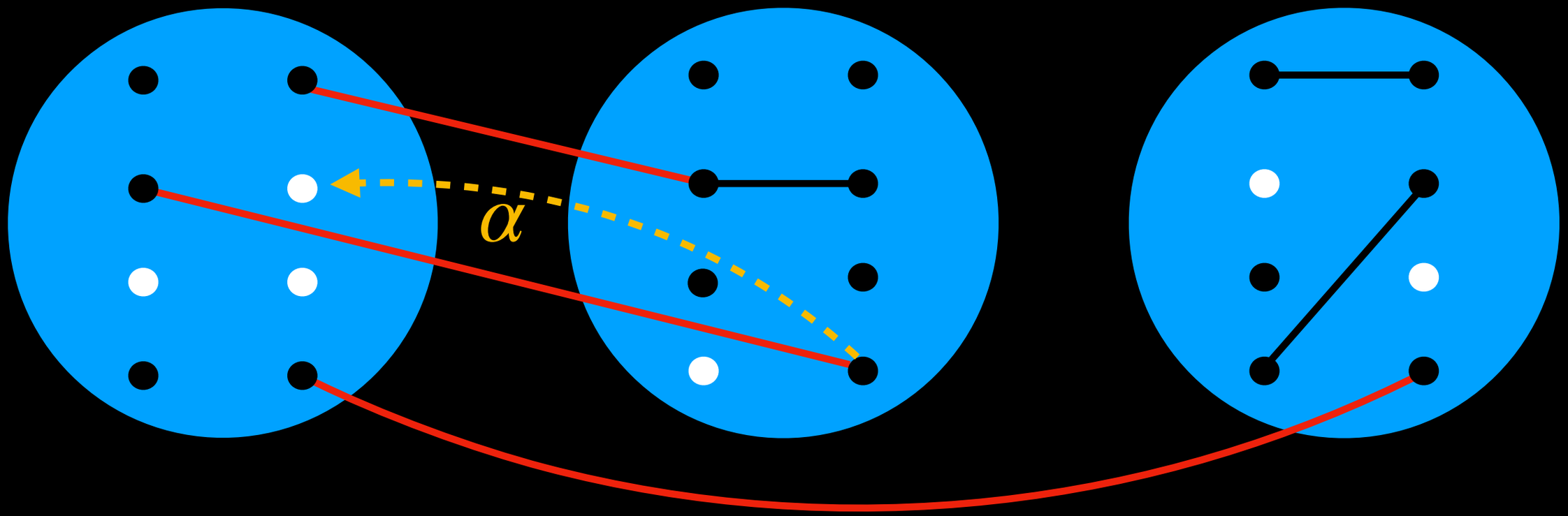
The Model



Re-locate VMs to avoid cross-server communication

re-location cost
$\alpha > 1$

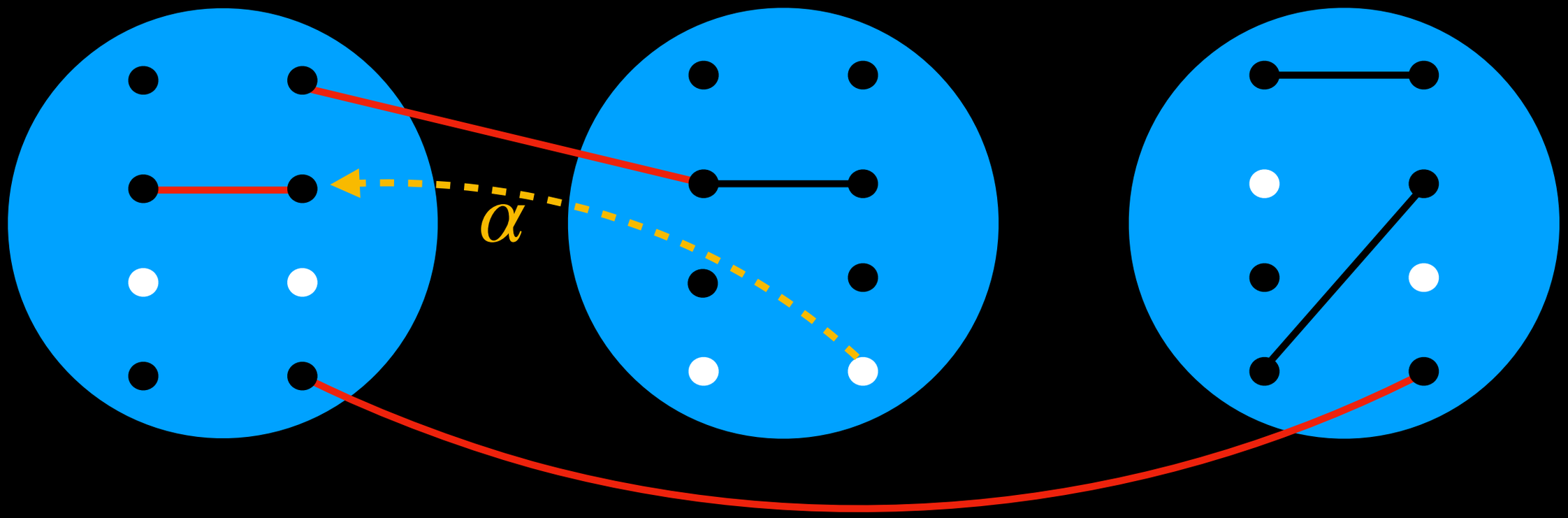
The Model



Re-locate VMs to avoid cross-server communication

re-location cost
$\alpha > 1$

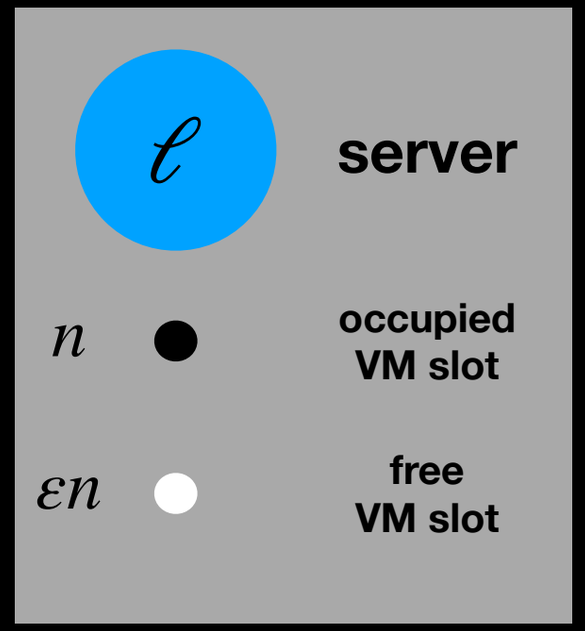
The Model

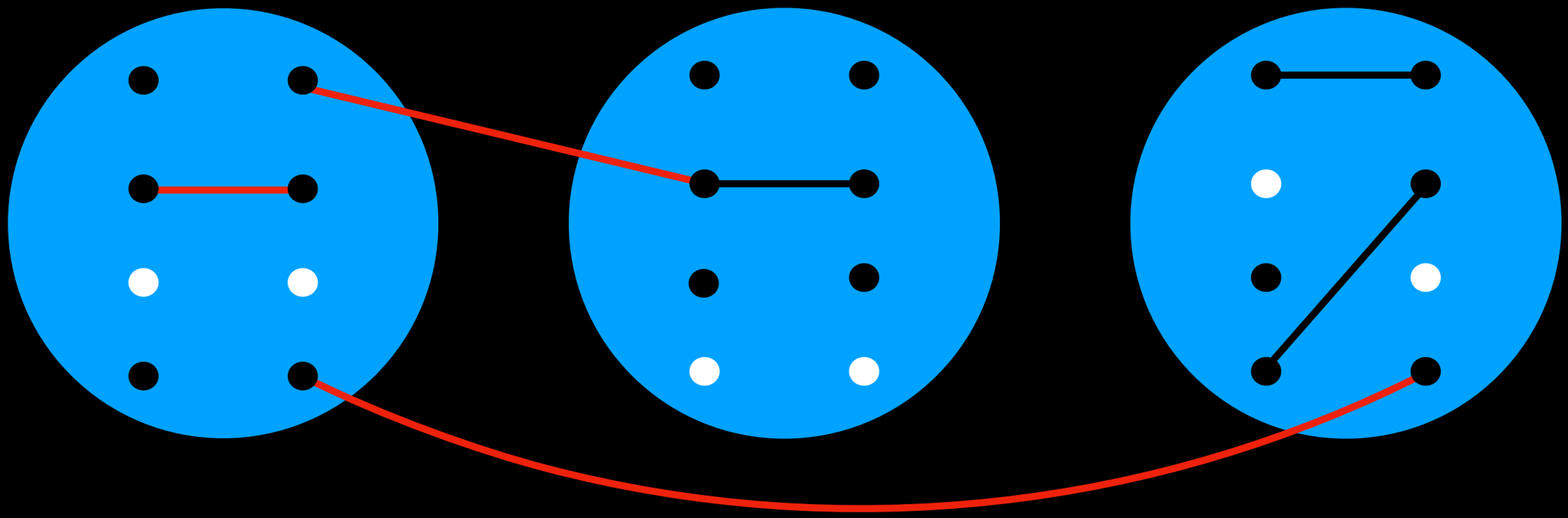


Re-locate VMs to avoid cross-server communication

re-location cost
$\alpha > 1$

The Model

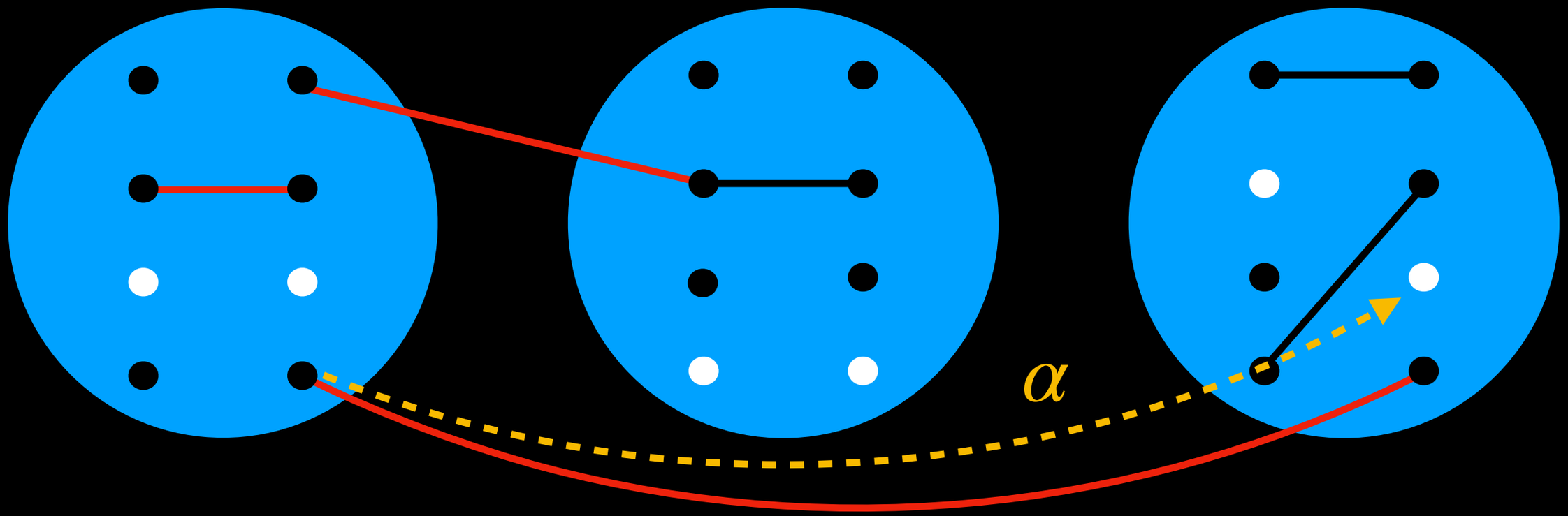




Re-locate VMs to avoid cross-server communication

re-location cost
$\alpha > 1$

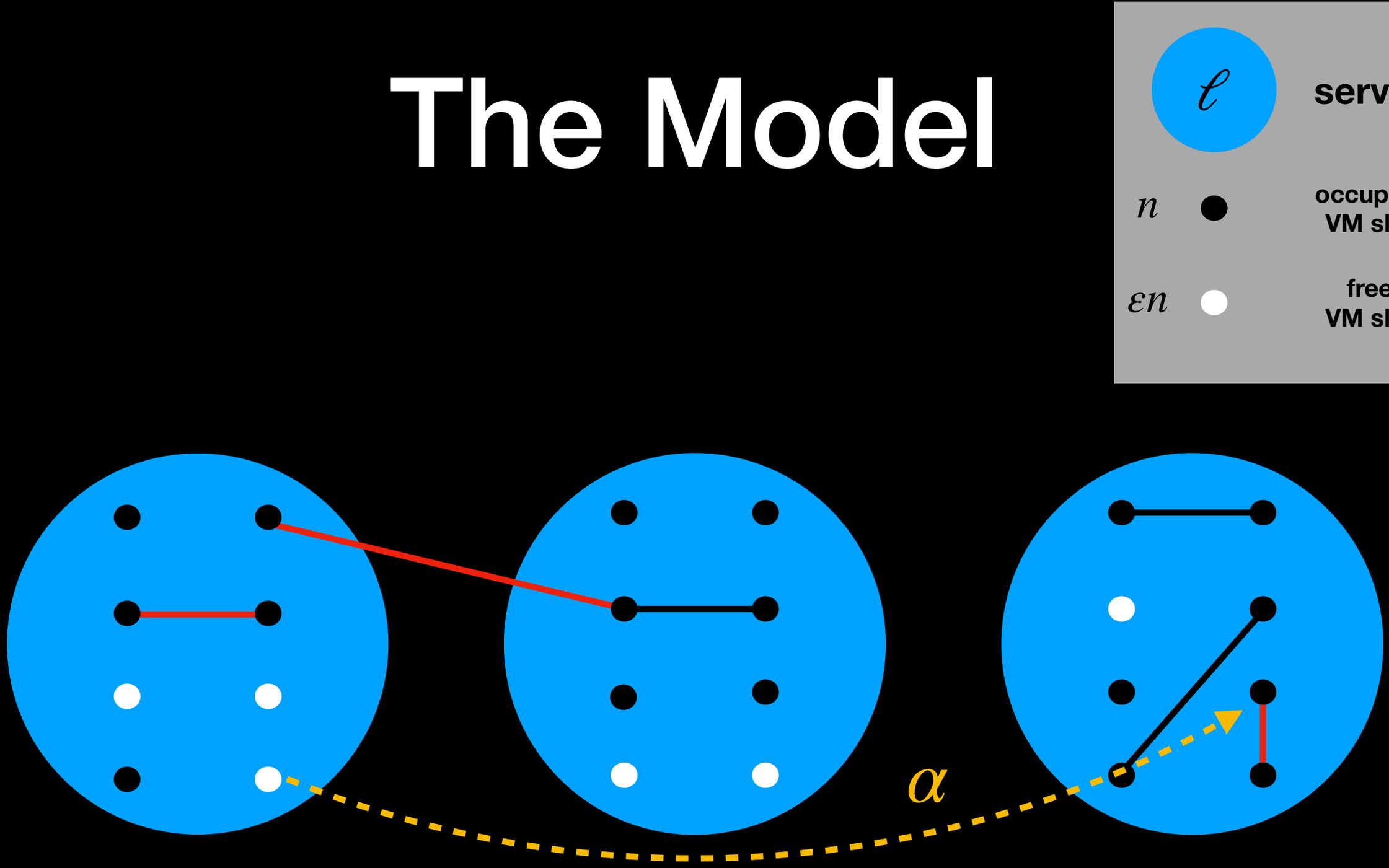
The Model



Re-locate VMs to avoid cross-server communication

re-location cost
$\alpha > 1$

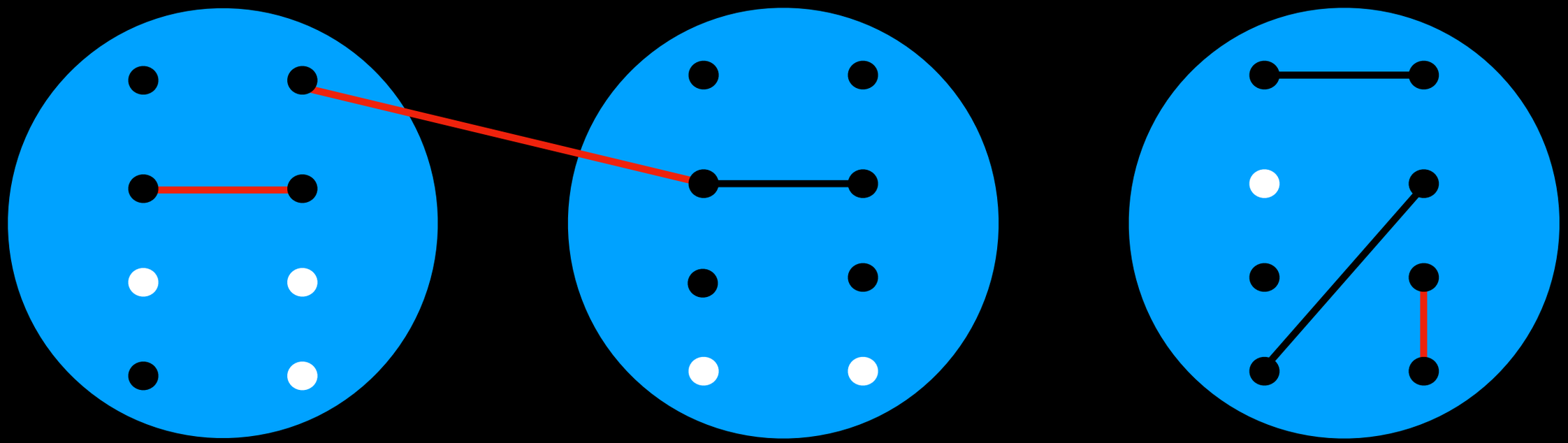
The Model



Re-locate VMs to avoid cross-server communication

re-location cost
$\alpha > 1$

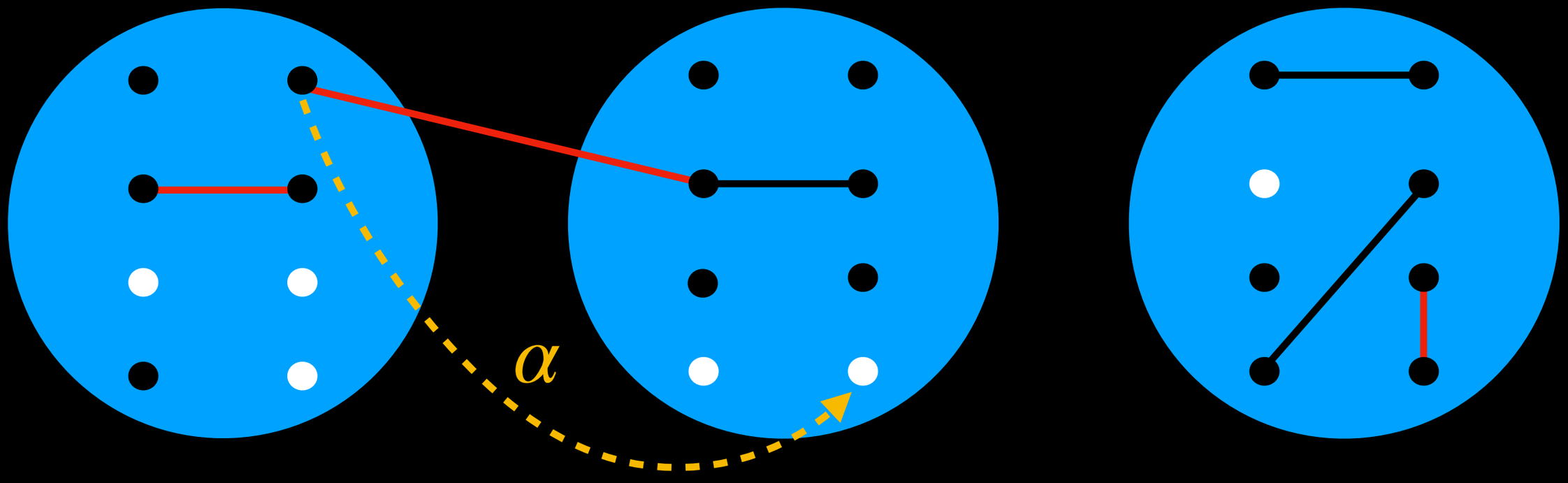
The Model



Re-locate VMs to avoid cross-server communication

re-location cost
$\alpha > 1$

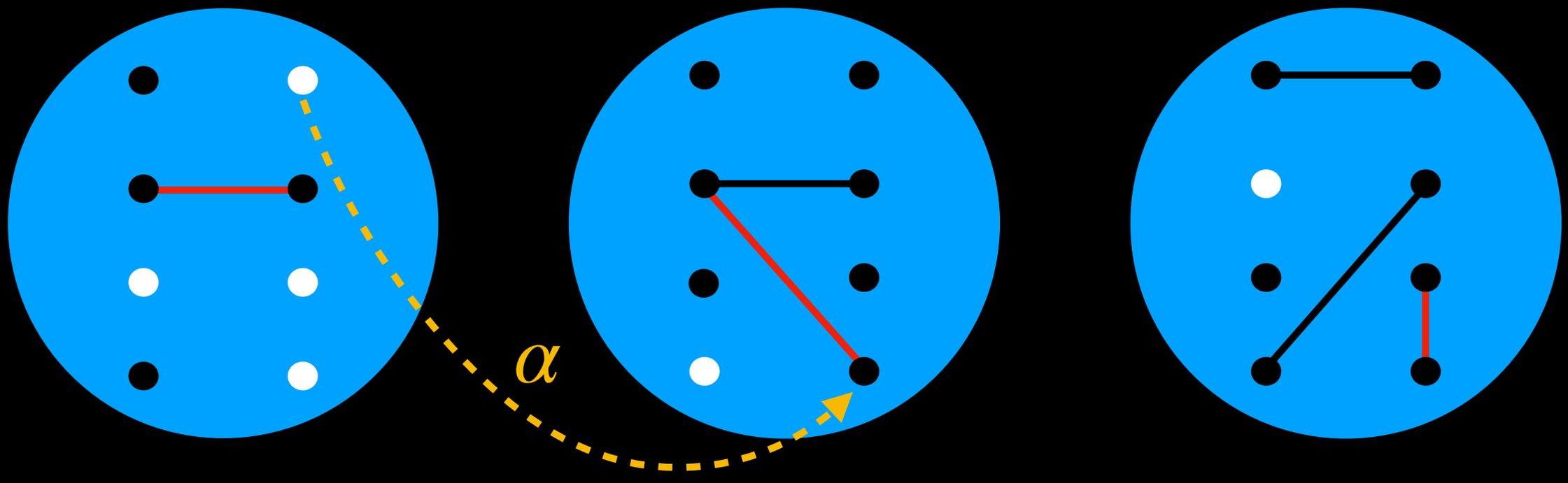
The Model



