Virtual Machine Image Management in Geo-Distributed Cloud Environment

Jad Darrous, Inria, Avalon/Ascola
Shadi Ibrahim, Ascola
Christian Perez, Avalon

CONTEXT

GEO-DISTRIBUTED CLOUDS

Designed by freepik.com

FOG

Designed by freepik.com

Virtual Machine Images Management

- Challenges:
  - VM Images are **huge** and “frequently” updated.
  - Fog nodes have **limited** storage capacity.
  - World-Area Networks (WANs) have **low bandwidth**[1].
  - WANs have **monetary cost** of communication.
  - The data centres where the VMs will be provisioned are not known in advance.

[1] X15 times in average comparing to LAN. K. Hsieh, et al. Gaia: Geo-distributed machine learning approaching LAN speeds. In NSDI, 2017.

Virtual Machine Images Management

- Objectives:
  - Minimize **storage cost**, at rest, on each site.
  - Minimize **network cost**, by reducing the amount of transferred data over WAN.
  - Minimize the **transfer time (i.e. increase the system bandwidth)**, by balancing the load as much as possible.

RELATED WORK

RELATED WORK

- Broadcast-based methods:
  - Tree-based broadcast, Chain-based broadcast and BitTorrent-based broadcast[1].
- Main focus is the distribution in single datacenter/cluster.
- Shortcoming for Geo-distribution:
  - Designed to work with local high speed (homogenous) networks.
  - Network cost is NOT taken into account.
  - The only considered factor is *Time*.

[1] Jeanvoine et al. Kadeploy3: Efficient and Scalable Operating System Provisioning for HPC Clusters. [Research Report] RR-8002, INRIA. 2012.

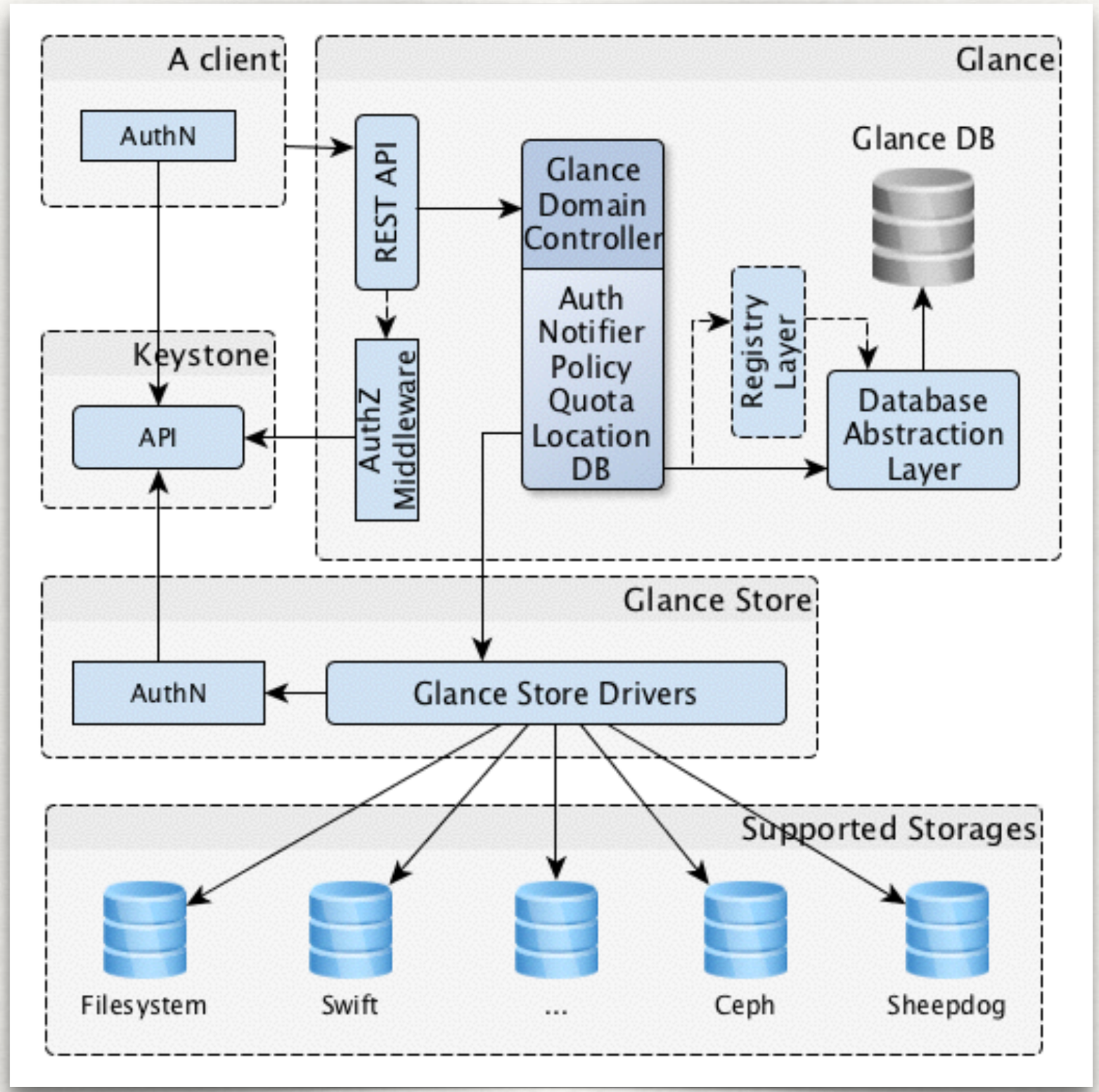
OPENSTACK

GLANCE & SWIFT

OPENSTACK GLANCE

- Responsible for managing VM assets.
  - Discovering, registering, and retrieving virtual machine (VM) images.
  - Querying of VM image metadata.
- Actual VMIs are stored in a dedicated storage system
  - Store/Retrieve VMI
  - Support multiple storage backends:
    - Swift, RADOS, S3, Sheepdog, local file system…

OPENSTACK GLANCE



OPENSTACK SWIFT

- Distributed, eventual consistent, object store.
- *Write Once / Read Many* access pattern.
- Starting from version 1.9*, Swift supports *Global Clusters*
  - The concept of “region” has been introduced
  - Read & write affinity
- Internally, Swift relies on **rsync** to replicate the data between regions.

* https://github.com/openstack/swift/blob/stable/ocata/CHANGELOG

APPROACH

APPROACH

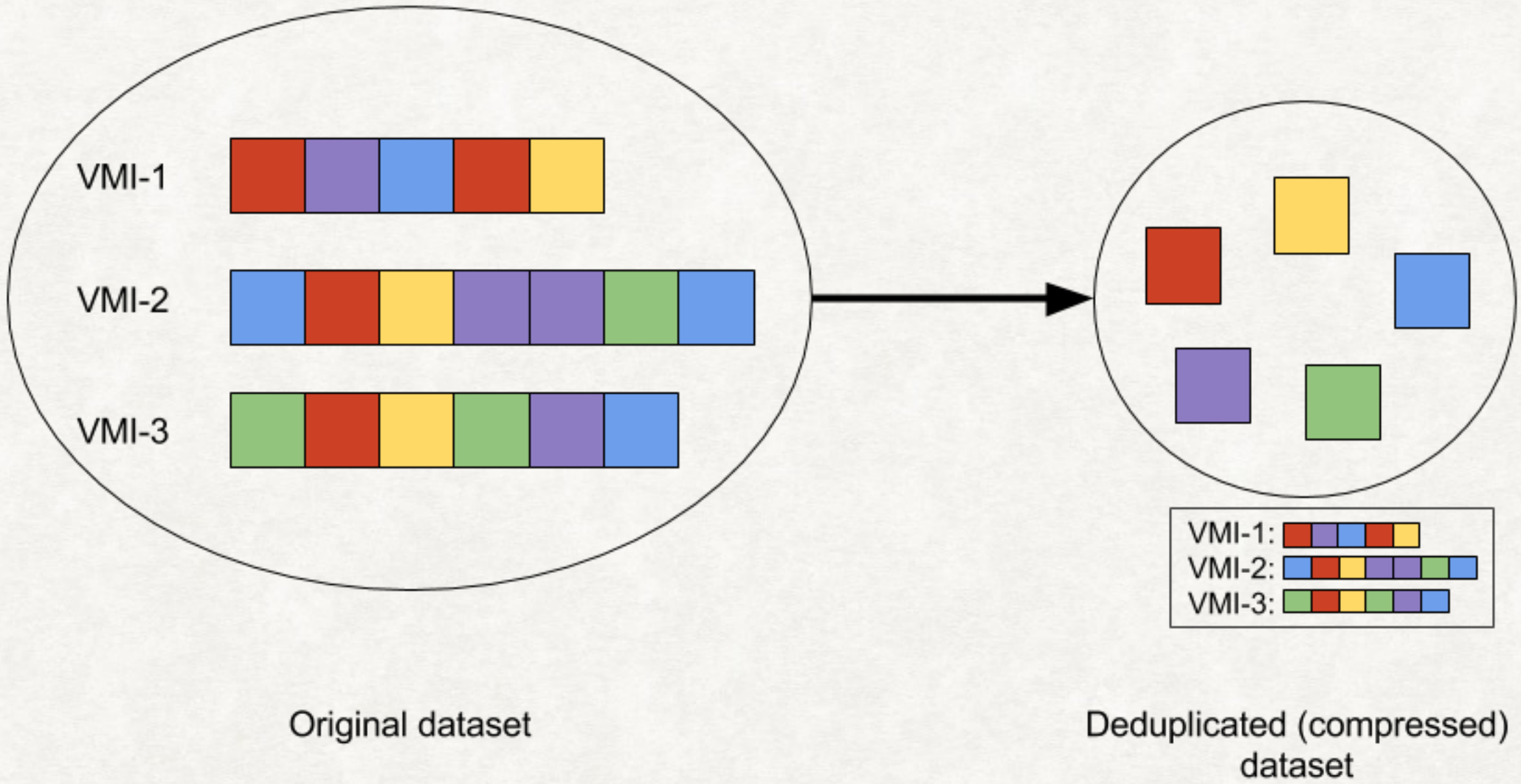
- Fragment the VM Images into pieces (i.e. chunks)
- Scheduling algorithm for efficient retrieval of VM Image chunks
- Chunk placement

OUR SOLUTION

DEDUPLICATION

A COMPRESSION TECHNIQUE

- Deduplication ratio in VMIs dataset could reach 80% [1,2]



[1] K. R. Jayaram, et al. “An empirical analysis of similarity in virtual machine images,” in Proceedings of the Middleware 2011.
[2] K. Jin and E. Miller. The effectiveness of deduplication on virtual machine disk images. In Proc. SYSTOR 2009.