Re-locate VMs to avoid cross-server communication

re-location cost
$\alpha > 1$

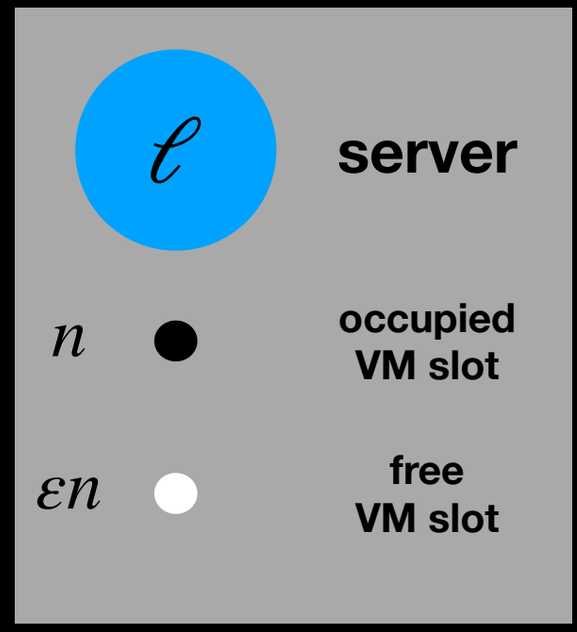
The Model

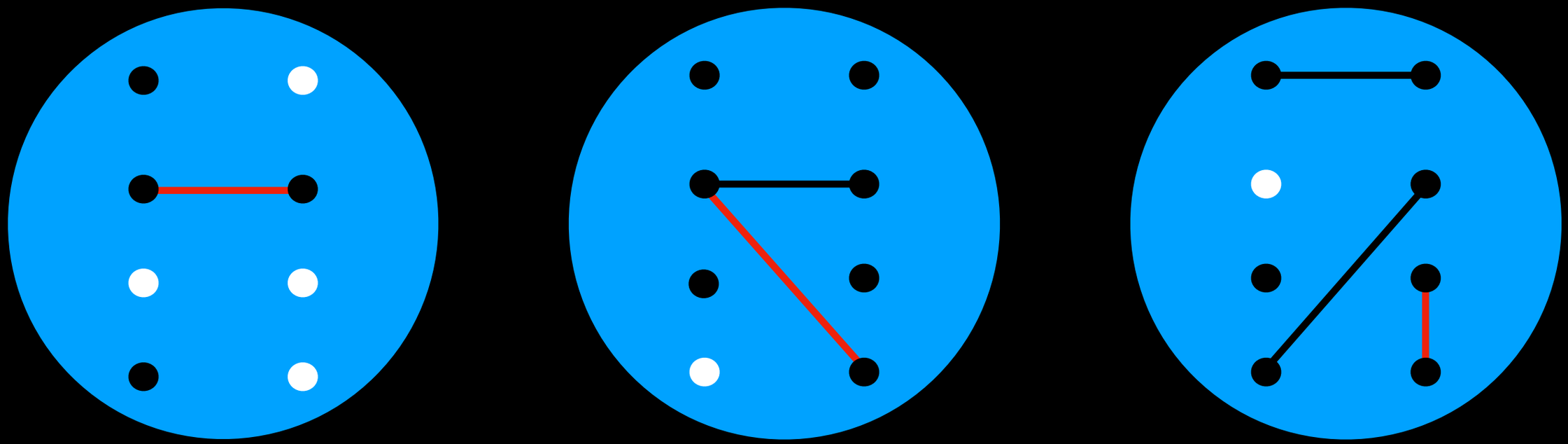


Re-locate VMs to avoid cross-server communication

re-location cost
$\alpha > 1$

The Model

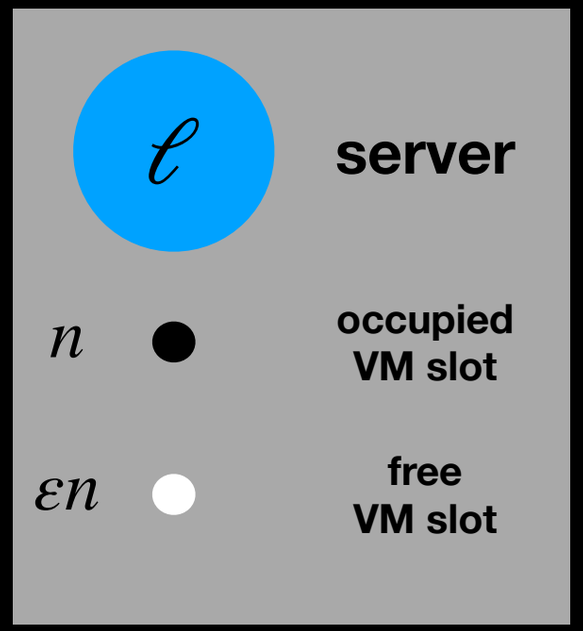




Re-locate VMs to avoid cross-server communication

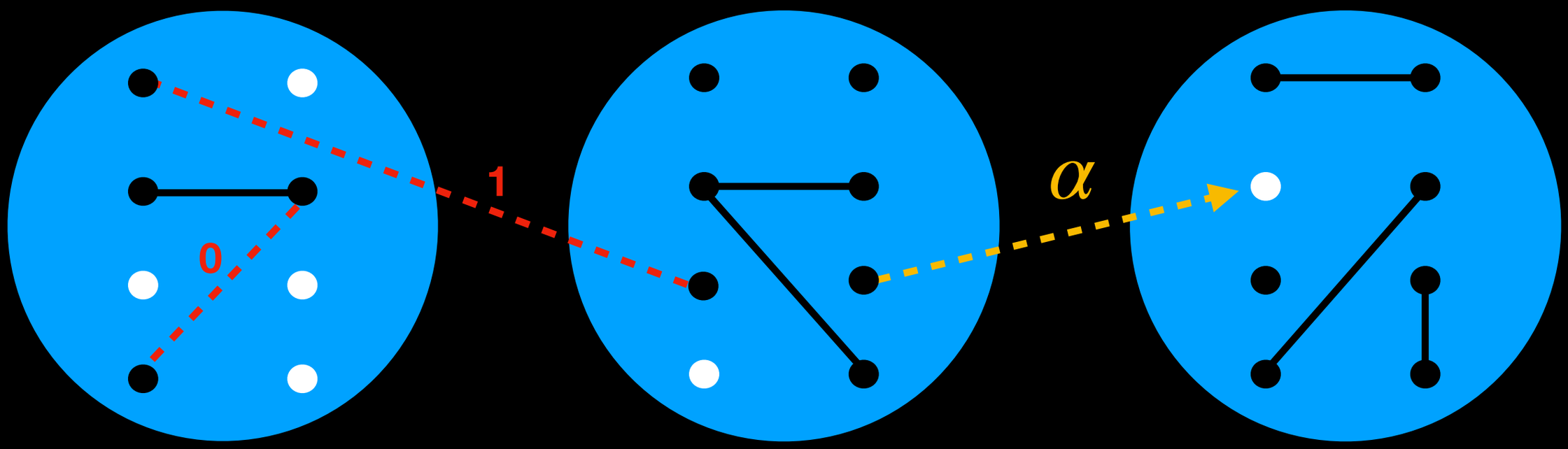
re-location cost
$\alpha > 1$

The Model



- Internal server communication cost: **0**
- Server-server communication cost: **1**
- VM re-location cost: α

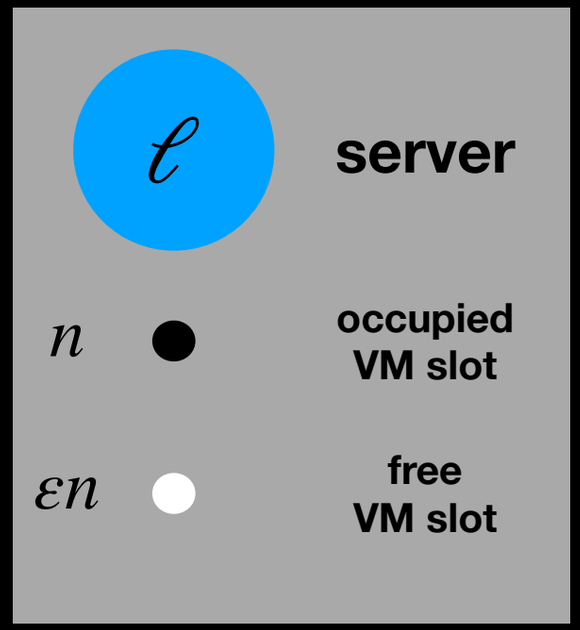
➡ Given an *online* sequence of communication requests,
minimize total cost paid for communication



The Model

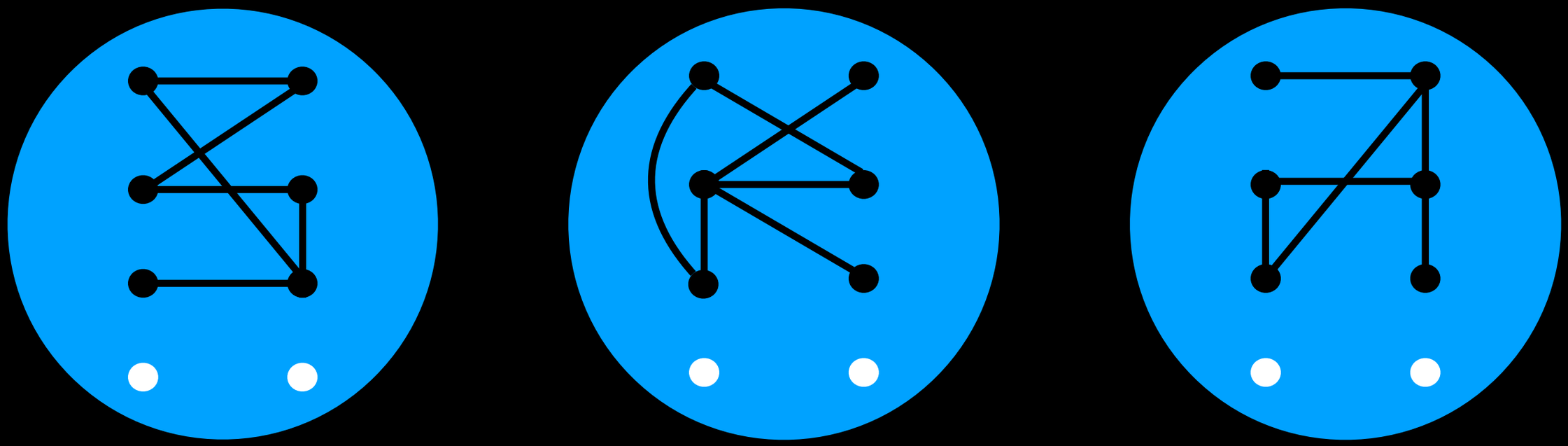
- Internal server communication cost: **0**
- Server-server communication cost: **1**
- VM re-location cost: α

➡ Given an *online* sequence of communication requests, minimize total cost paid for communication

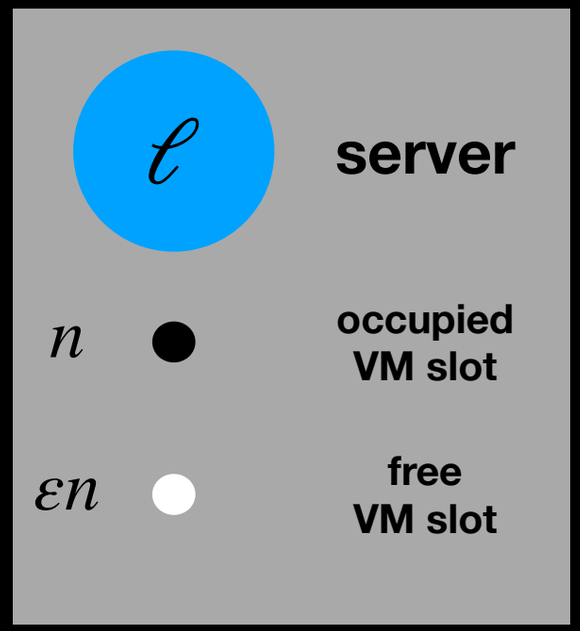


After all communications finished:

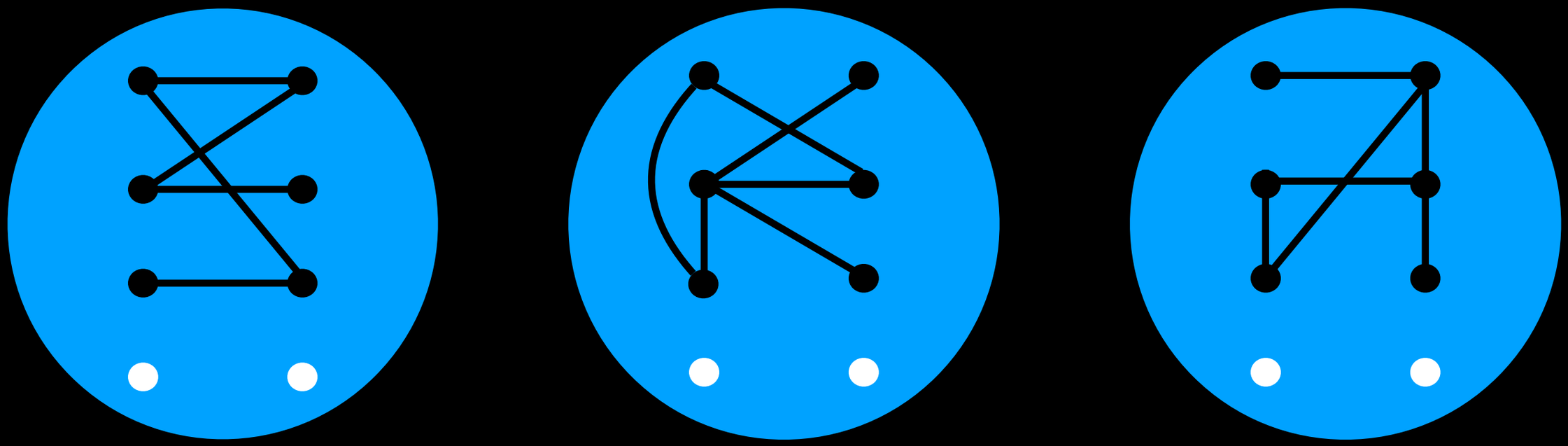
1 server
=
1 component



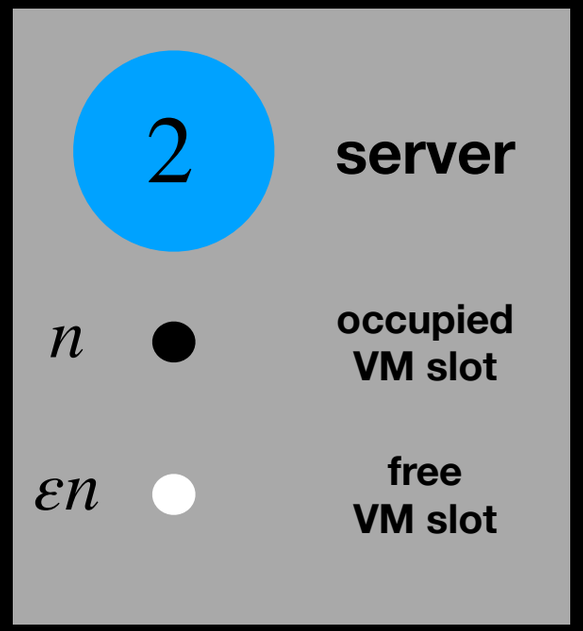
Analysis



- Competitive analysis comparing to OPT:
  - OPT knows all communications in advance
  - OPT computes solution with optimal cost
- (Strict) competitive ratio = $\frac{\text{ALG}}{\text{OPT}}$

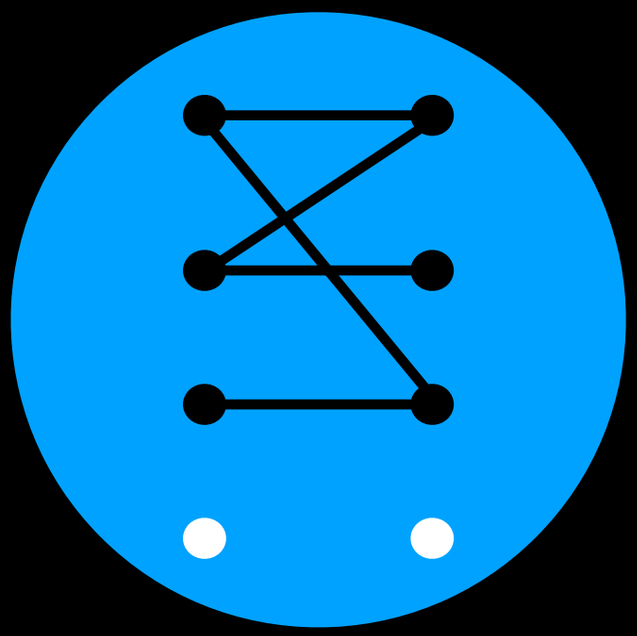


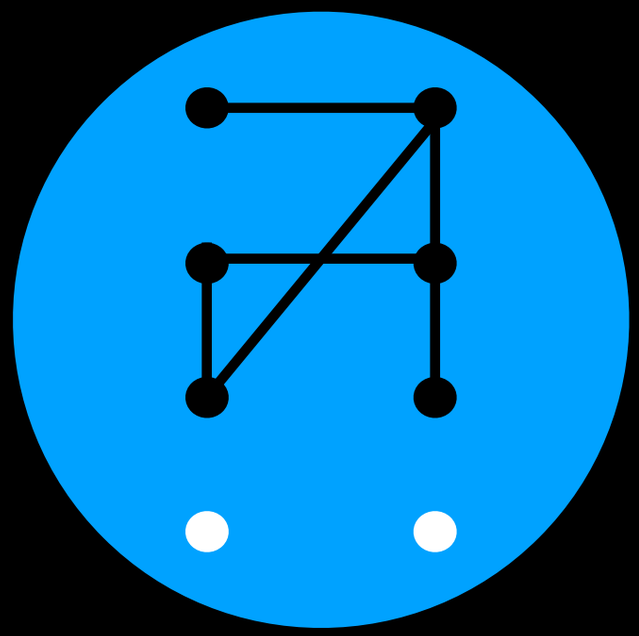
Results



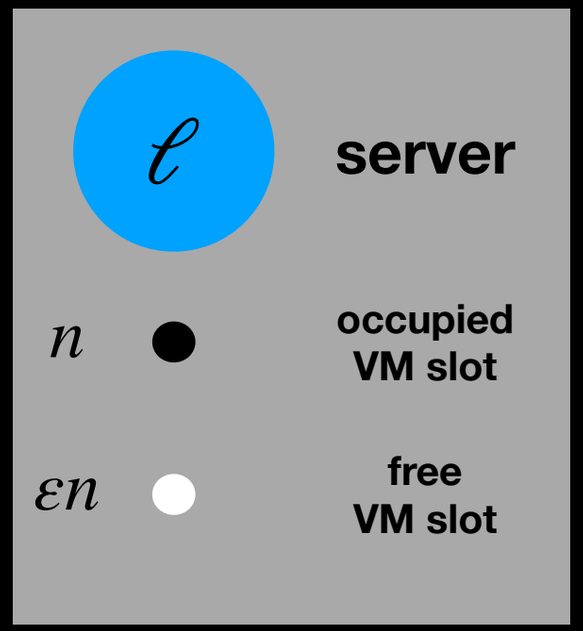
- For $\ell = 2$ servers:
  - Algorithm which is $O\left(\frac{\log n}{\varepsilon}\right)$-competitive
  - Lower bound: Any algorithm must be $\Omega(1/\varepsilon + \log n)$-competitive

➡ Our results are almost tight for two servers





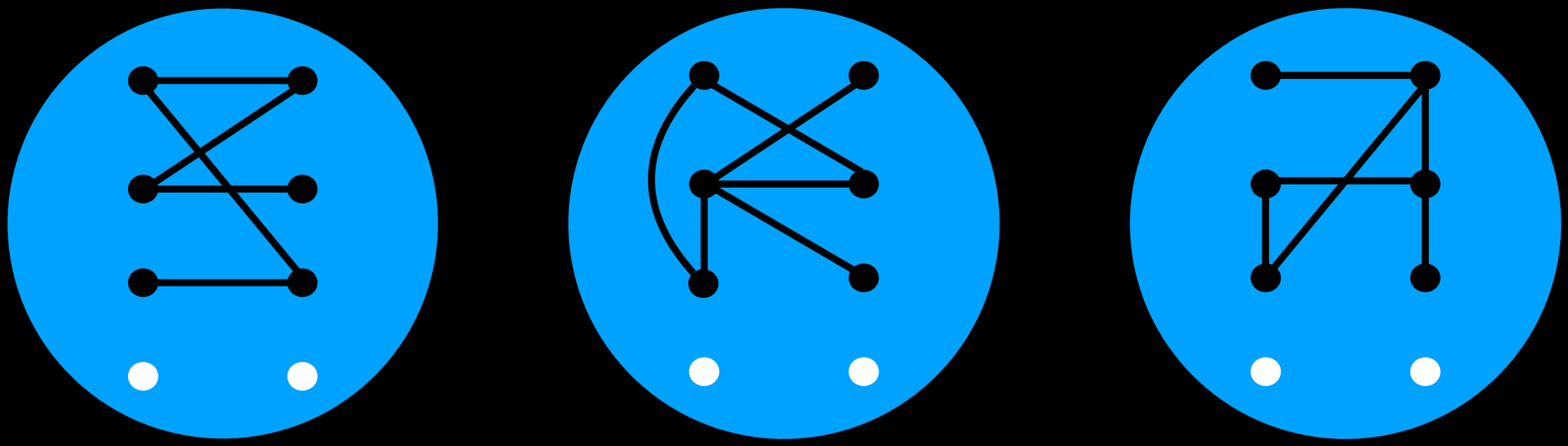
Results



- For ℓ servers:
  - Algorithm which is $O\left((\ell \log n \log \ell)/\varepsilon\right)$-competitive

➡ Efficient when ℓ is small,
e.g., for communication across data centers

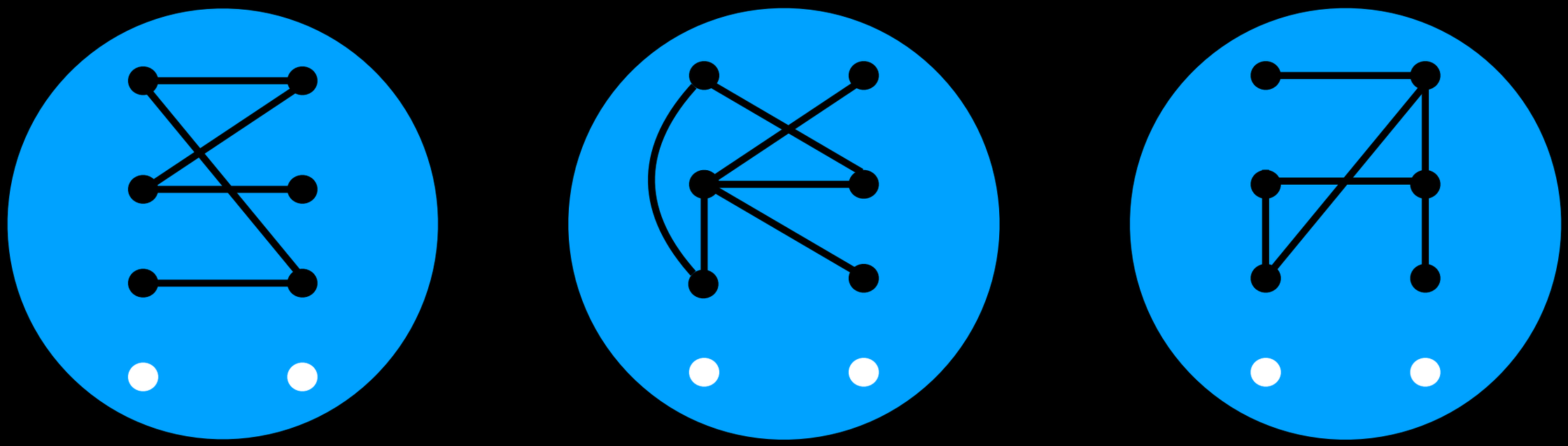
➡ Implementable for distributed computation
communication cost ≤ communication for re-locating VMs (if $\ell = O\left(\sqrt{\varepsilon n}\right)$)



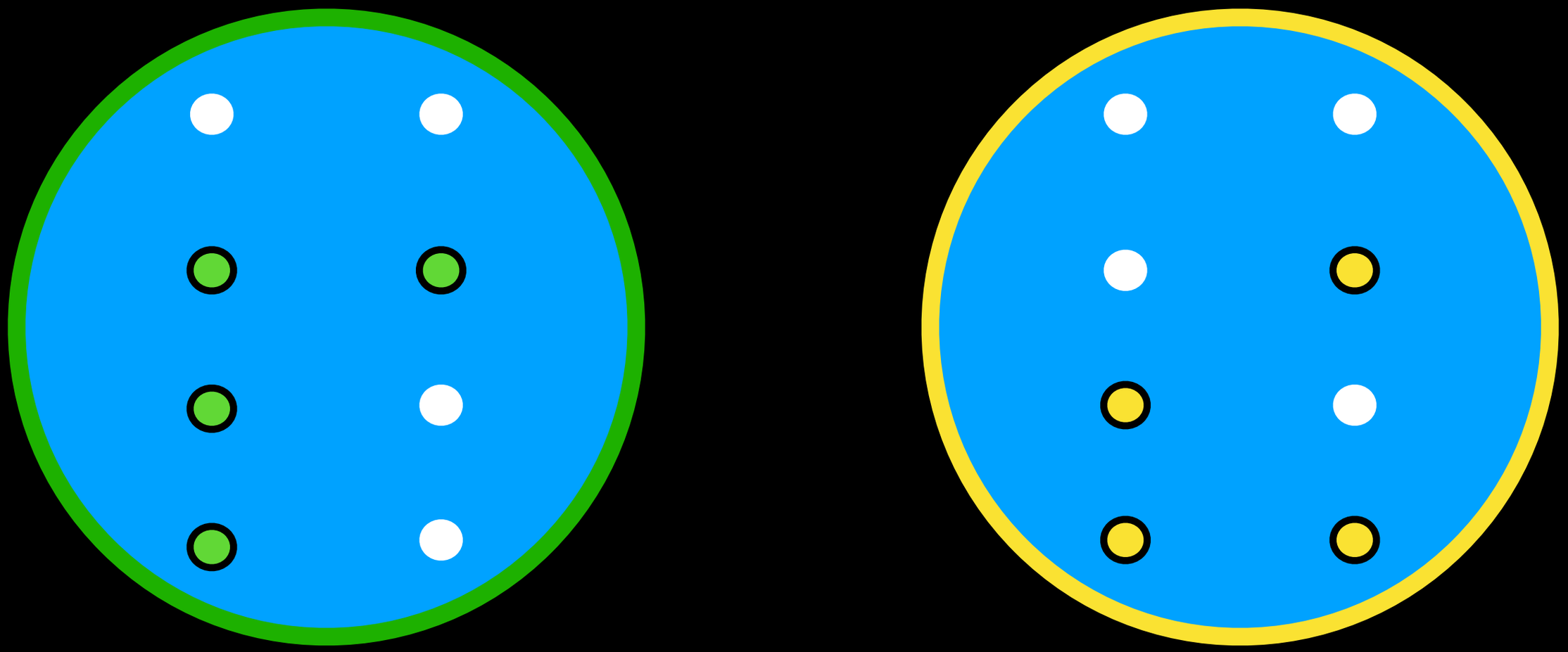
Applications

- Distributed **Union Find** Data Structure
  (with small cost for re-locating the sets across servers)

- Online Balanced **k-way Partition**
  (with small cost for re-assigning numbers to balanced partitions)