GEO-DISTRIBUTED DATA CENTRES

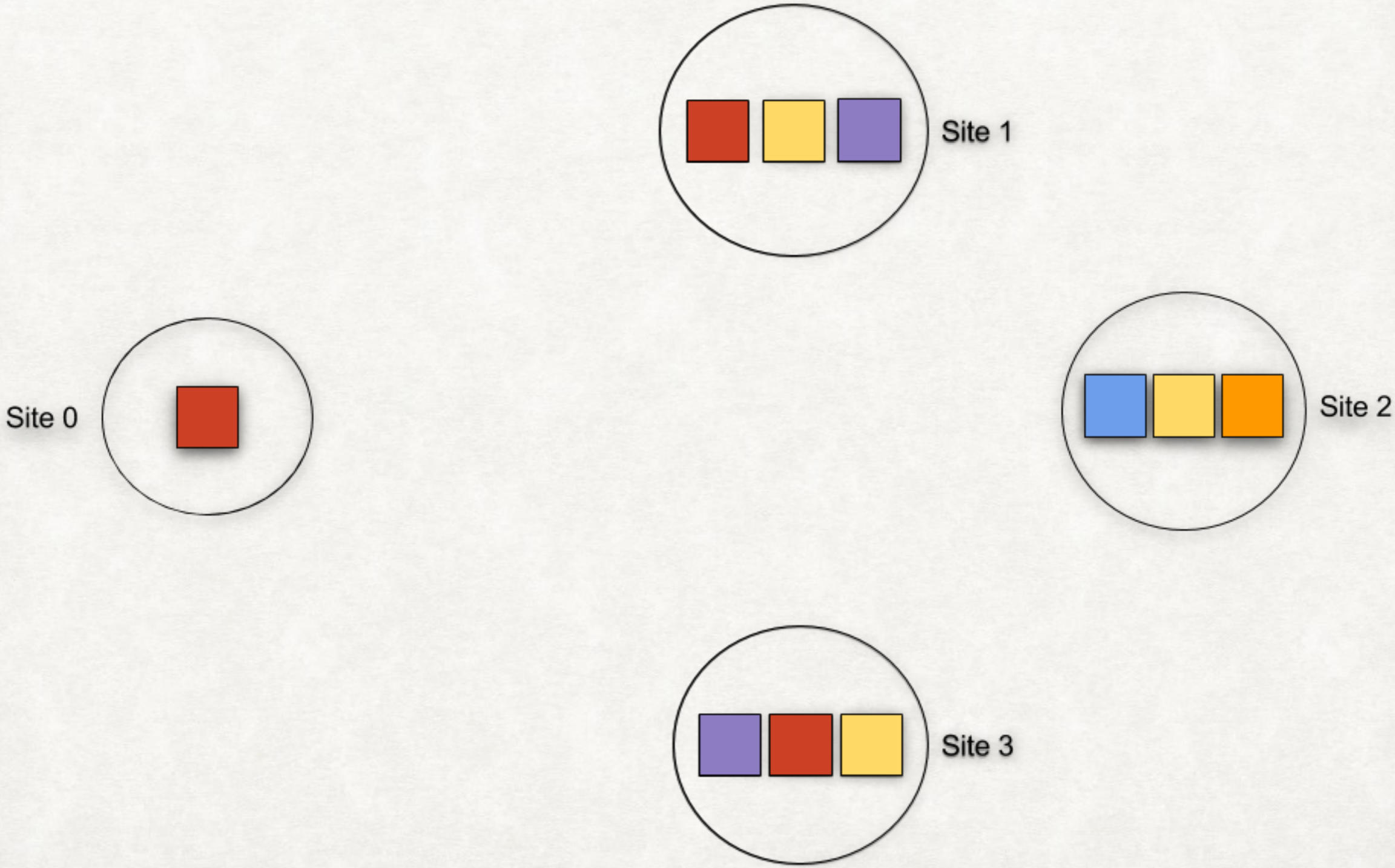


IMAGE REQUEST

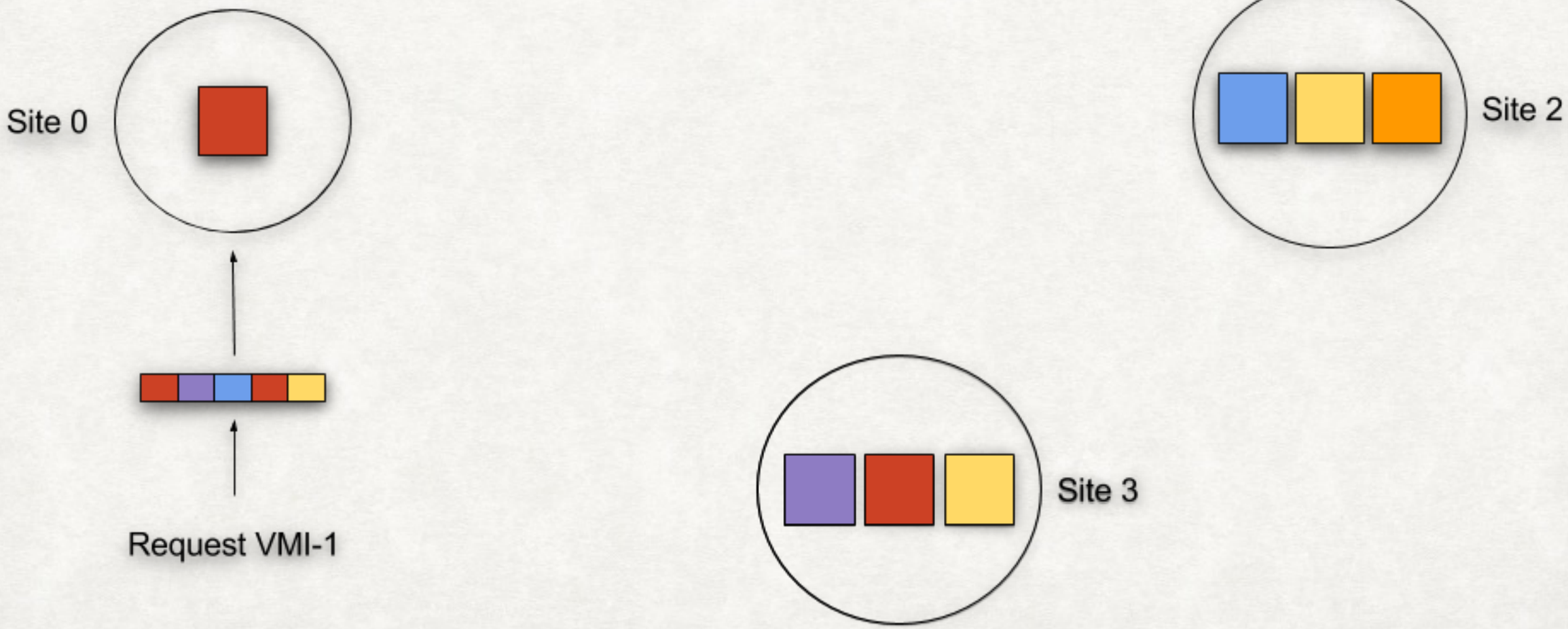


IMAGE REQUEST

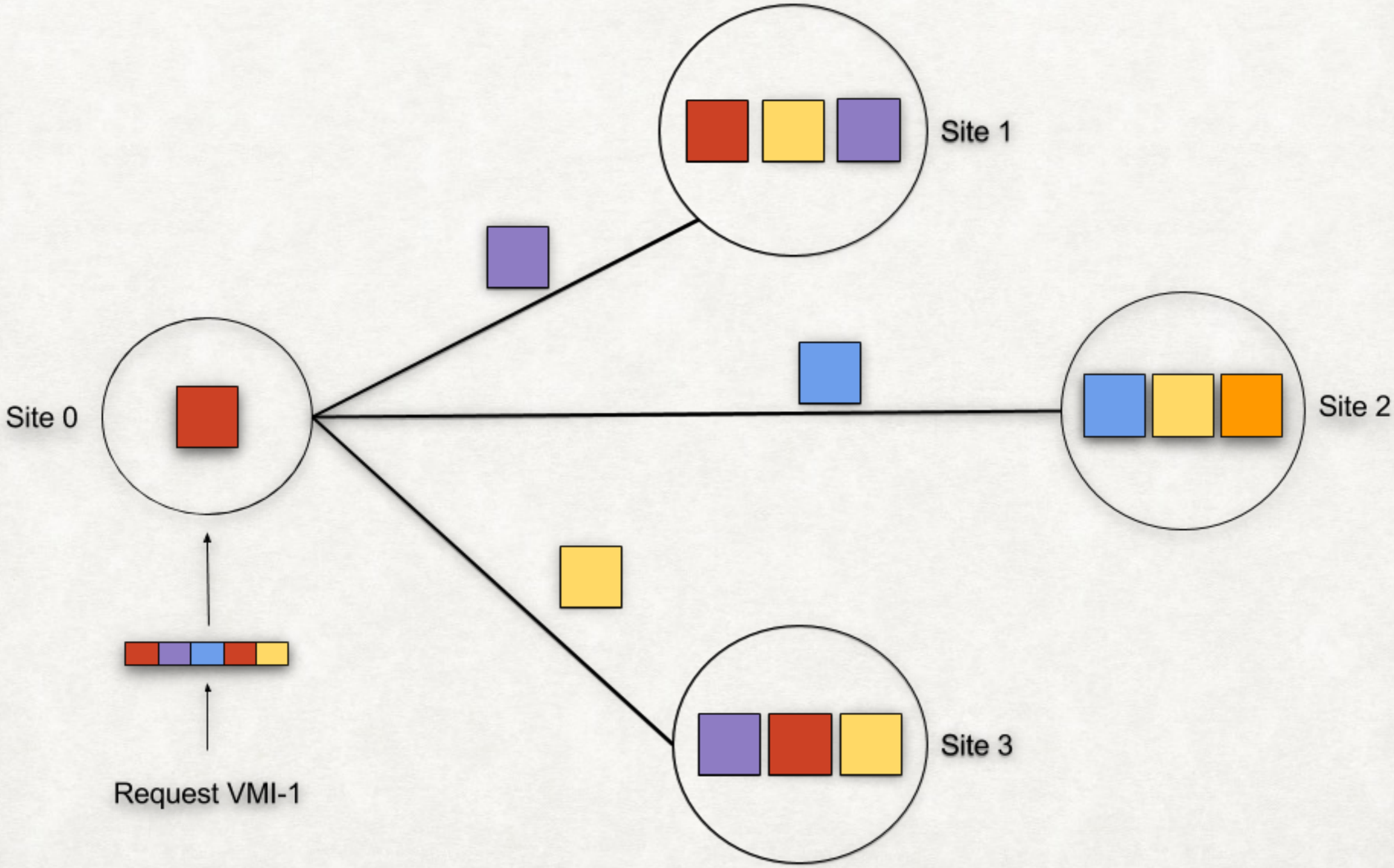