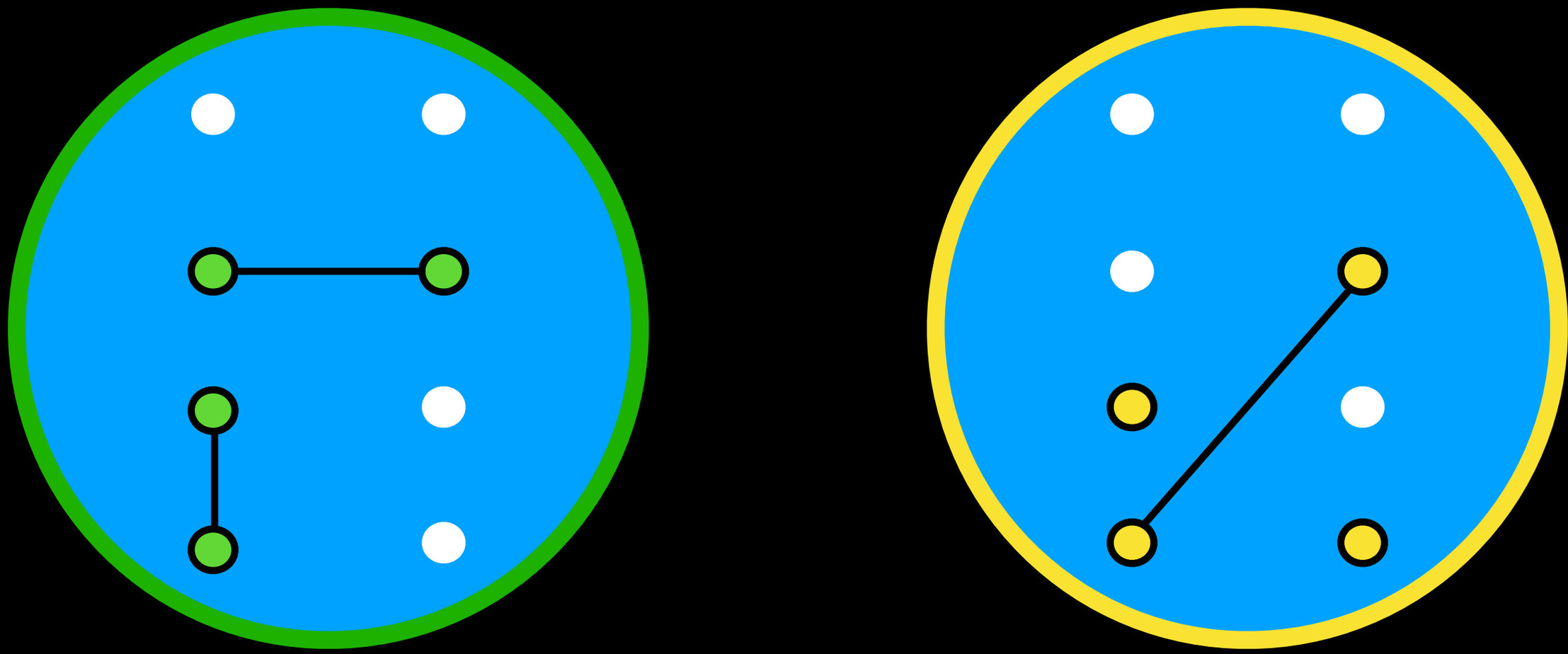


Algorithm for Two Servers

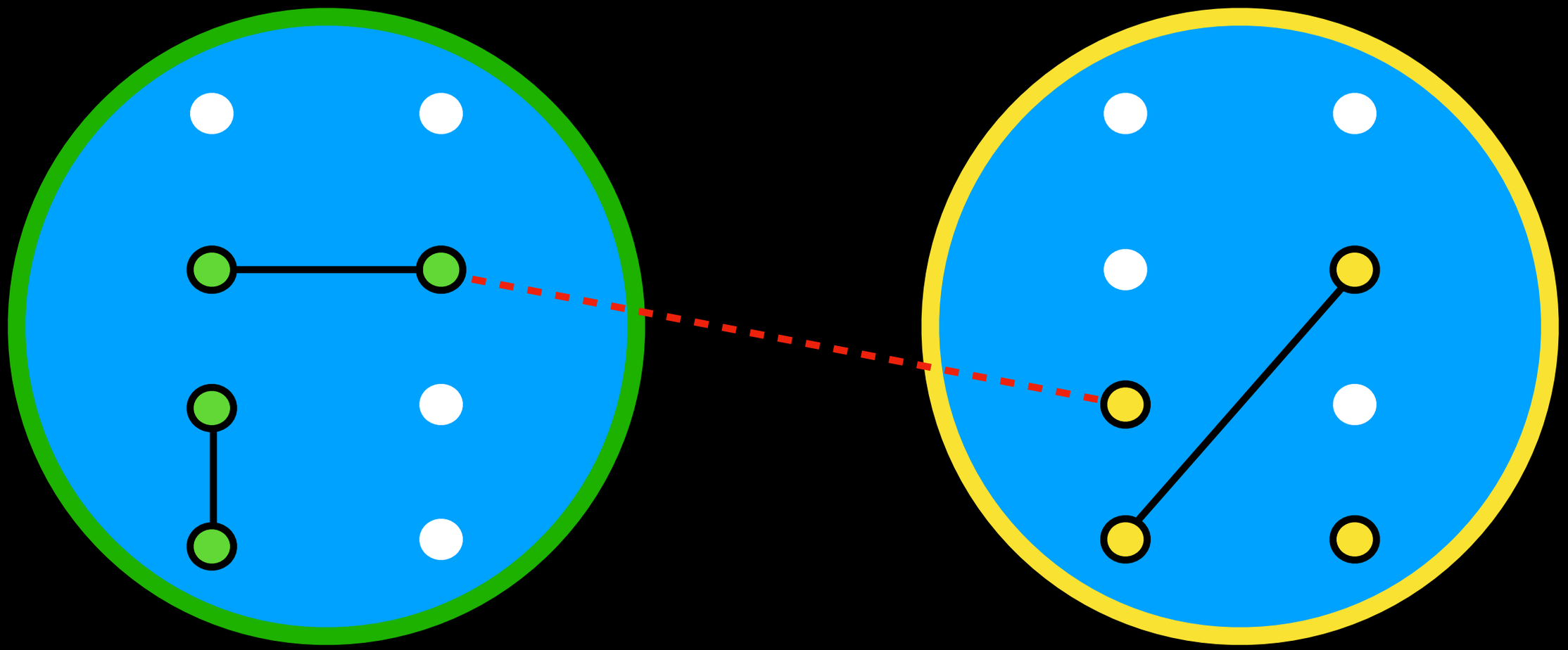


Color each VM based on its initial server

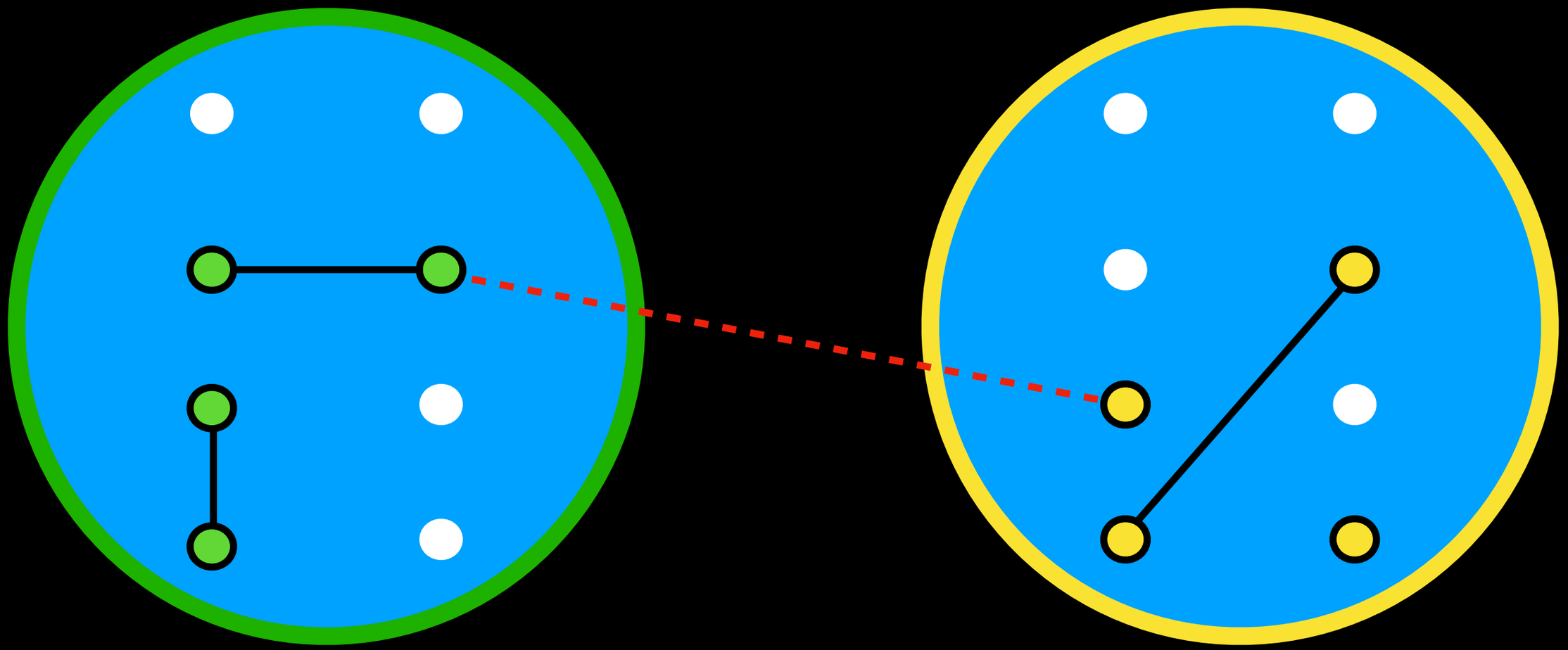
Algorithm for Two Servers



Algorithm for Two Servers

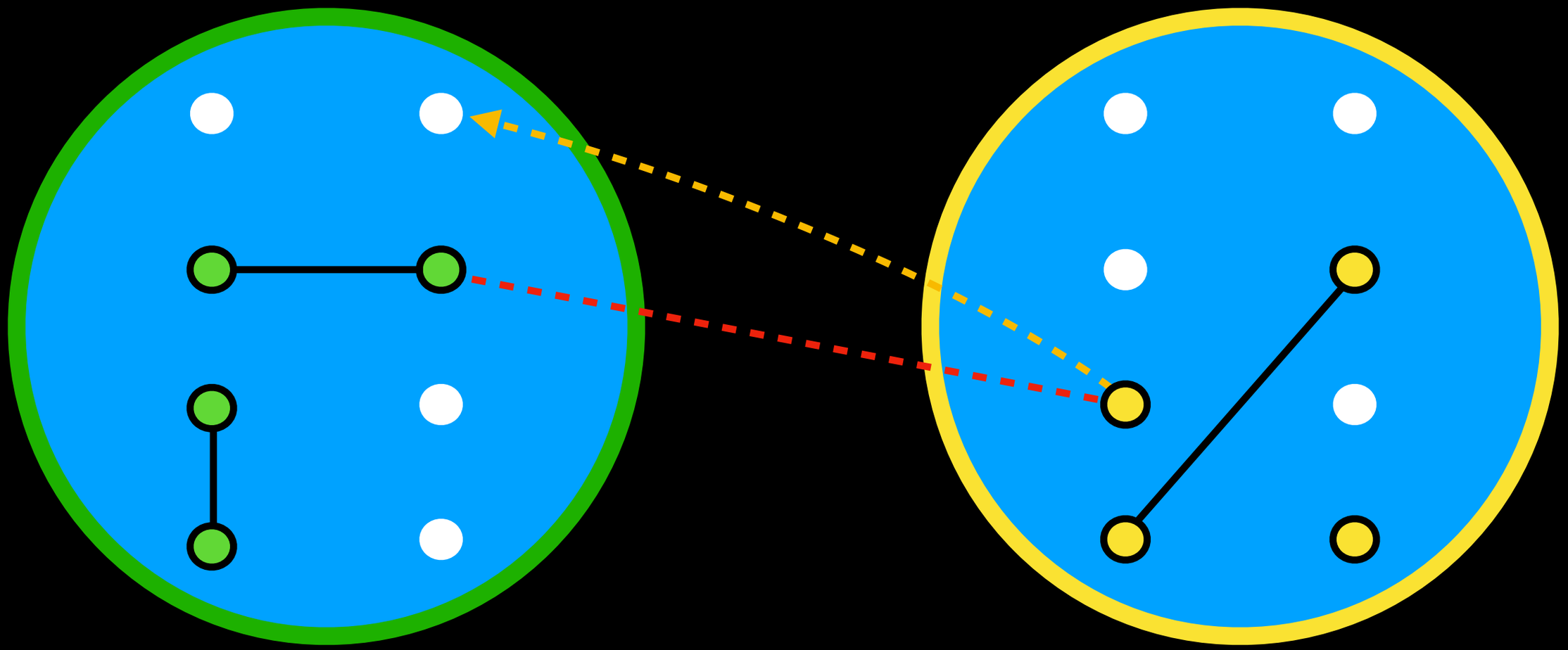


Algorithm for Two Servers



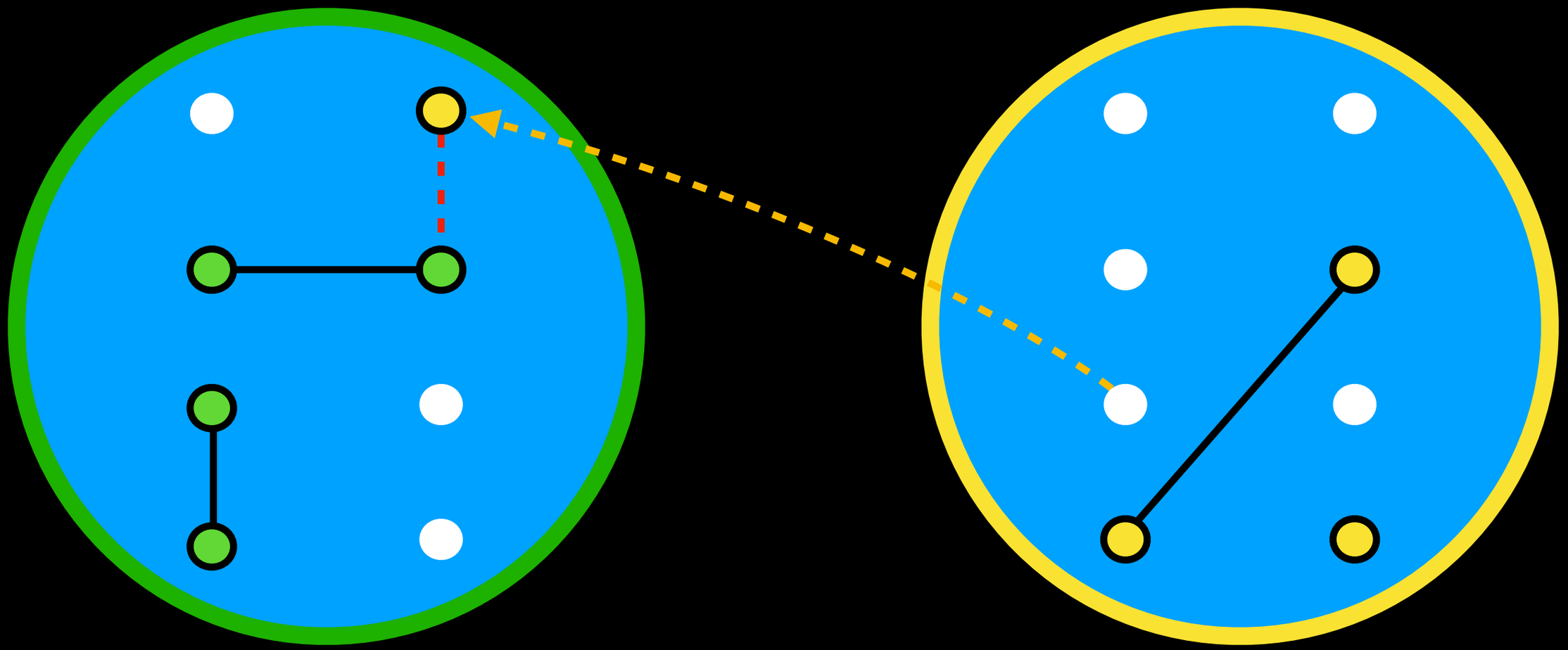
Move small component to larger one

Algorithm for Two Servers



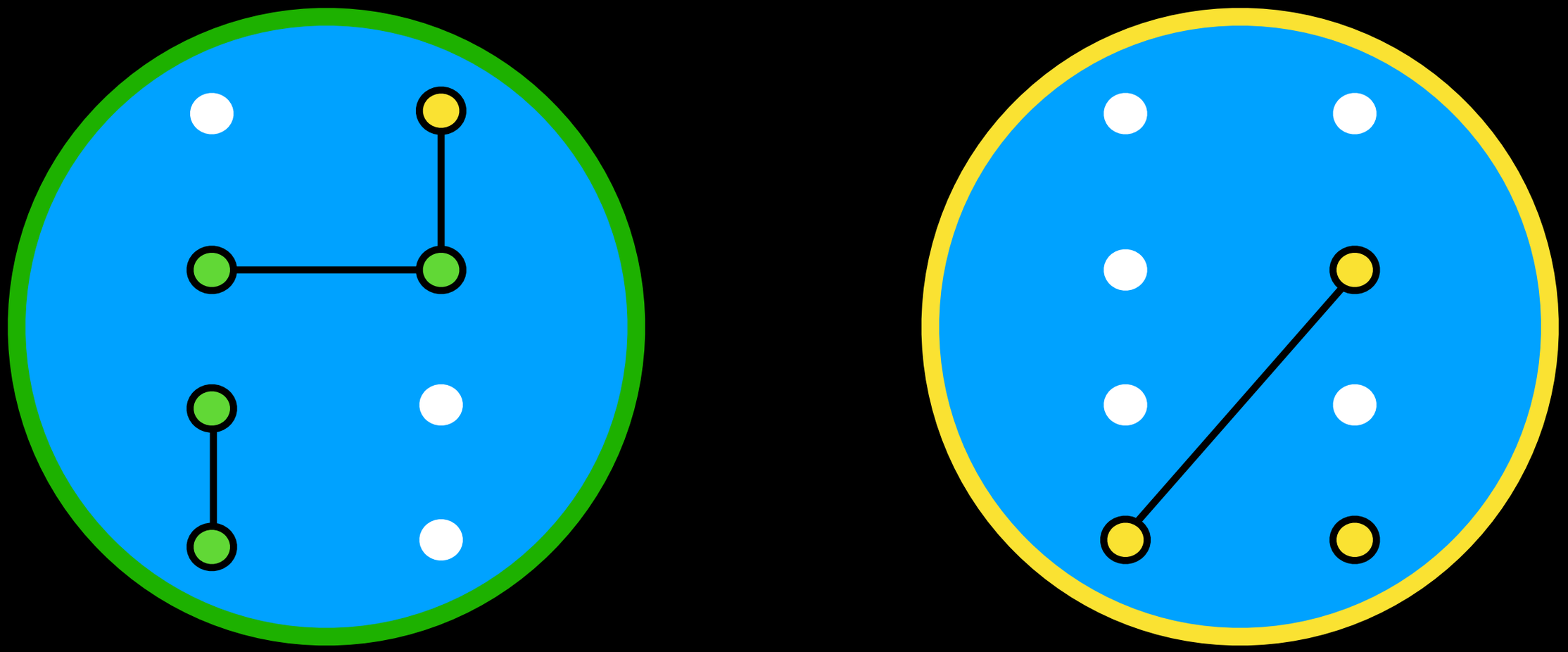
Move small component to larger one

Algorithm for Two Servers



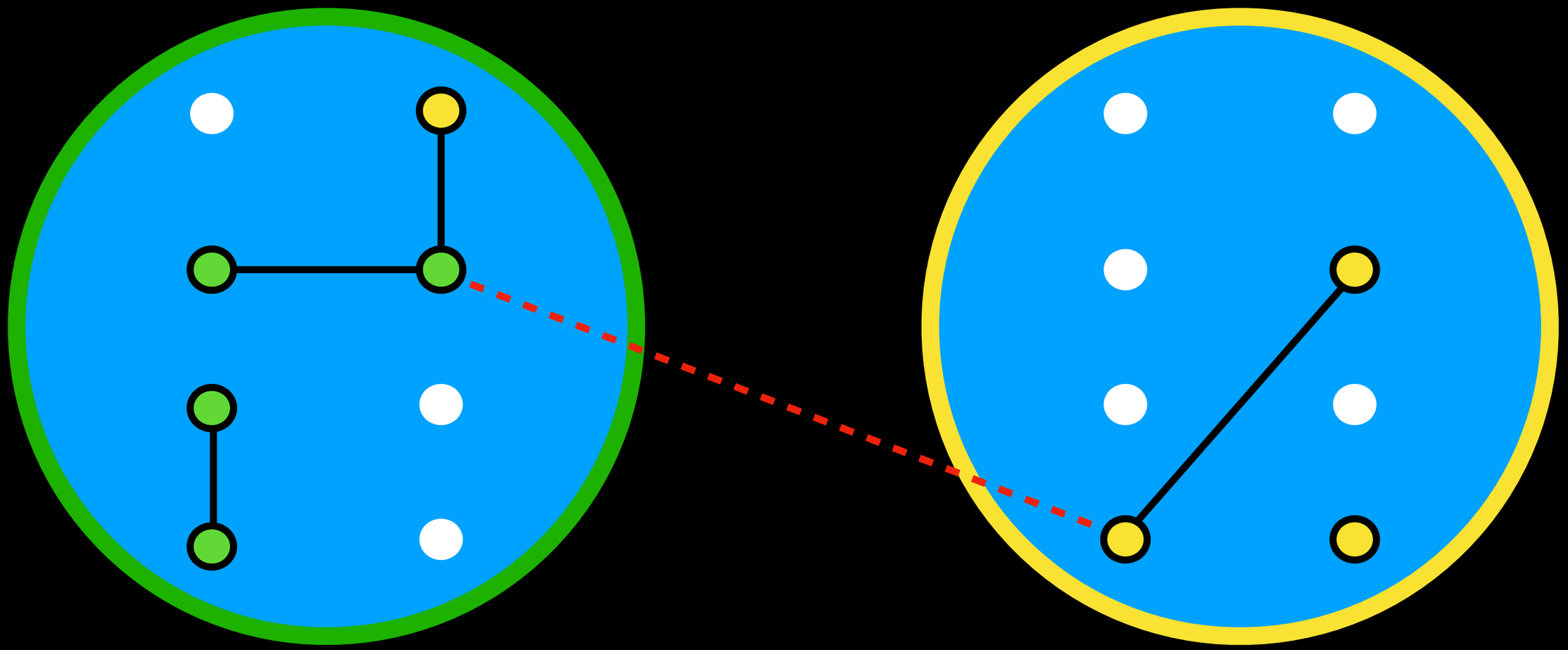
Move small component to larger one