IMAGE REQUEST

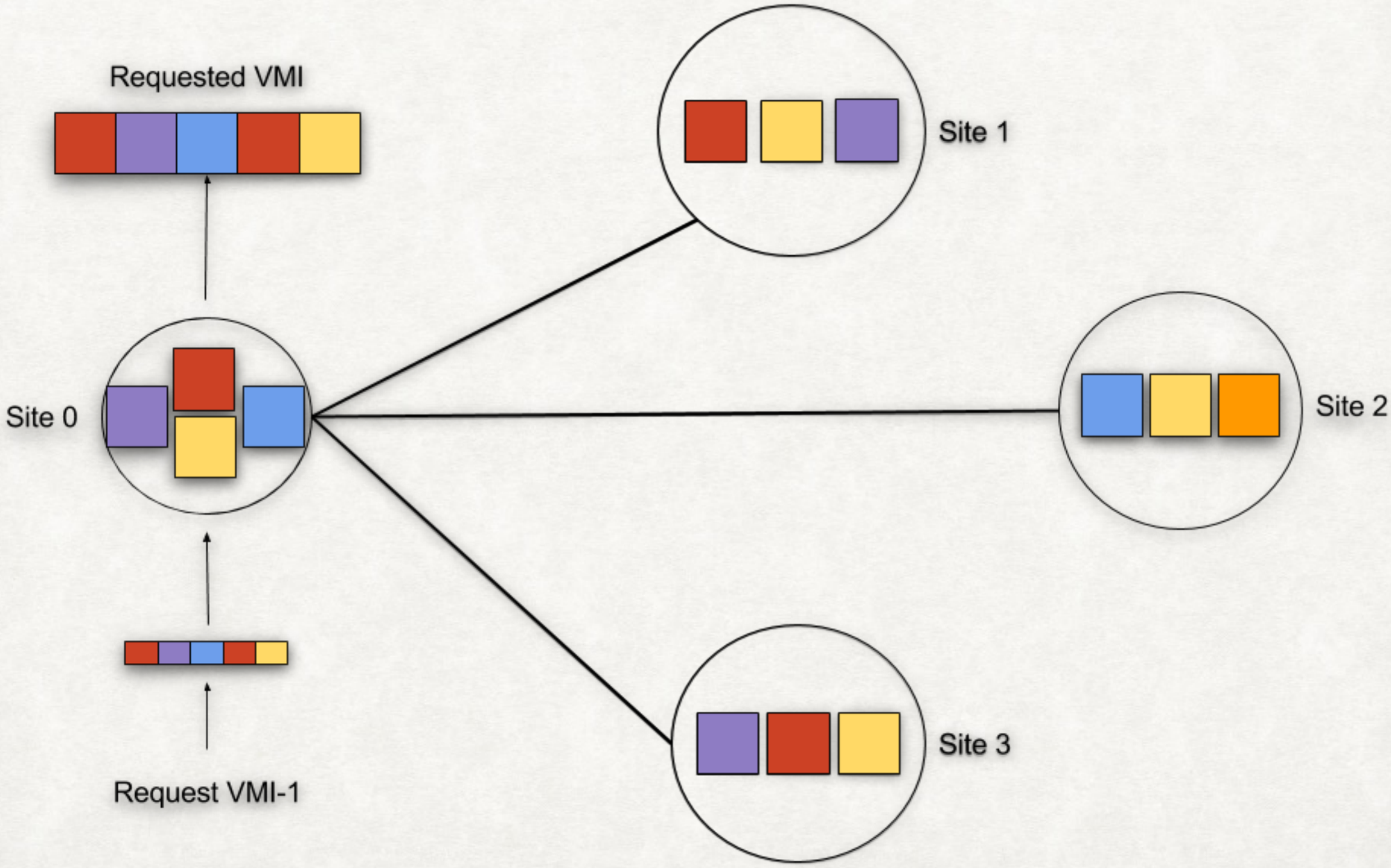


IMAGE REQUEST

HETEROGENEOUS NETWORK

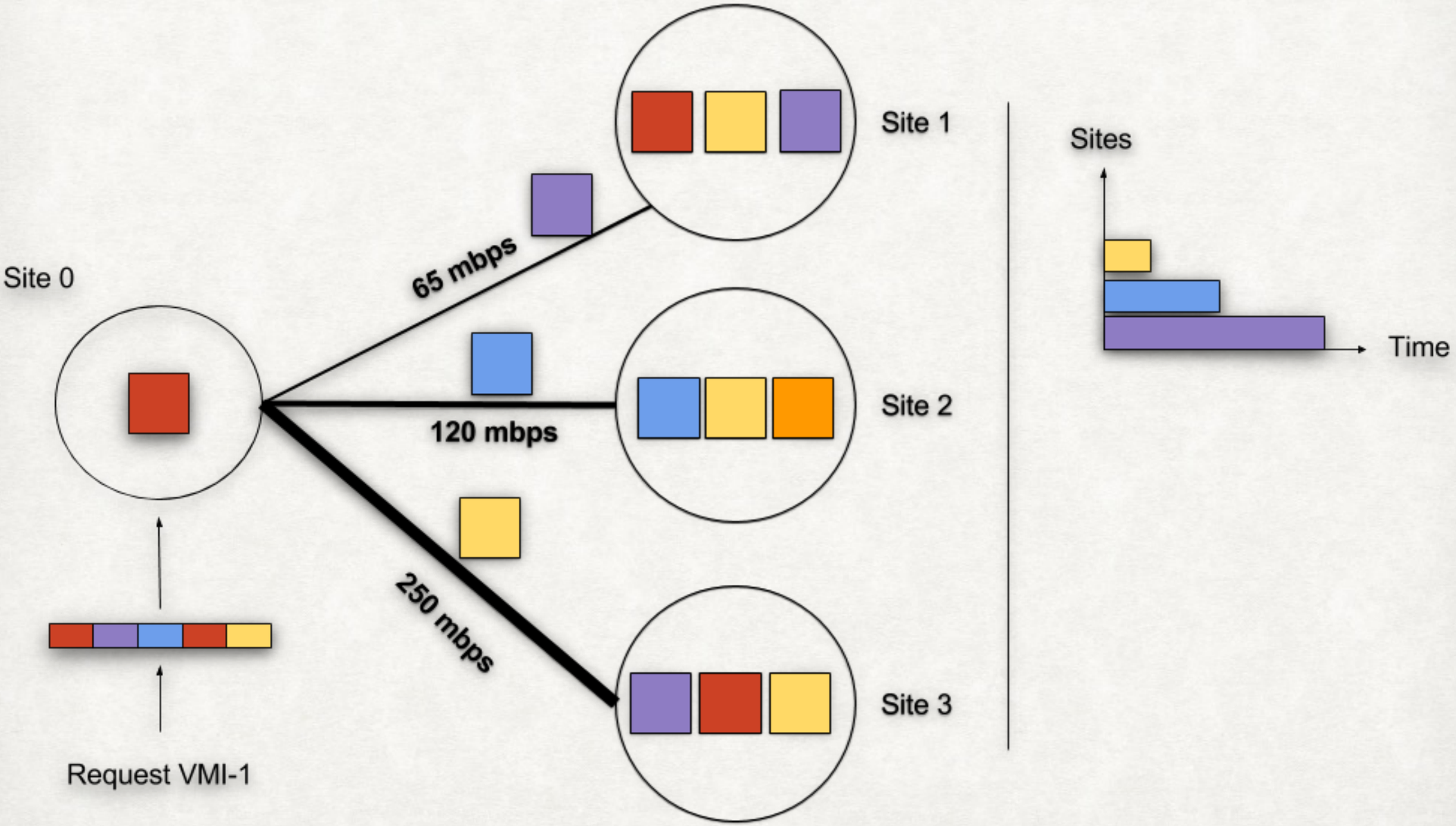
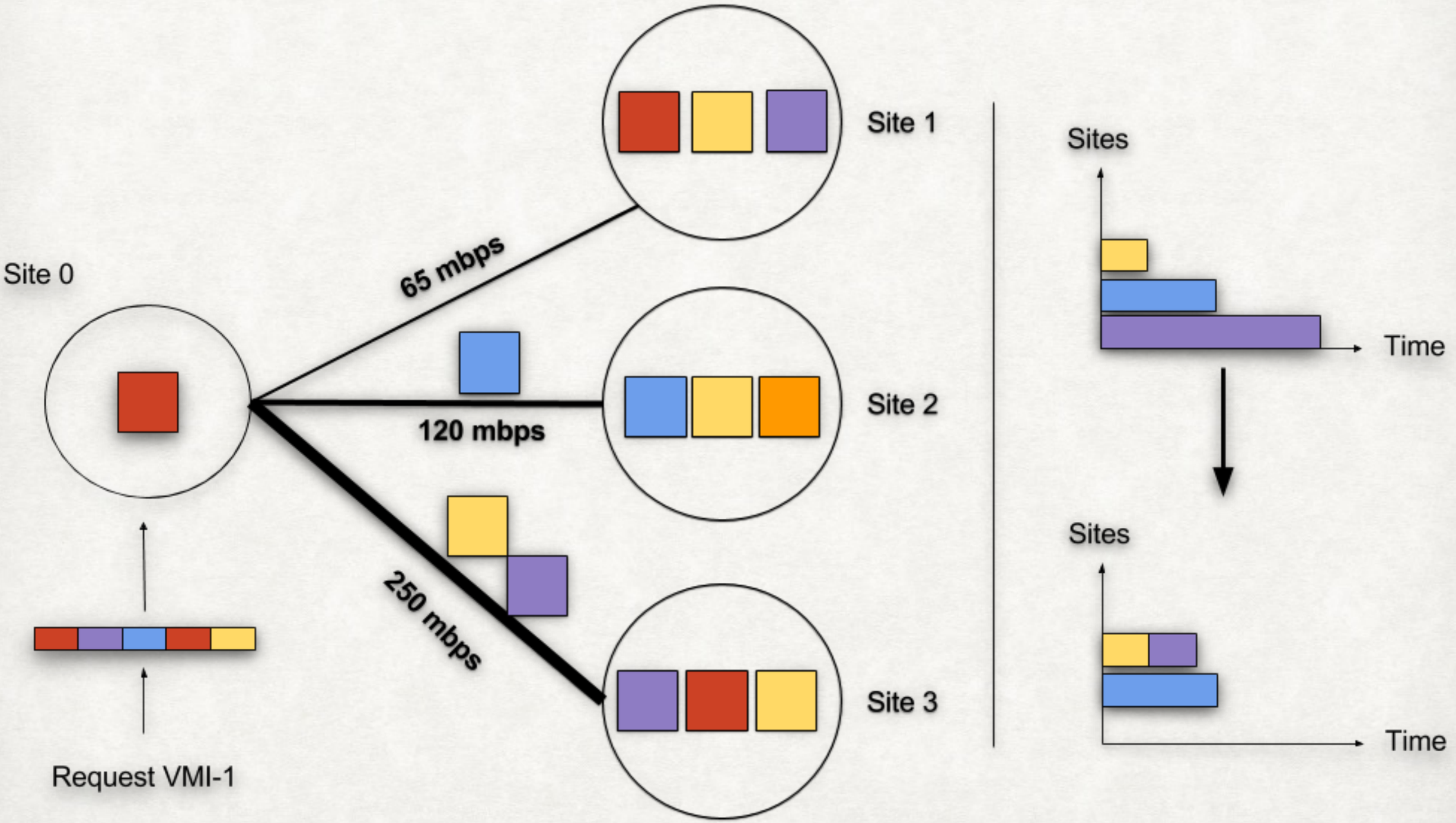