Algorithm for Two Servers



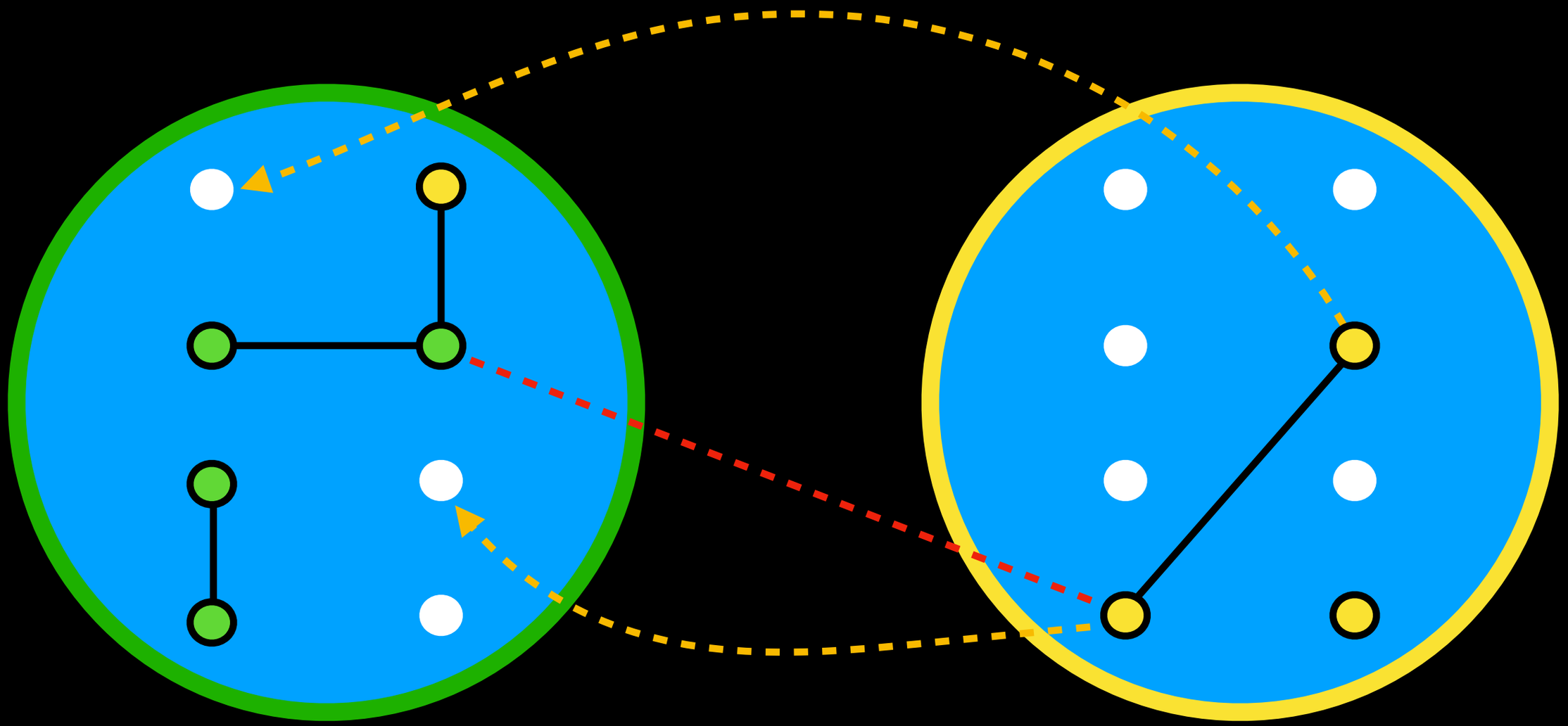
Move small component to larger one

Algorithm for Two Servers



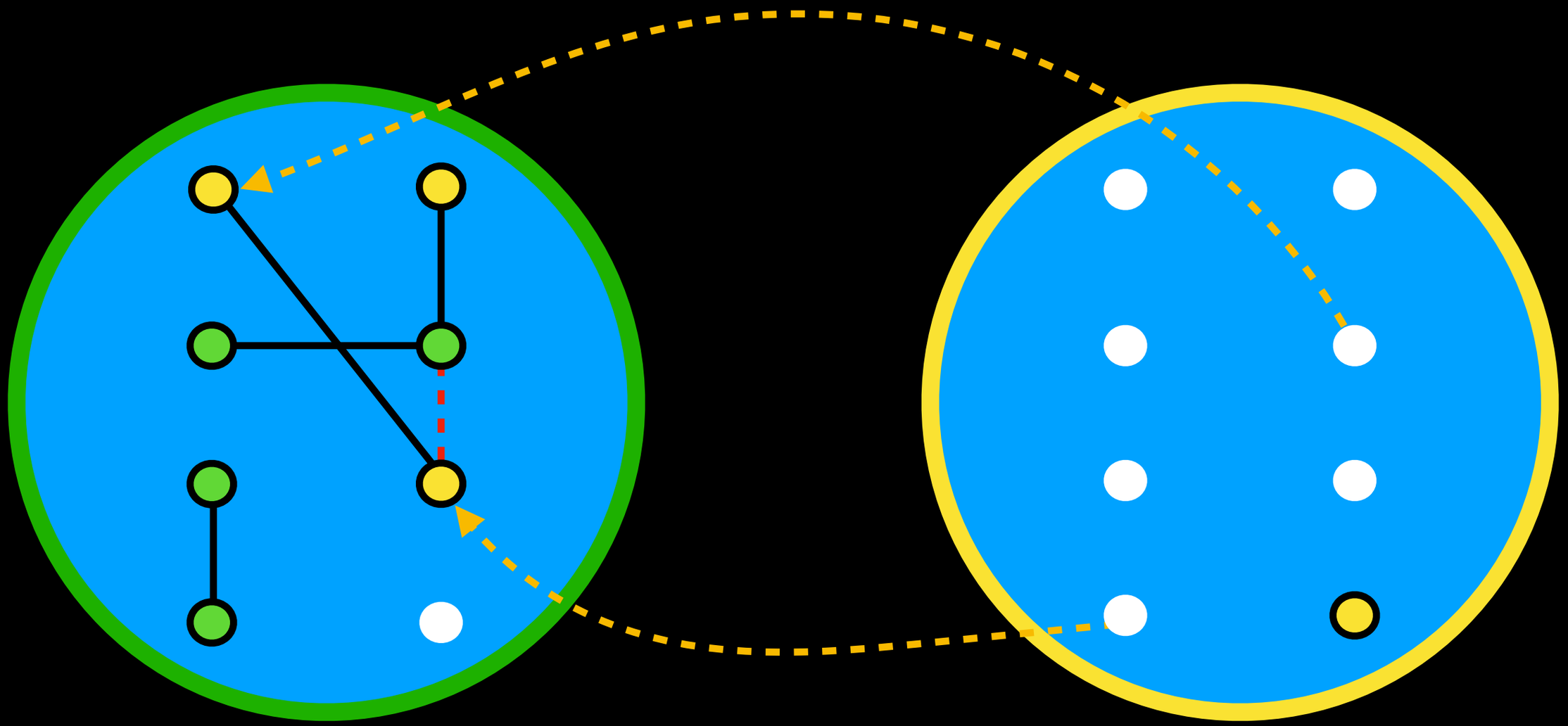
Move small component to larger one

Algorithm for Two Servers



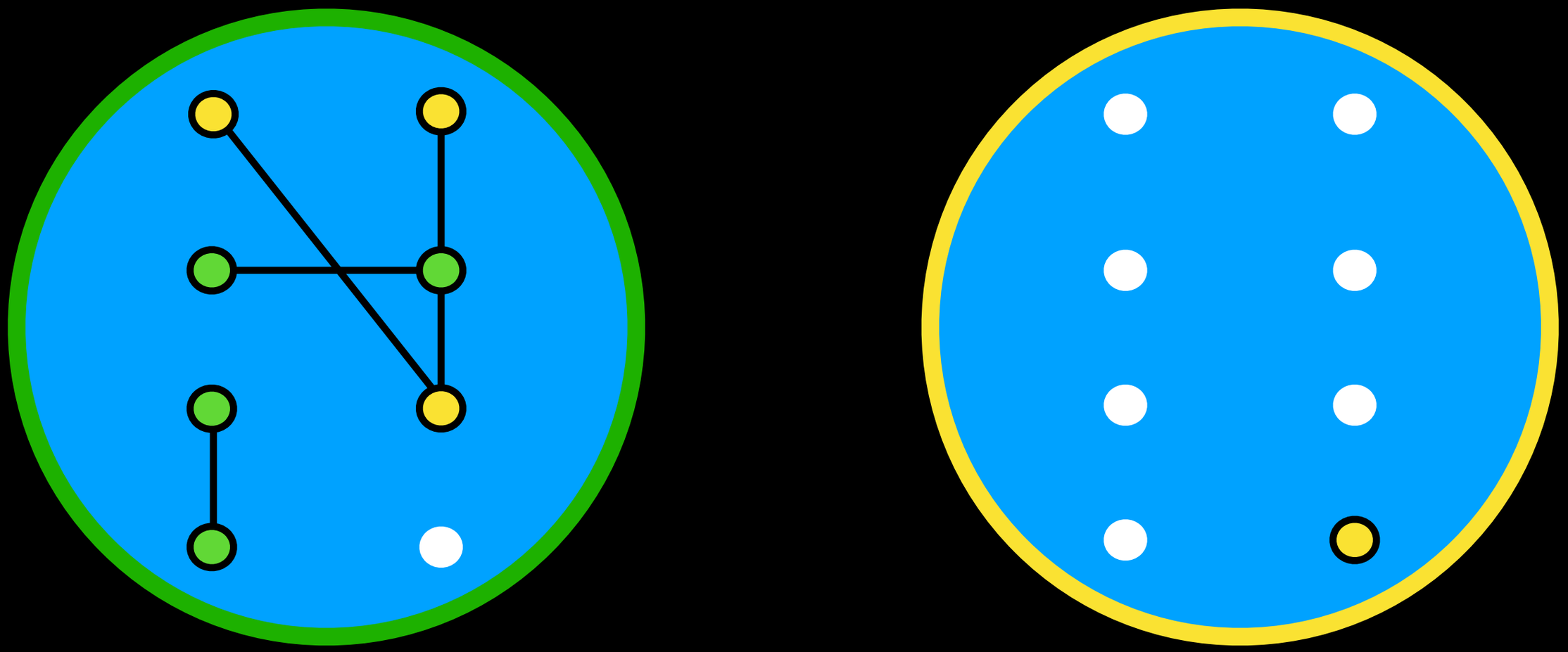
Move small component to larger one

Algorithm for Two Servers



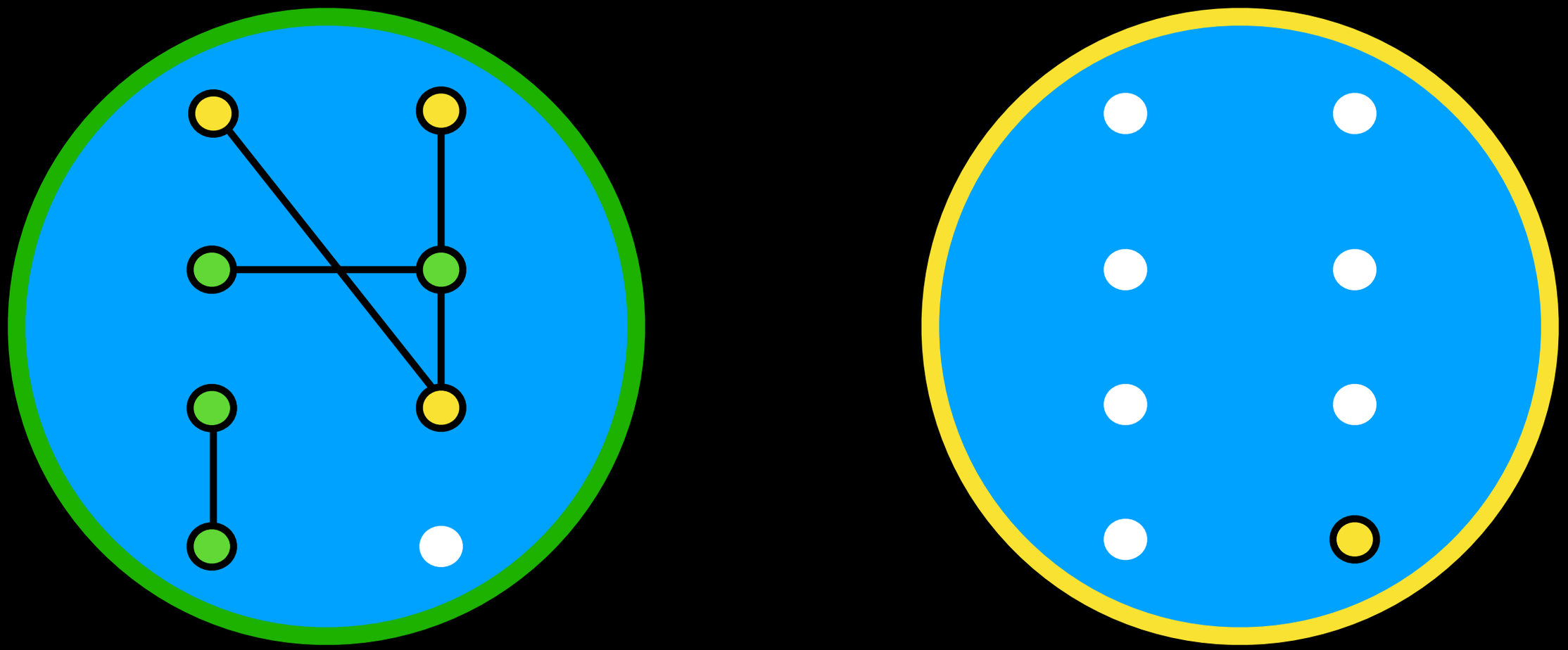
Move small component to larger one

Algorithm for Two Servers

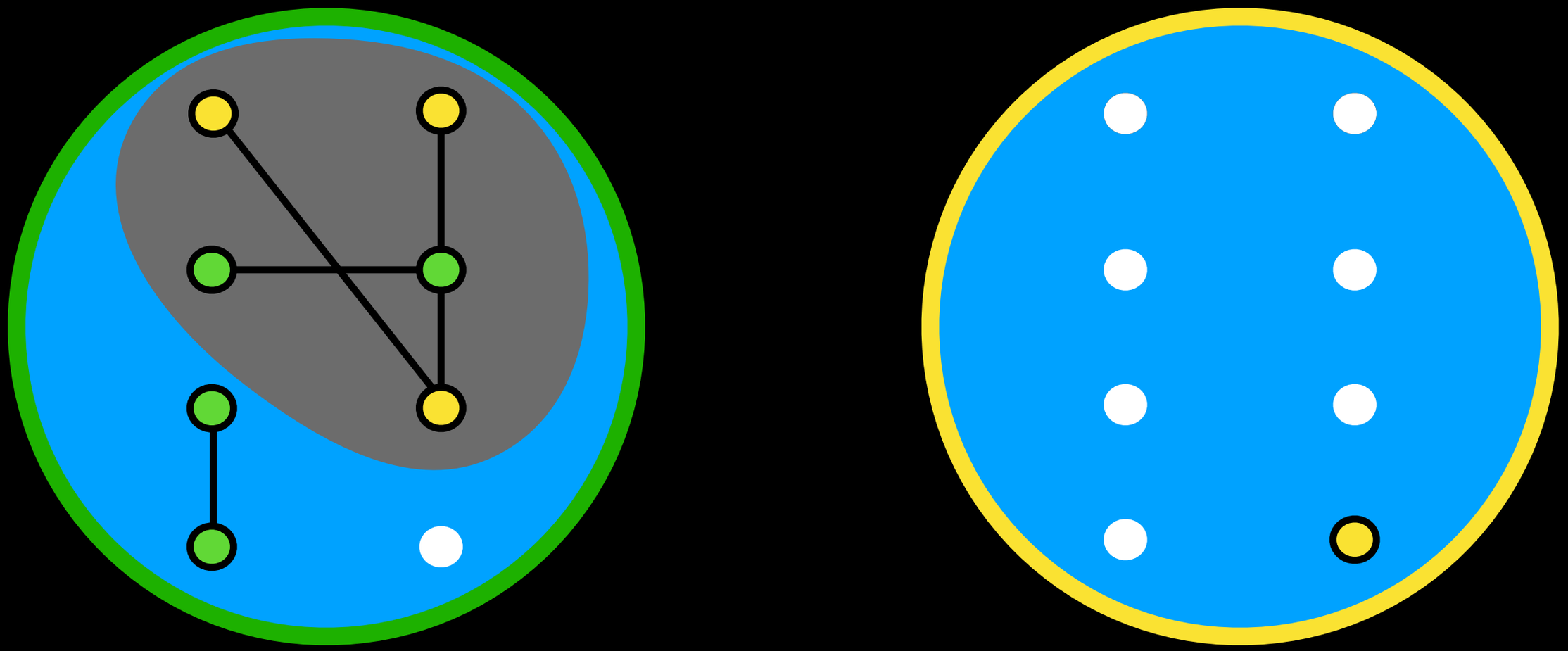


Move small component to larger one

Algorithm for Two Servers

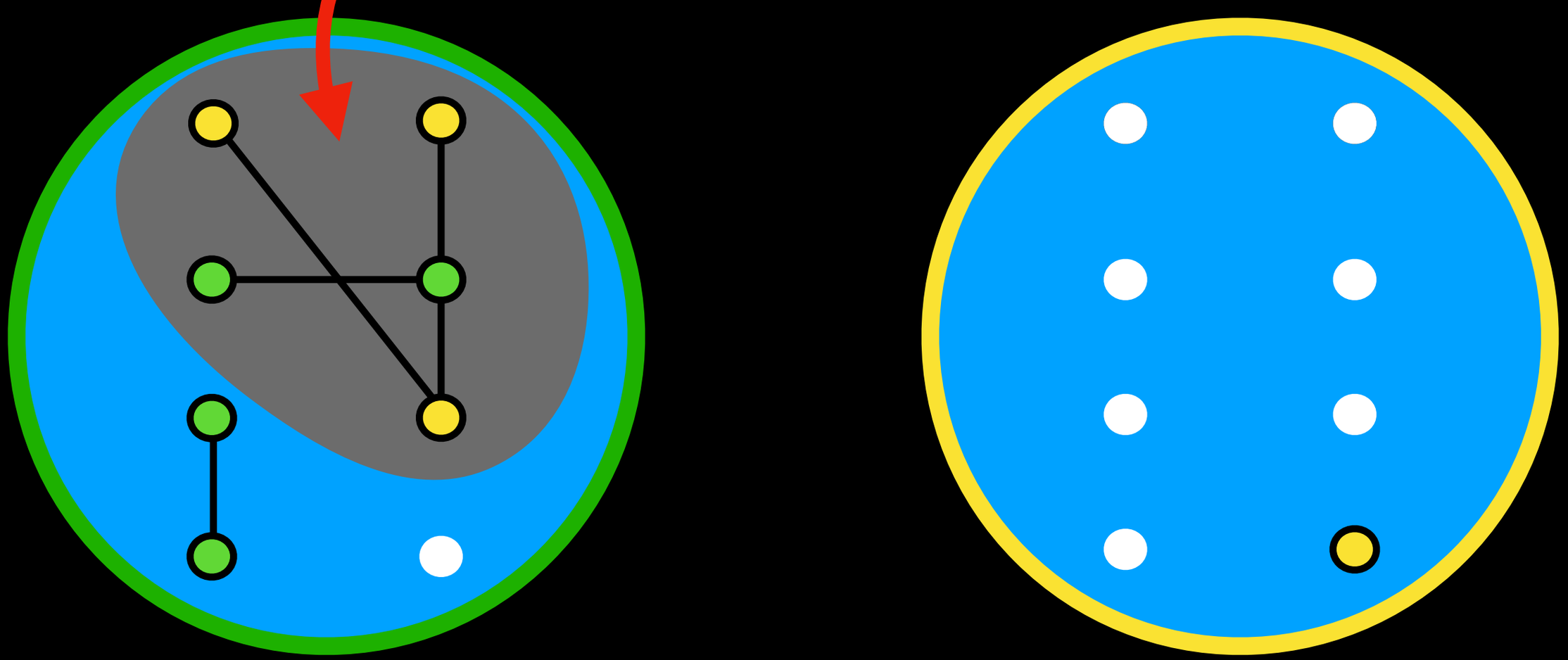


Algorithm for Two Servers



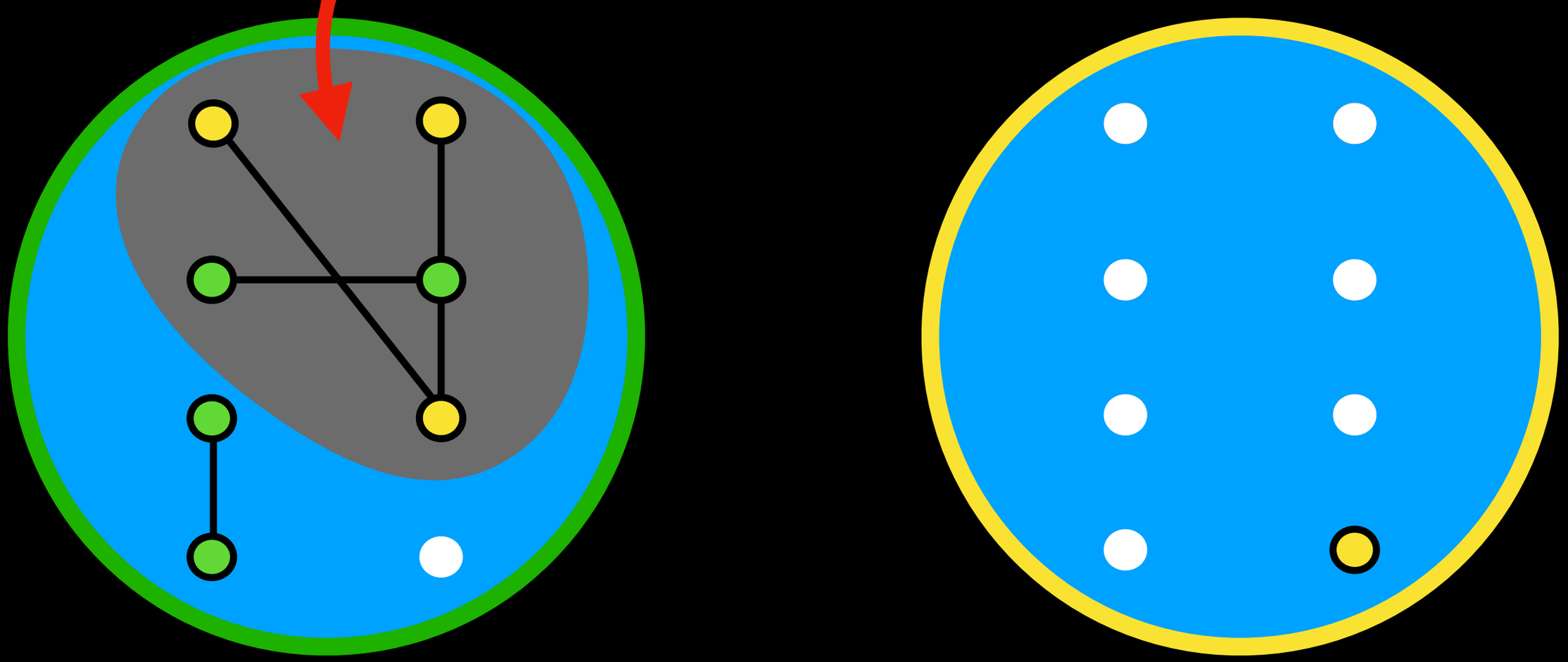
Algorithm for Two Servers

Contains more
yellow than green VMs



Algorithm for Two Servers

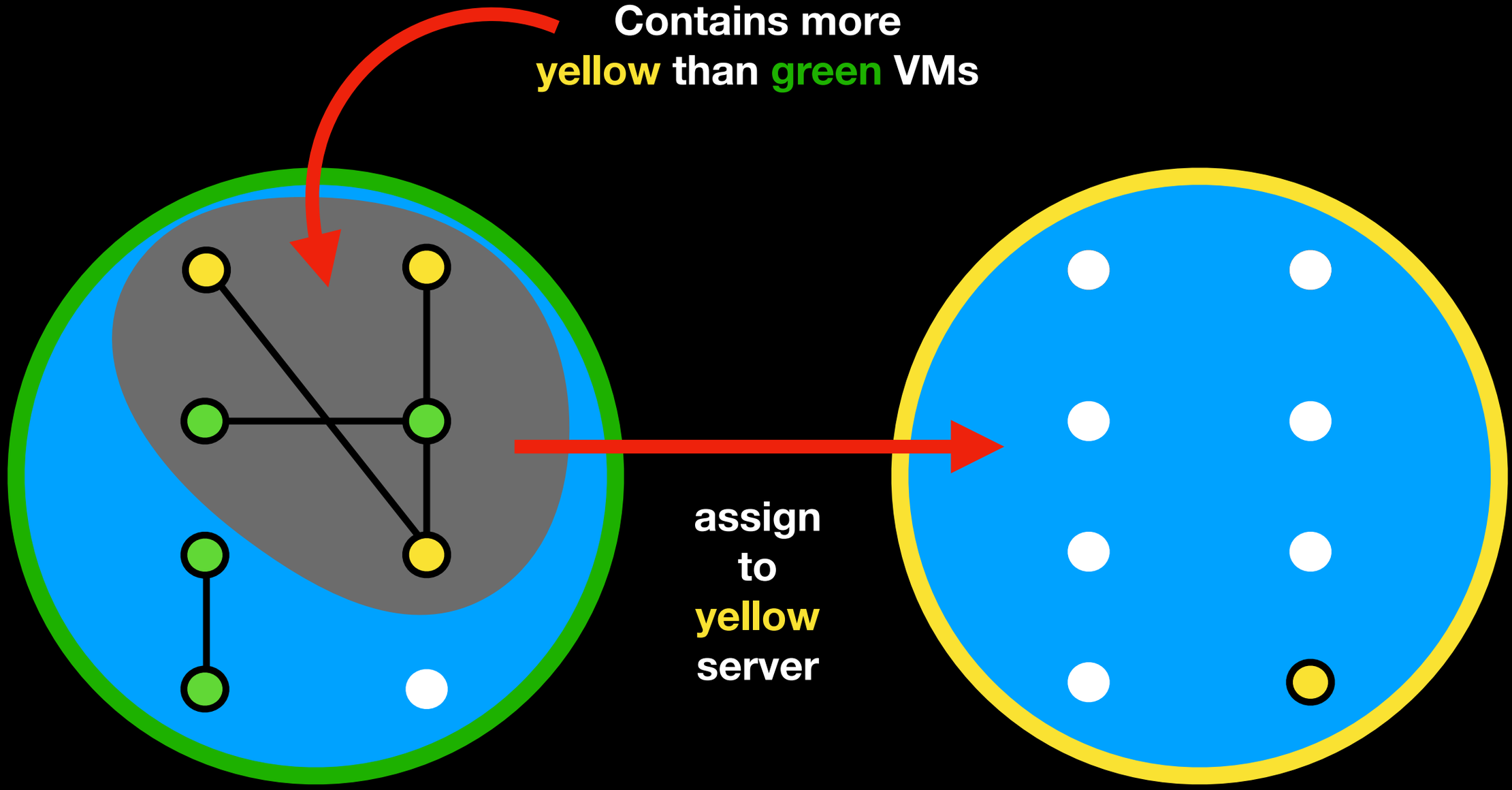
Contains more
yellow than green VMs



Majority-voting step

Algorithm for Two Servers

Contains more
yellow than green VMs

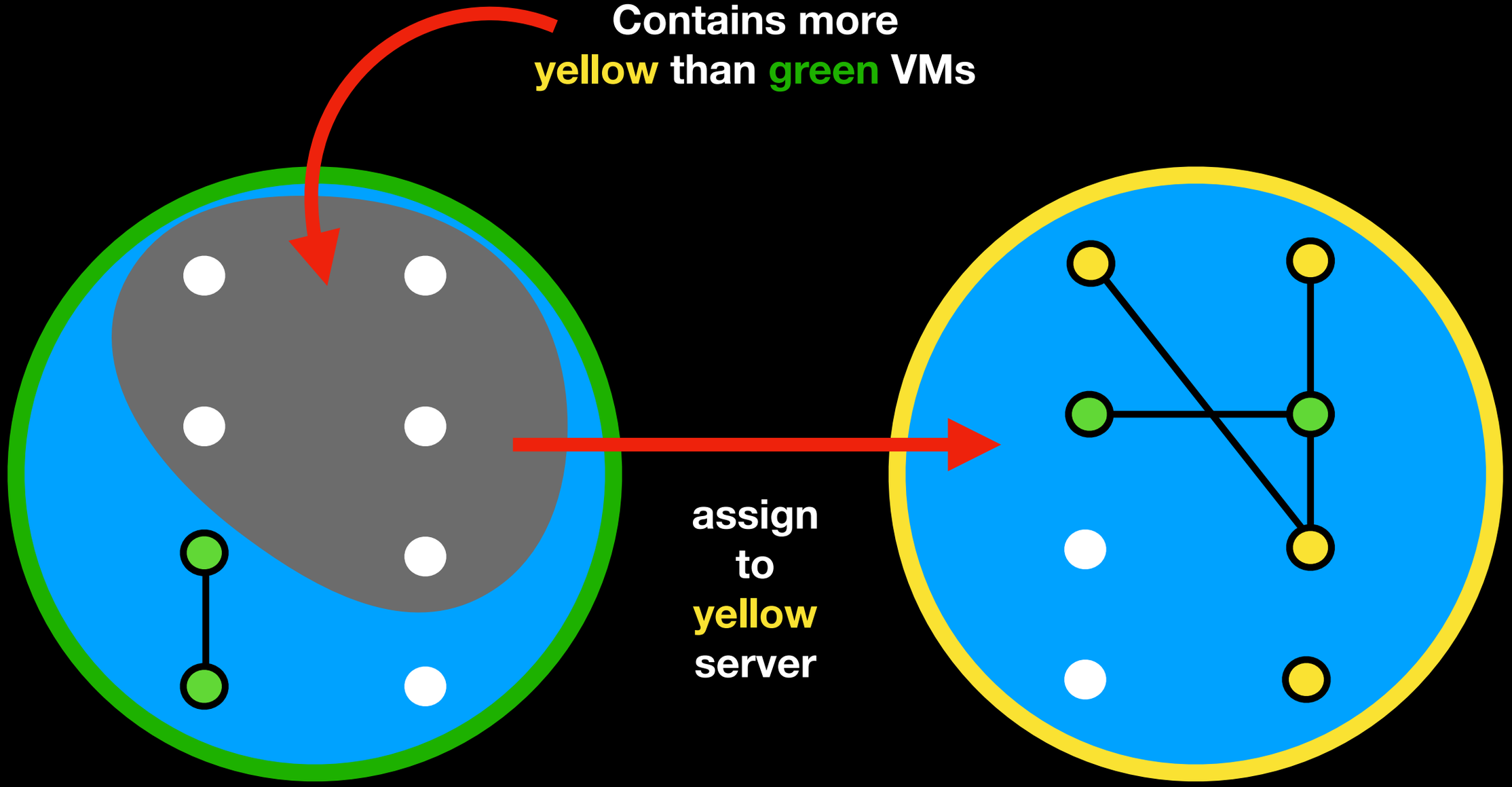


assign
to
yellow
server

Majority-voting step

Algorithm for Two Servers

Contains more
yellow than green VMs

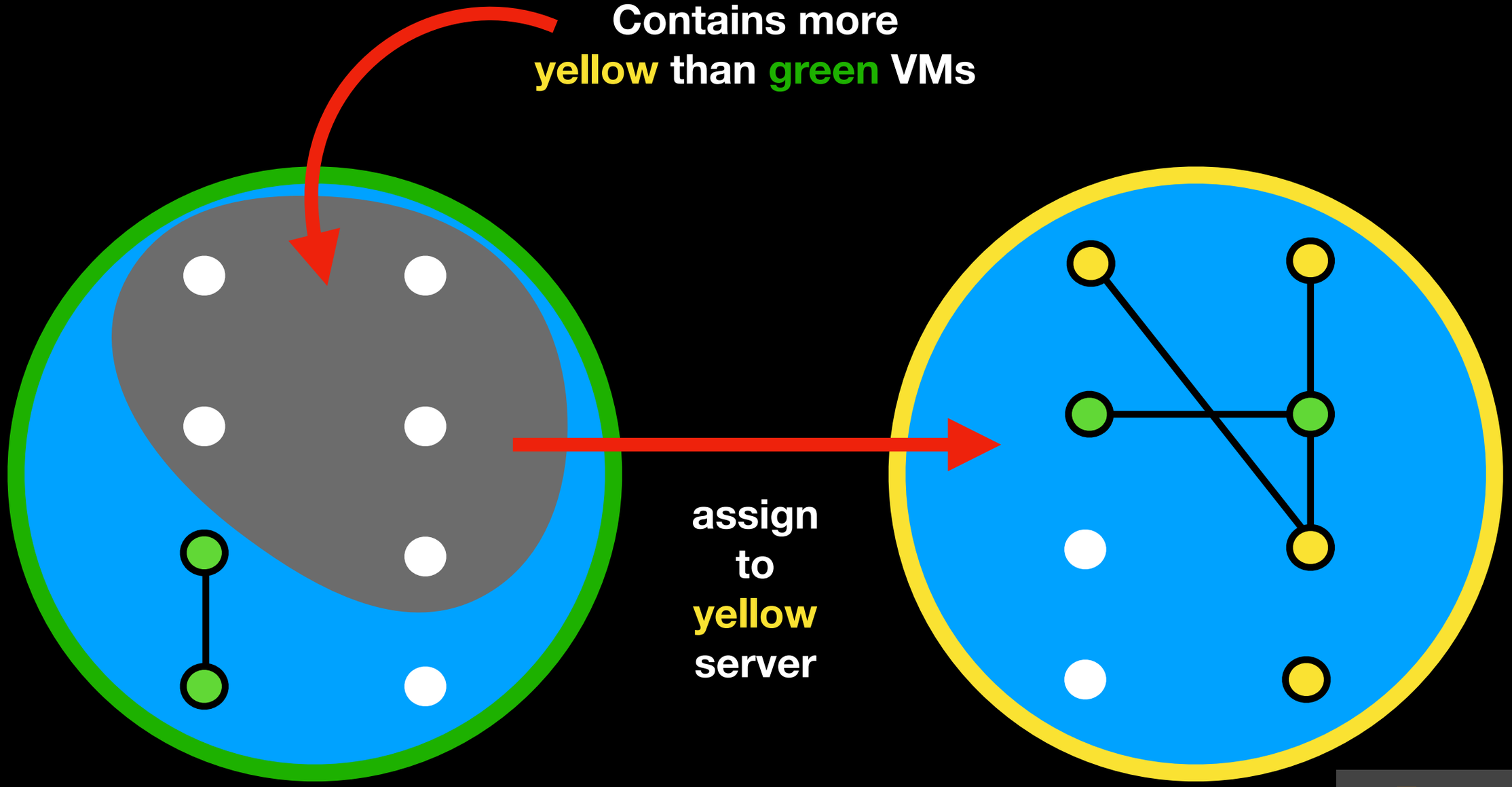


assign
to
yellow
server

Majority-voting step