IMAGE REQUEST

HETEROGENEOUS NETWORK



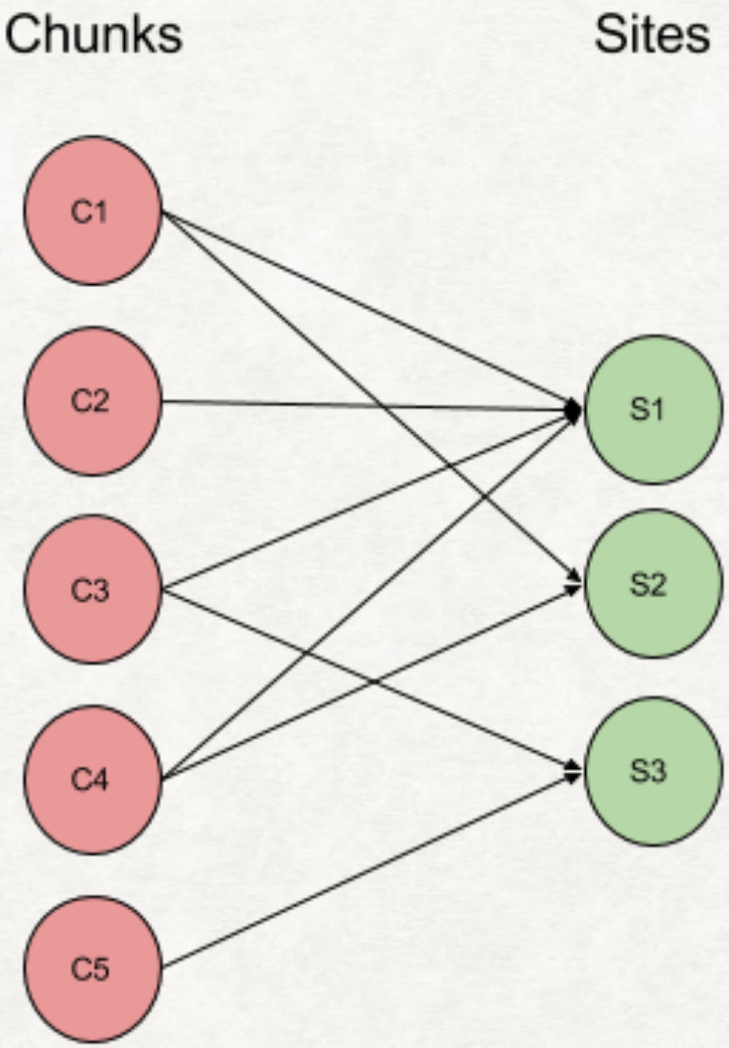
NITRO

NITRO

- A prototype of Geo-distributed deduplicated storage system.
- Fine-grained **network-awareness**.
- Chunk scheduling algorithm
  - Produces **exact** solution in **polynomial time**.
  - Based on *Matching* algorithm in *Bipartite Graph*.
- Metadata is replicated synchronously.
- Implemented in Python, approx. 2000 LOC.