Algorithm for Two Servers

Contains more
yellow than green VMs



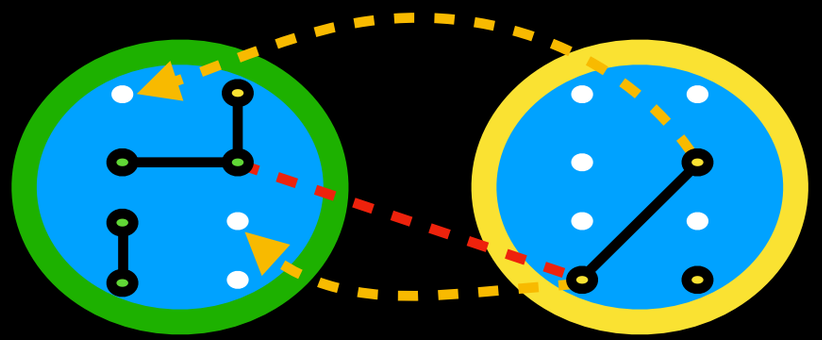
assign
to
yellow
server

Majority-voting step

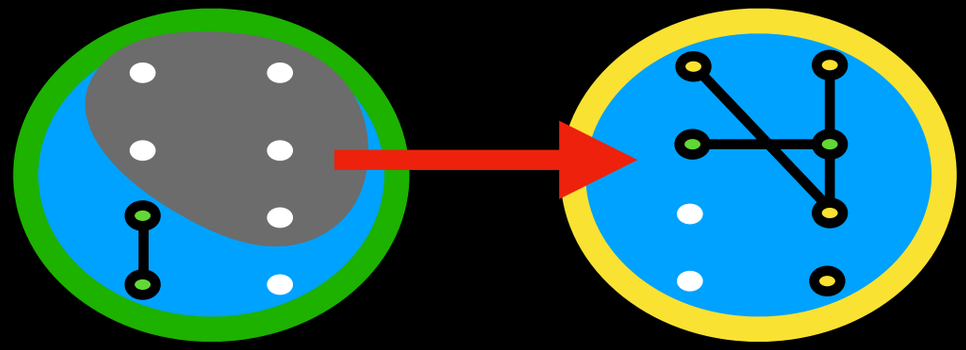
Ensures
that we stay
close to initial
assignment

For each new communication request:

- Move smaller component to the server of the larger one



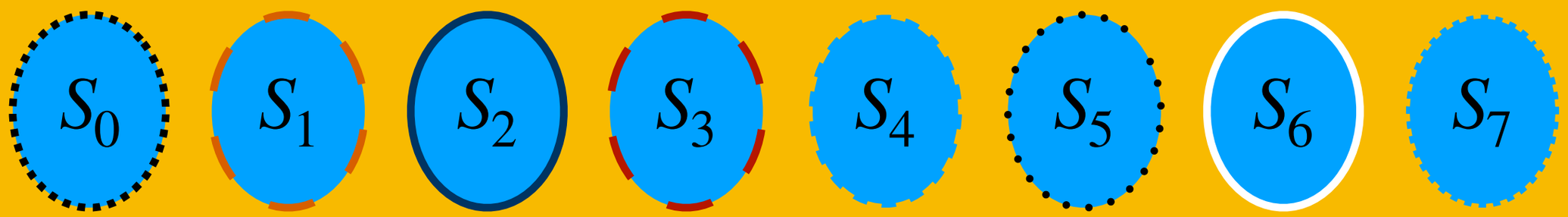
- *If size of new component exceeds a power of 2:*
  Perform majority-voting step



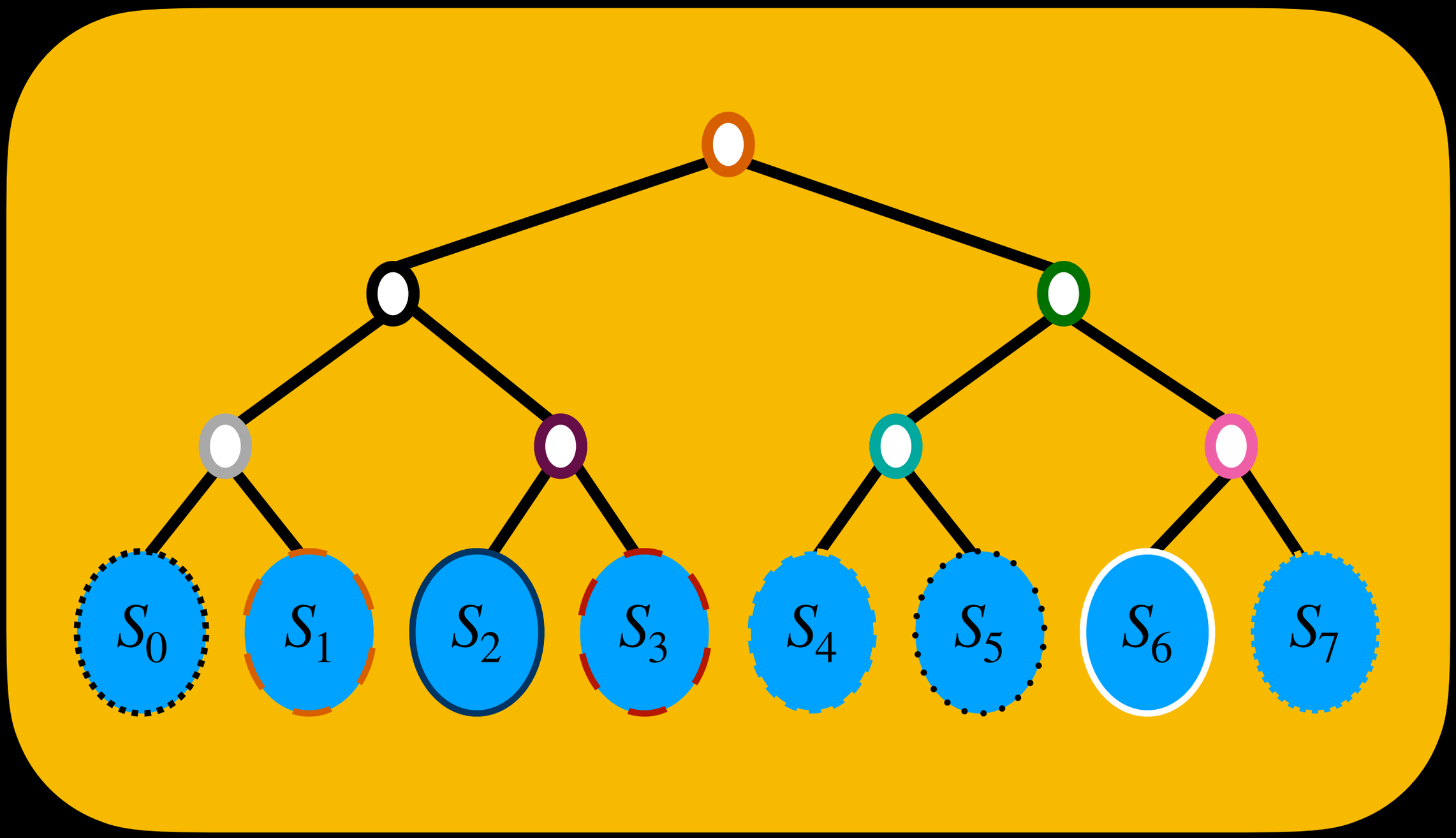
- *If server capacity exceeded:*
  Find cheapest balanced assignment using brute-force enumeration