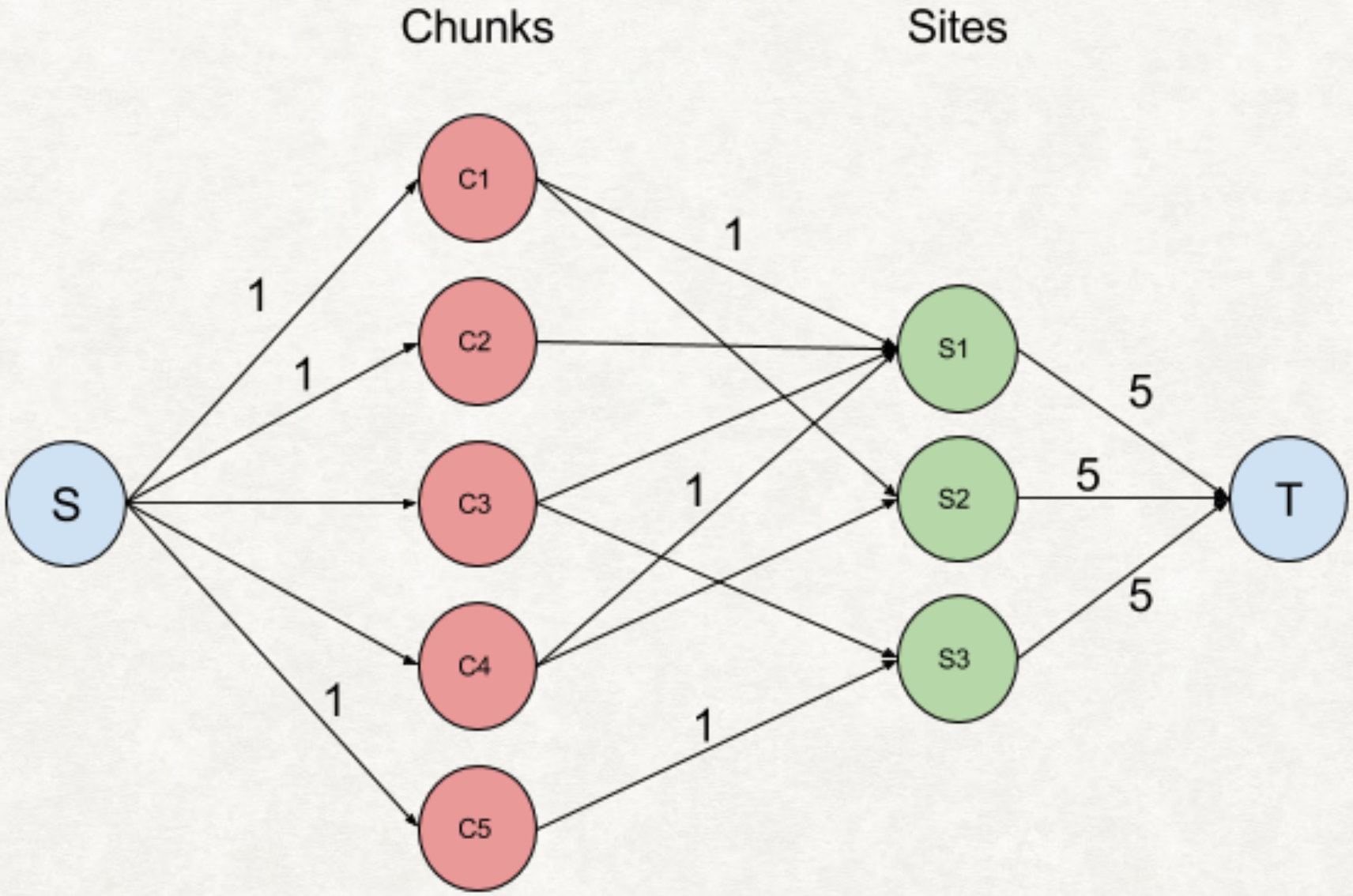
CHUNK SCHEDULING

MODELING



CHUNK SCHEDULING

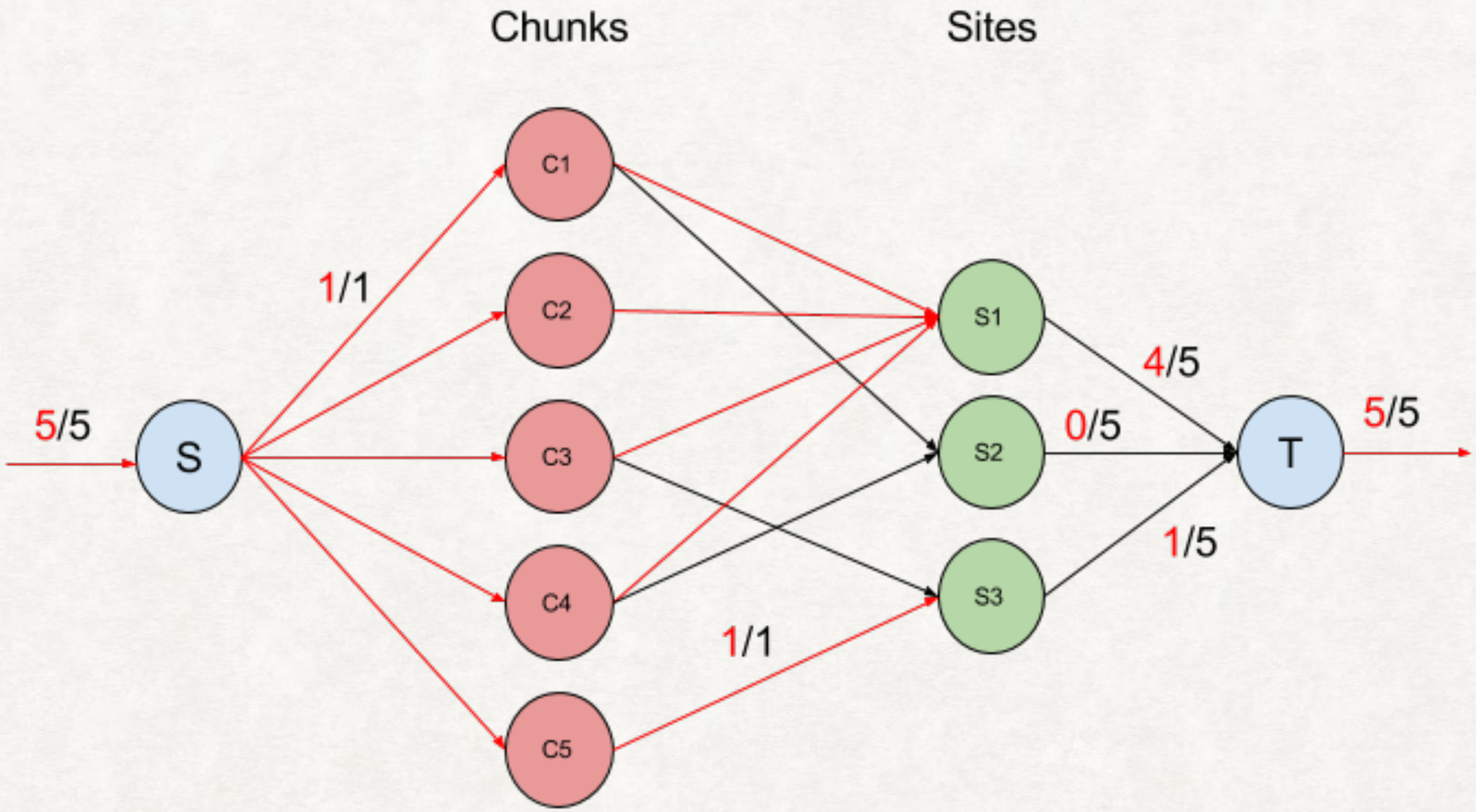
MATCHING IN BIPARTITE GRAPH*



* https://en.wikipedia.org/wiki/Matching_(graph_theory)#Maximal_matchings

CHUNK SCHEDULING

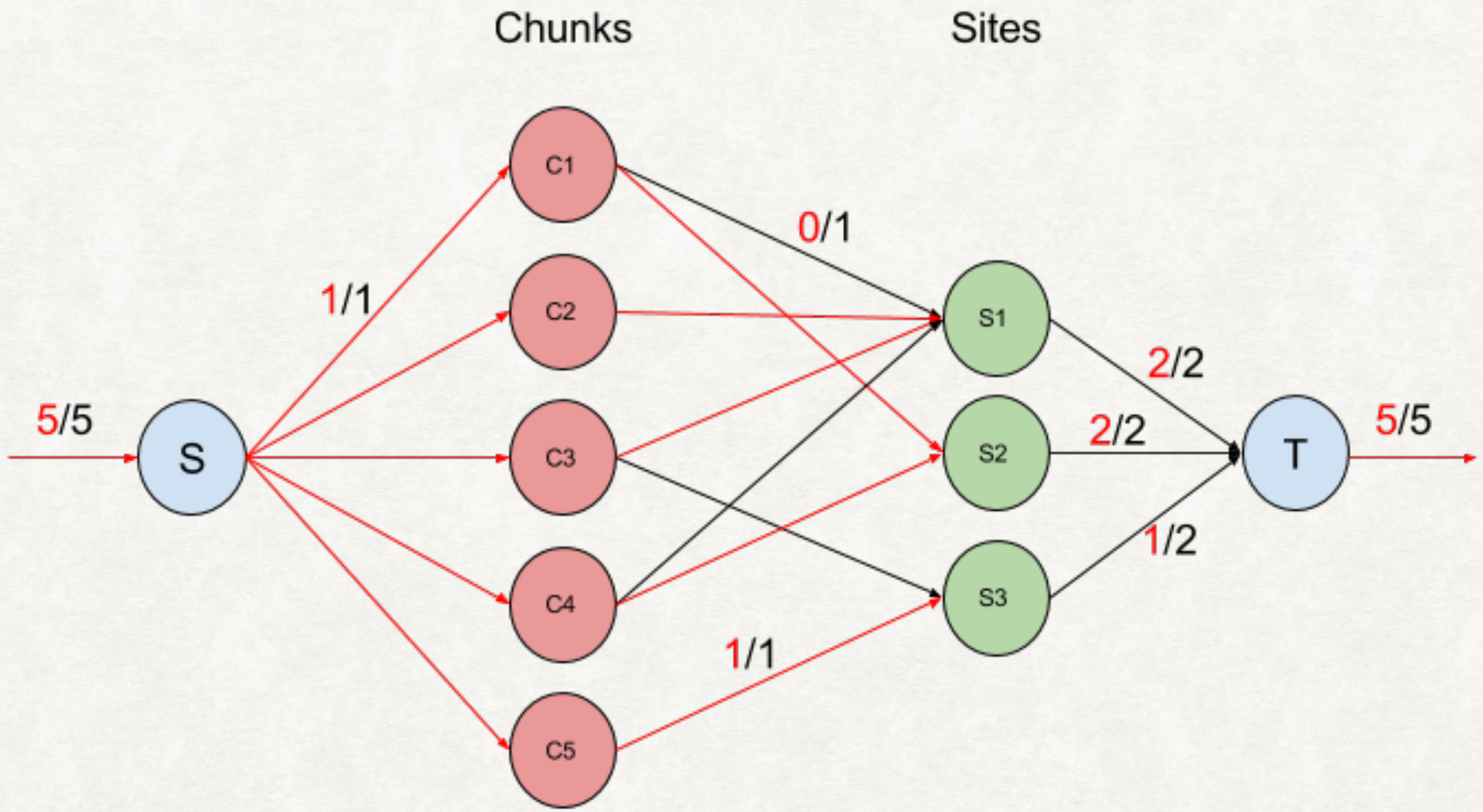
MATCHING IN BIPARTITE GRAPH*



* https://en.wikipedia.org/wiki/Matching_(graph_theory)#Maximal_matchings

CHUNK SCHEDULING

MATCHING IN BIPARTITE GRAPH*



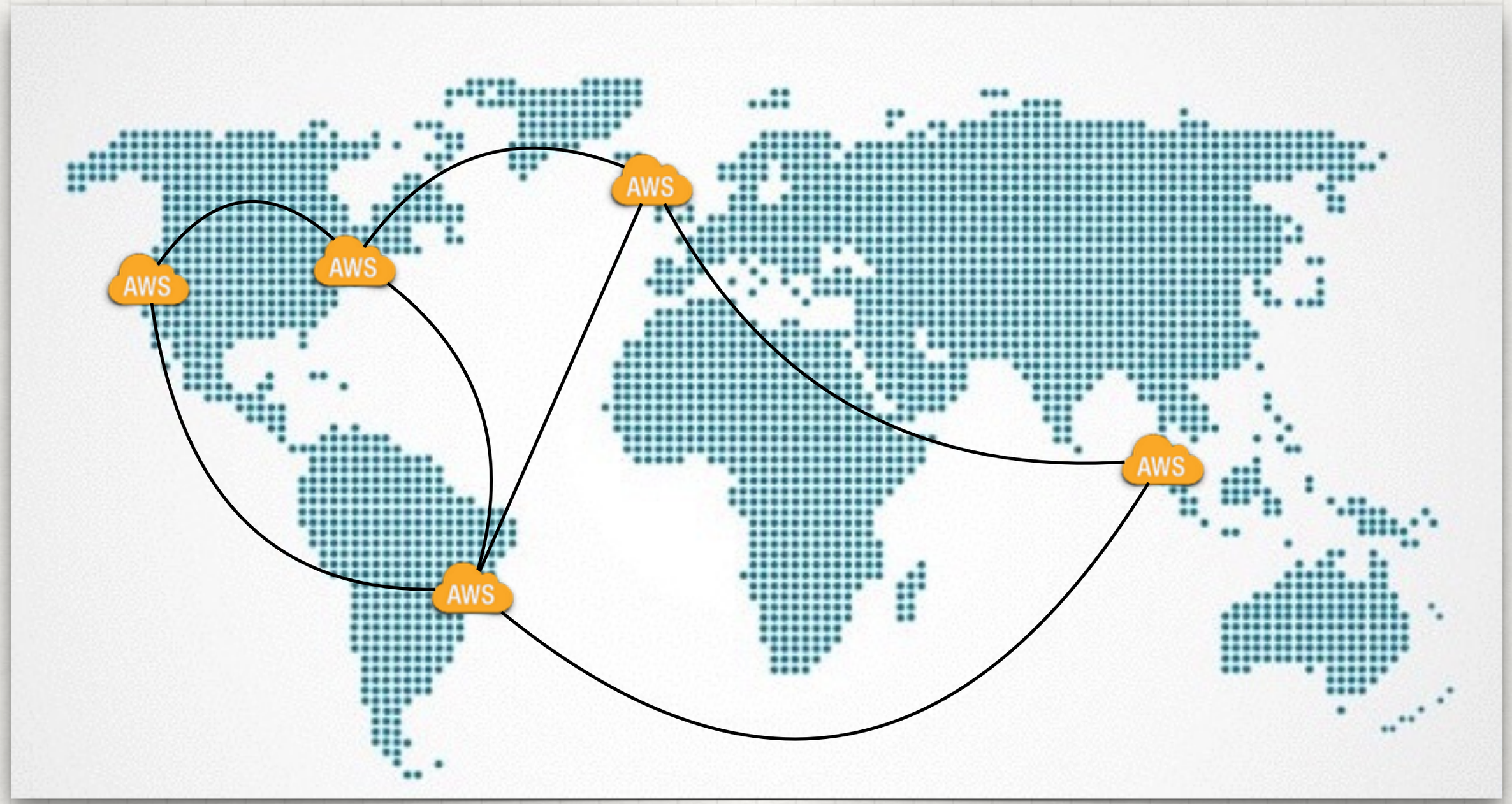
* https://en.wikipedia.org/wiki/Matching_(graph_theory)#Maximal_matchings

EXPERIMENTS

EXPERIMENT SETUP

- Dataset
  - 24 VM images
    - 8 base images
      - 3 Debian: Wheezy, Jessie and Stretch
      - 3 Fedora: Fedora-23, Fedora-24 and Fedora-25
      - 2 Ubuntu: Trusty and Xenial
    - 3 software variations: Cassandra, Hadoop and LAMP stack
  - Total size around **100 GB**, each image between 2.5GB and 6.5GB
- Testbed
  - 5 nodes of *nova* cluster in Lyon site of **Grid5000**

EMULATED TOPOLOGY*

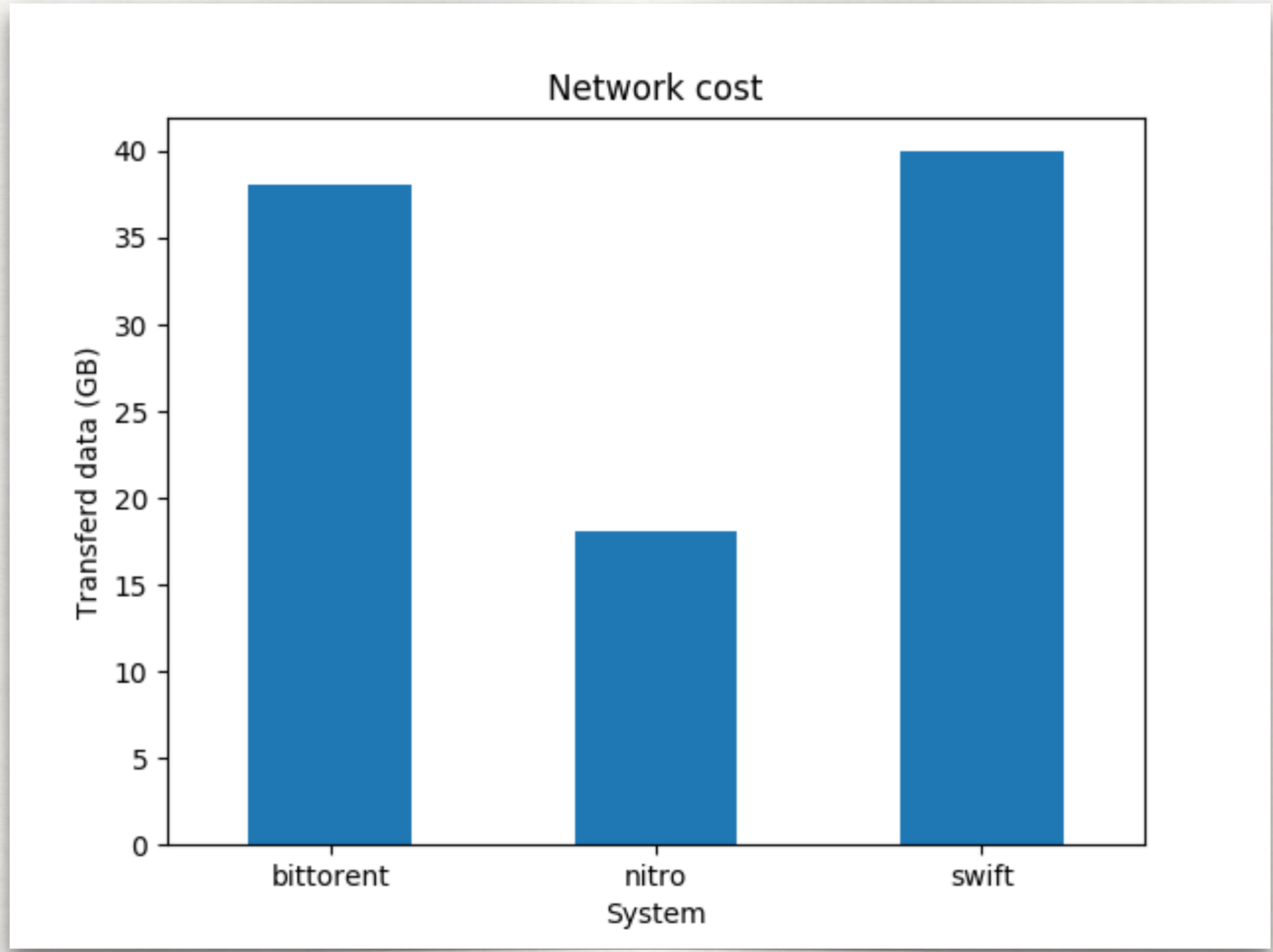


* C.Li et al. Making geo-replicated systems fast as possible, consistent when necessary. In OSDI, Oct 2012.