Can only happen $O\left(\frac{\log n}{\varepsilon}\right)$ **times**

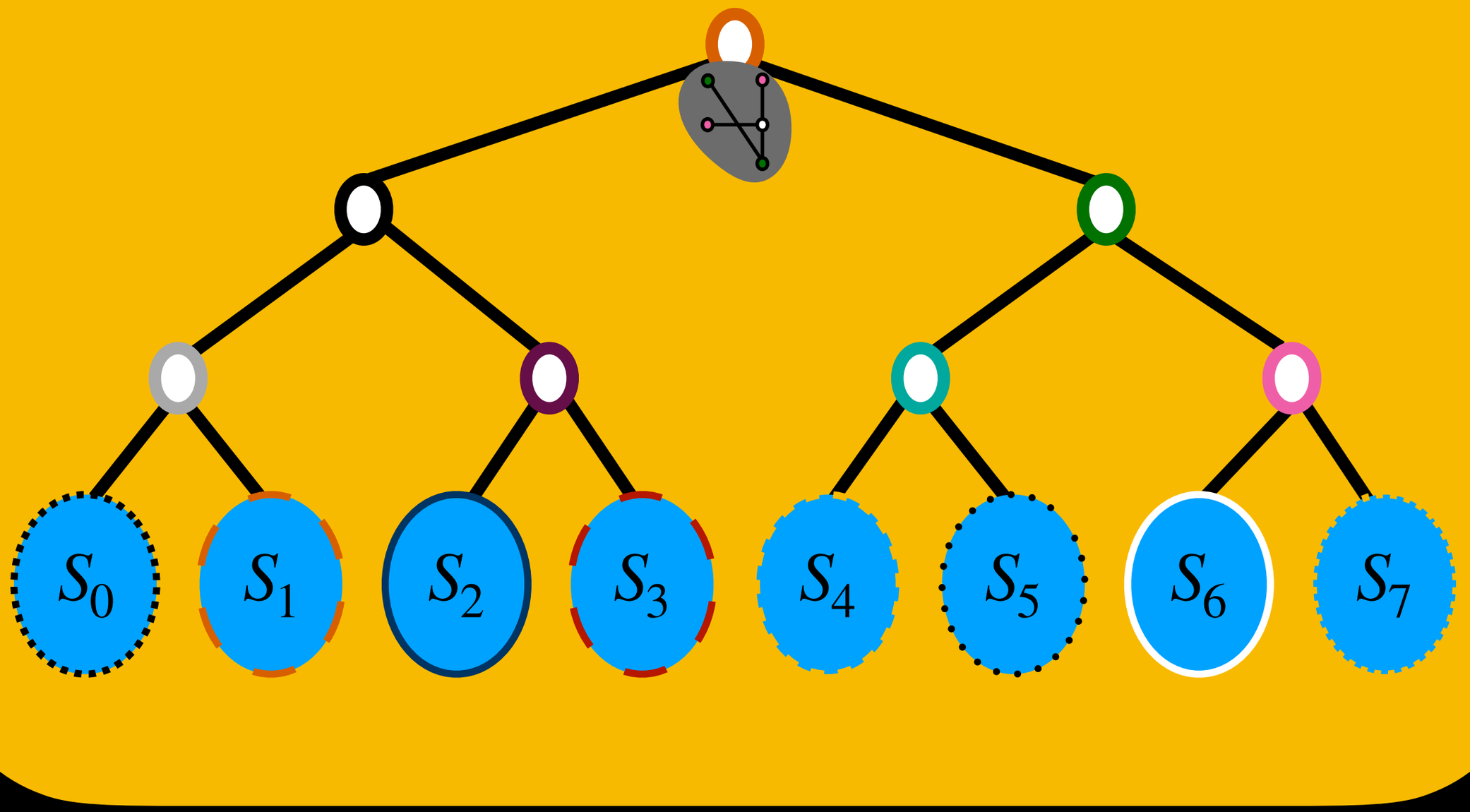
Generalization to ℓ Servers



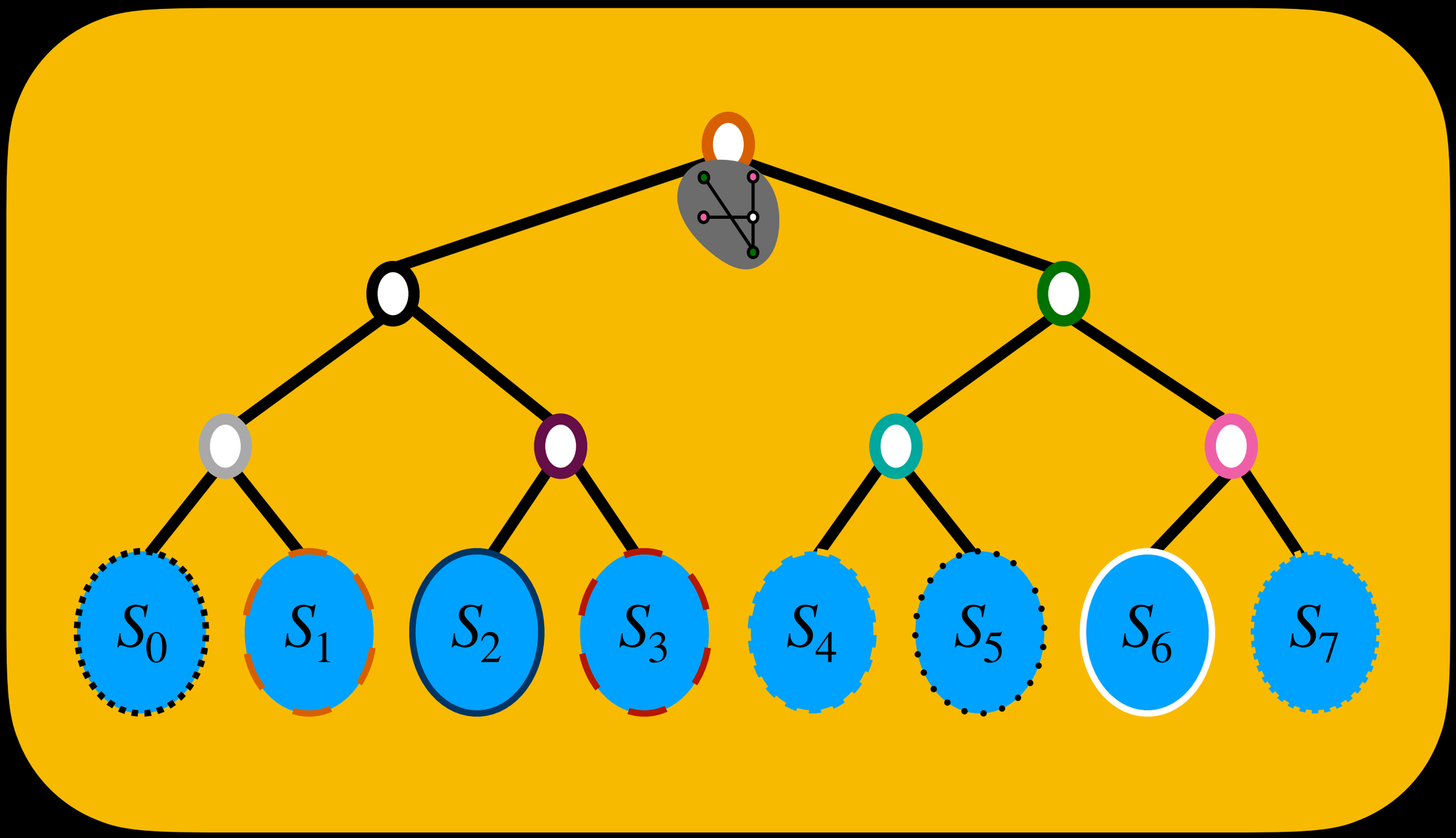
Generalization to ℓ Servers



Generalization to ℓ Servers

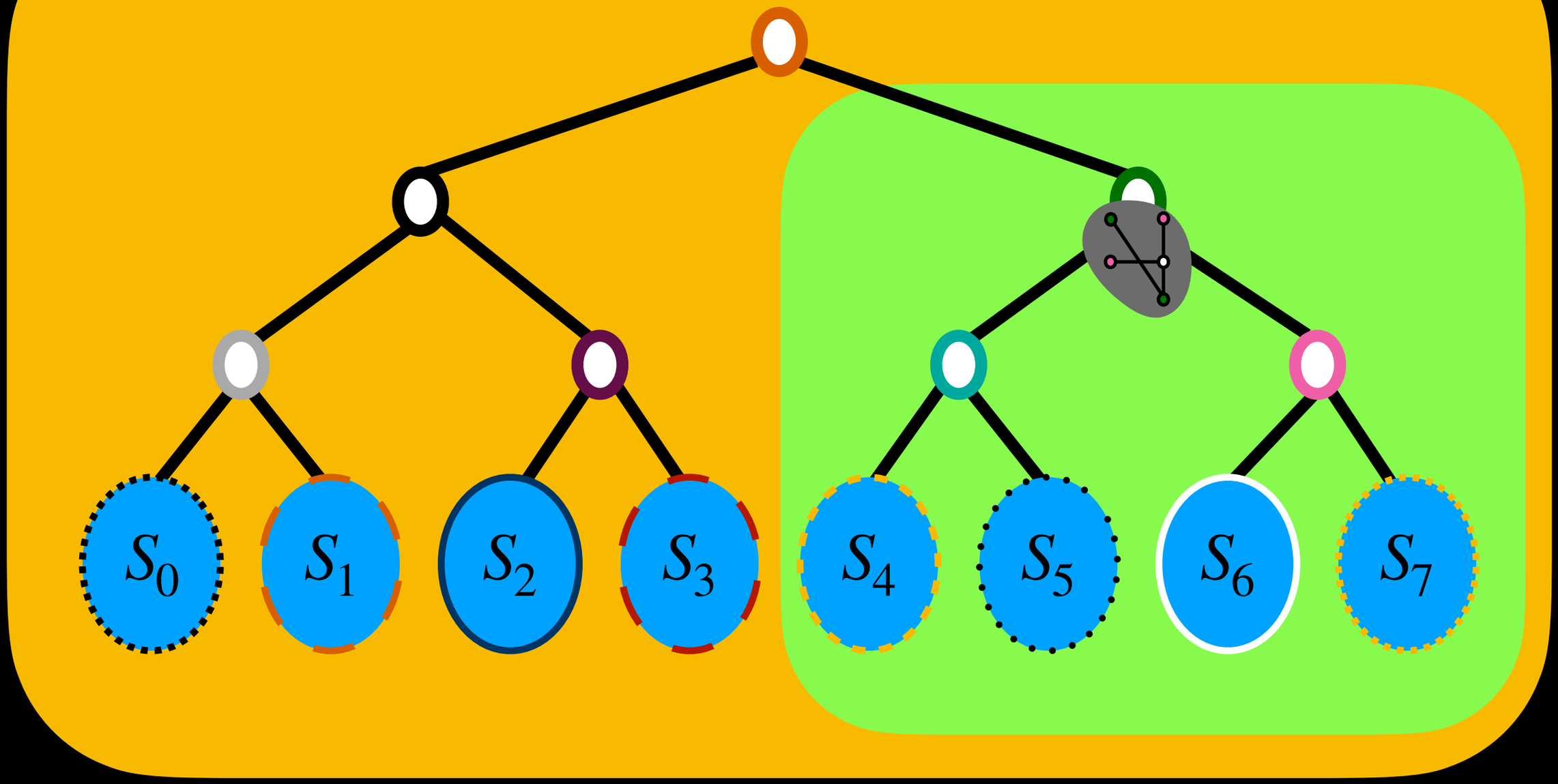


Generalization to ℓ Servers



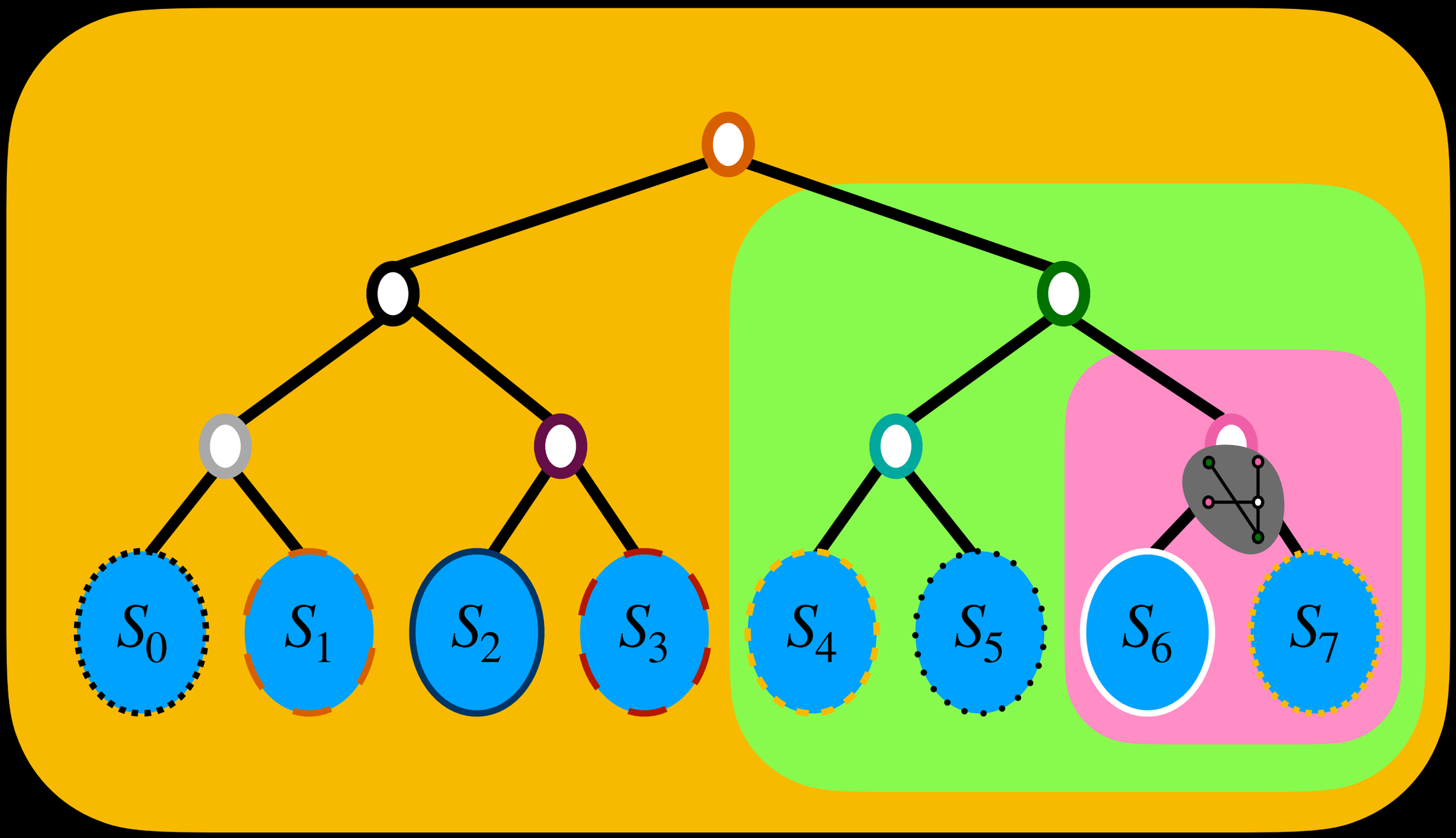
Traverse tree from root downwards and perform majority voting step at each internal node

Generalization to ℓ Servers



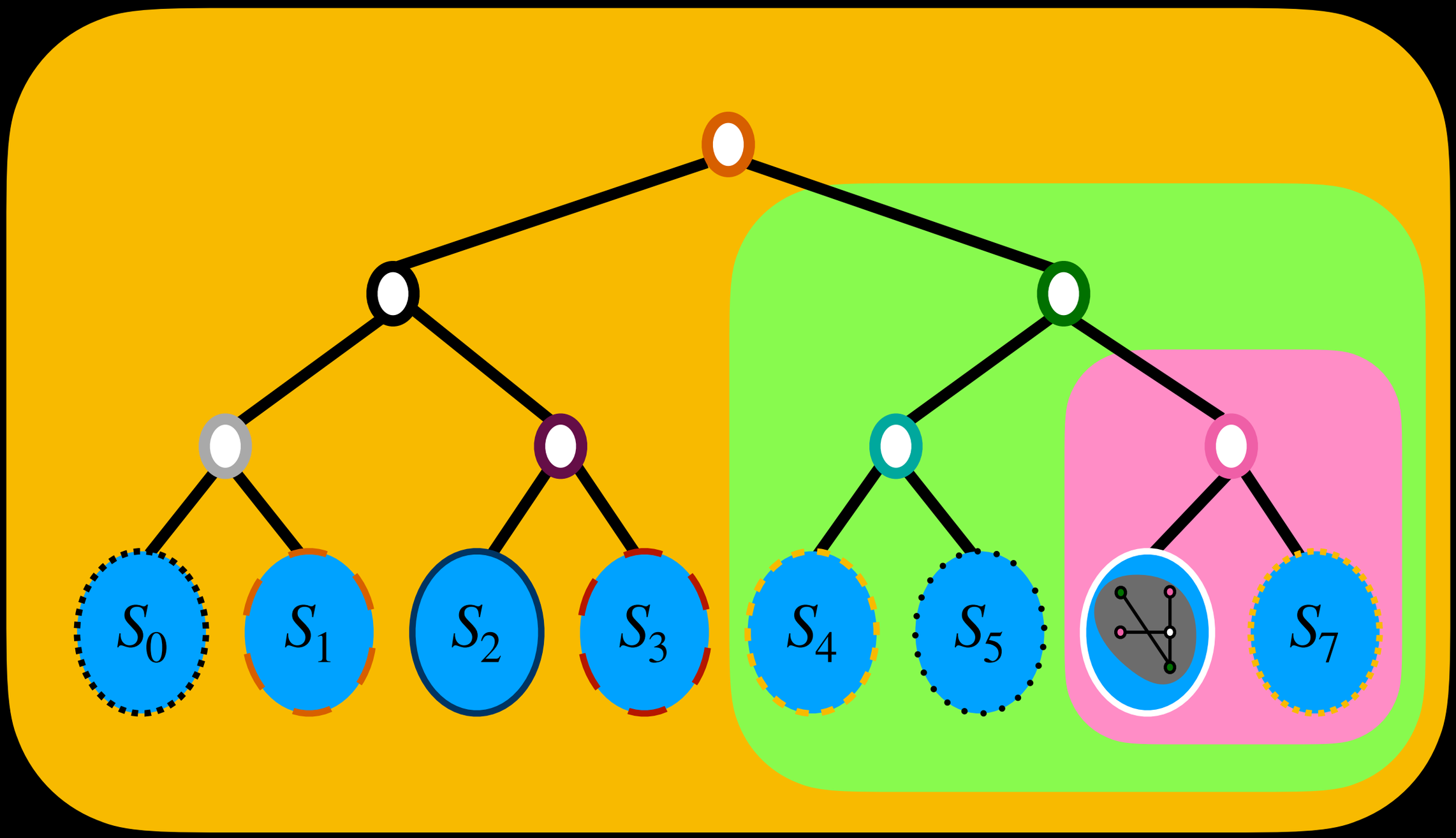
Traverse tree from root downwards and perform majority voting step at each internal node

Generalization to ℓ Servers



Traverse tree from root downwards and perform majority voting step at each internal node

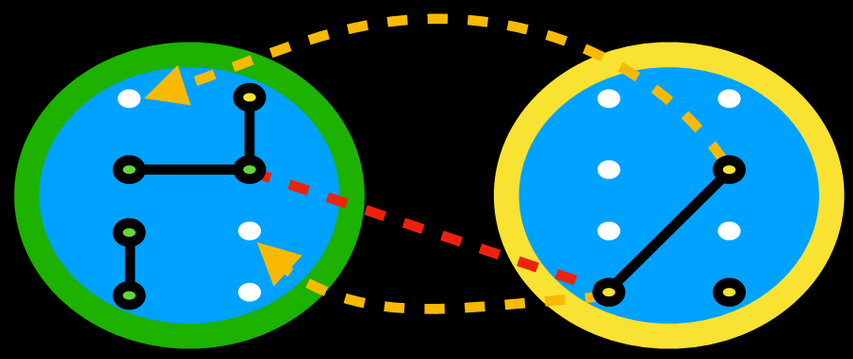
Generalization to ℓ Servers

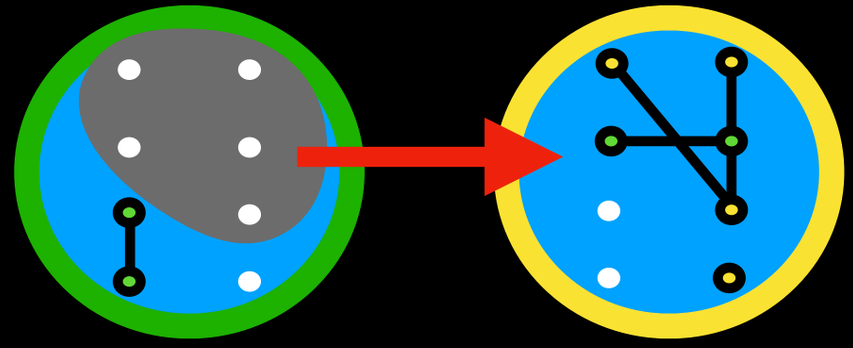


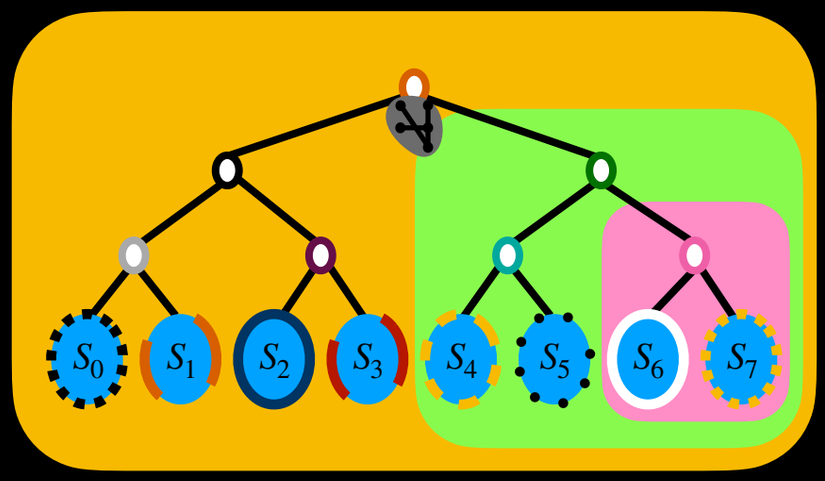
Traverse tree from root downwards and perform majority voting step at each internal node

Summary

- We introduced a new model for online workload (re-)embedding
- Distributed algorithm which is $O\left((\ell \log n \log \ell)/\varepsilon\right)$-competitive
- Applications to version of fundamental problems such as union find and k-way partition





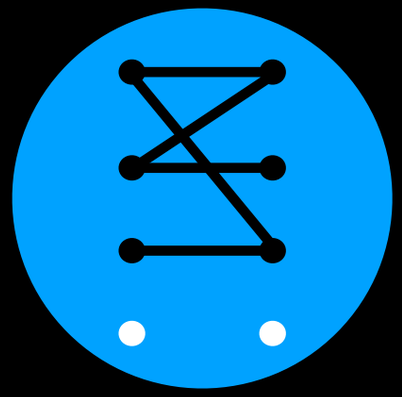


Open Problems

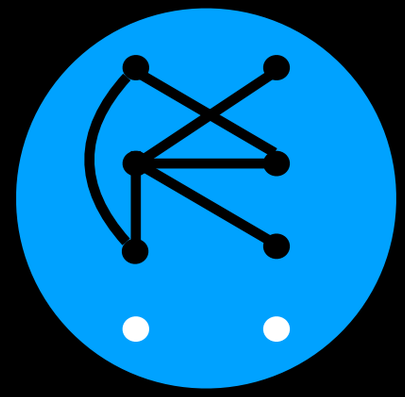
- Our algorithm for ℓ servers has competitive-ratio $O\left((\ell \log n \log \ell)/\varepsilon\right)$. Can we shave the ℓ-factor?

- Study generalized setting where communication patterns can change arbitrarily over time

- Tuning our algorithms further to perform even better in specific use cases



Thank you!



Efficient Distributed Workload (Re-)Embedding

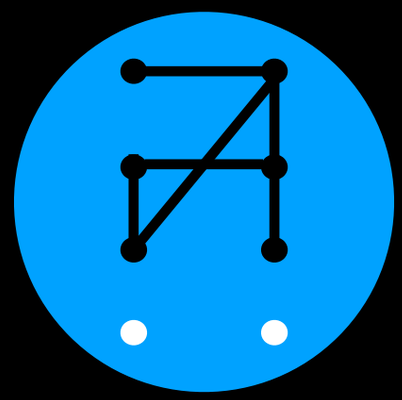
new model for distributed workload (re-)embedding

Applications

distributed algorithm with competitive ratio $O\left((\ell \log n \log \ell)/\varepsilon\right)$

Distributed Union Find Data Structure
(with small cost for re-locating the sets)

Online Balanced *k*-way Partition
(with small cost for assigning numbers to partitions)



Related Work

- Avin et al. (DISC'16, SIDMA'19):
  - No ground-truth assumption
  - $\tilde{O}(n/\ell)$-competitive algorithm
  - Competitive ratio for deterministic algorithms is $\Omega(n/\ell)$