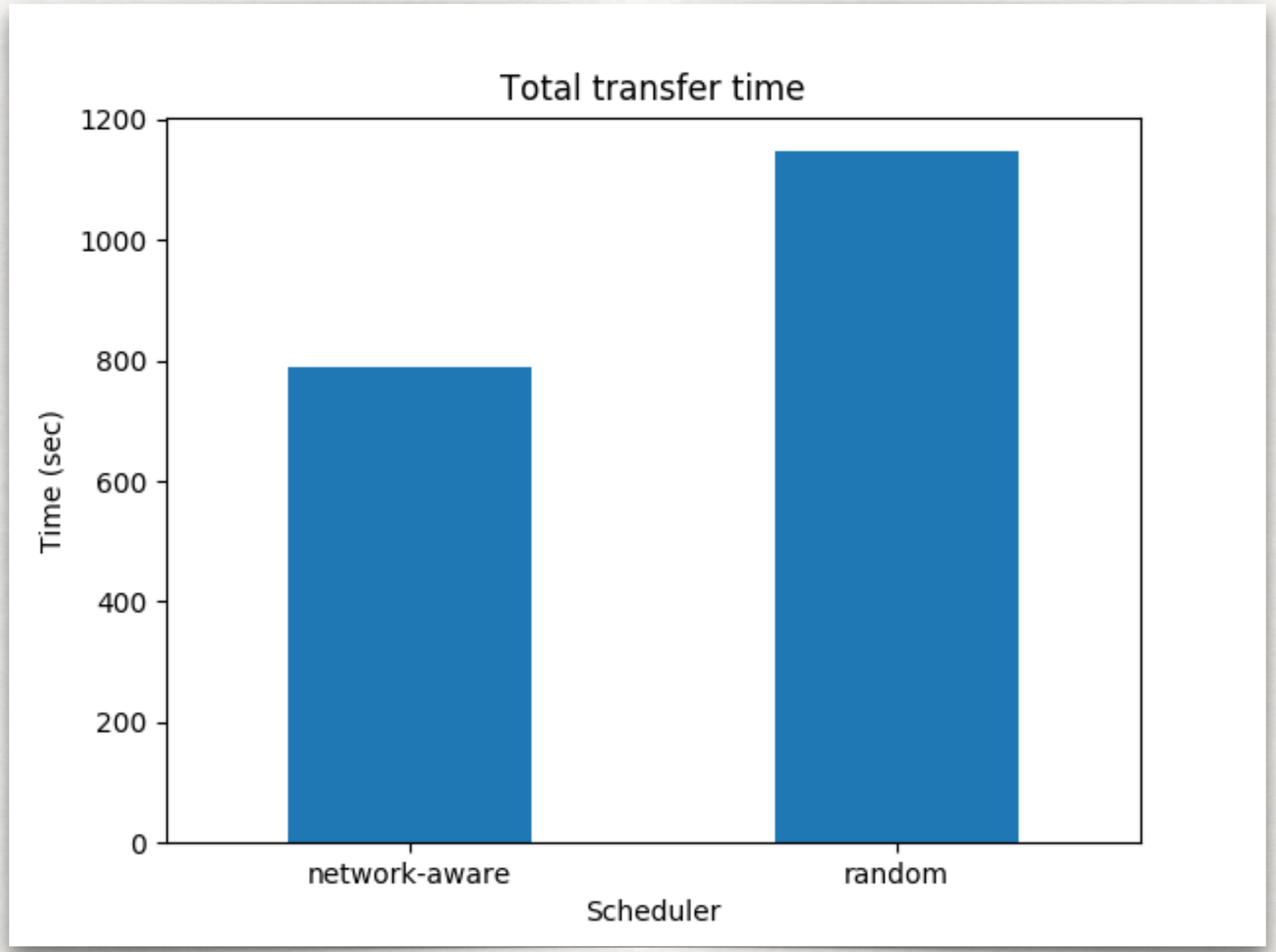
Designed by freepik.com

PRELIMINARY RESULTS

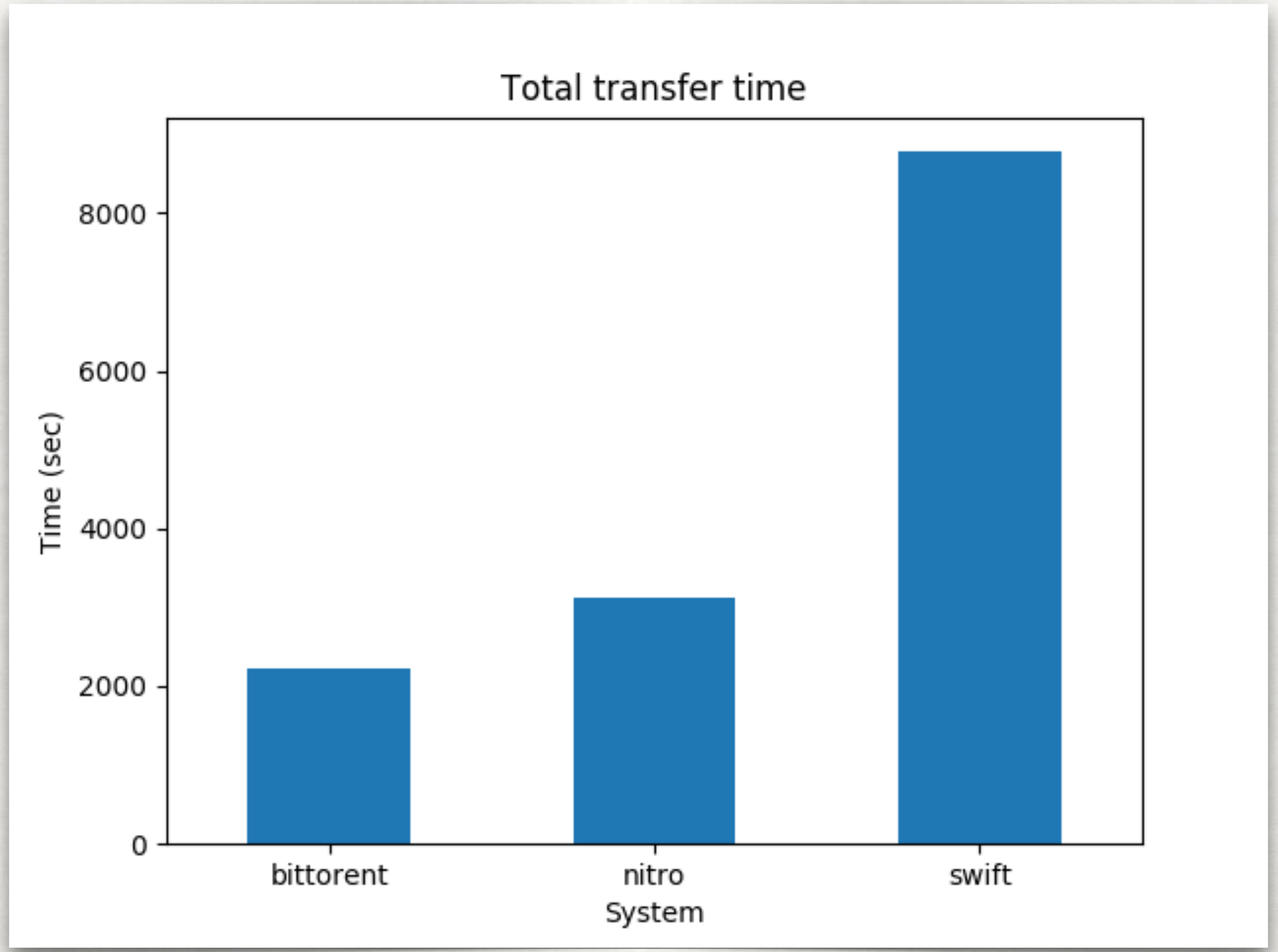
NETWORK COST



PRELIMINARY RESULTS



PRELIMINARY RESULTS



CONCLUSION

CONCLUSION

- The problem
  - Ensure the availability of huge image on multiple sites.
  - Minimize Storage/Transfer cost and Transfer time
- Nitro
  - Geo-distributed deduplication system.
  - Chunks scheduling algorithm
- Perspective
  - Extends Nitro to work in Fog environment
  - Chunk placement in Fog environment

NITRO + OPENSTACK

- Nitro can be implemented as a backend storage for Glance*
- The backend storage is selected in configuration files.
- The modification on OpenStack source code is **Zero**.
- Engineering work:
  - Implementing the driver (approx. 200-1000 LOC)
  - From prototype to “product”

* https://github.com/openstack/glance_store/tree/master/glance_store/_drivers