Cloud Computing

Software Defined Storage (SDS):

OpenStack Swift

--Jannatun.noor@bracu.ac.bd

BRAC UNIVERSITY
Inspiring Excellence

Available Object Storages in Open source

- SWIFT
- CEPH
- JETS3T

Swift Use Cases

- What is it?
    - Object Storage System.
    - Massively Scalable.
    - Runs on commodity hardware.
    - An S3 like solution
- What is not?
    - Hard drive / File system.
    - NFS / SMB share
    - Block Storage.
    - Any SAN/NAS/DAS
    - Not even a CDN ????

BRAC UNIVERSITY
Inspiring Excellence

Swift Use Cases contd.

- Multi-tenancy
  - Ideal for Public or Private Clouds
  - Different URLs, groups of users, access codes, fine-grained privileges.
- Backups
  - Write-once, read-never (long-term archiving).
  - Disaster recovery.
- Web content
  - Write many, read many
  - File sharing websites (temporary access)
  - Static websites or media focused blogs (i.e. Imgur)
- Large Objects
  - Medical / Scientific Images
  - Store your fancy Images from the moon (i.e. NASA)

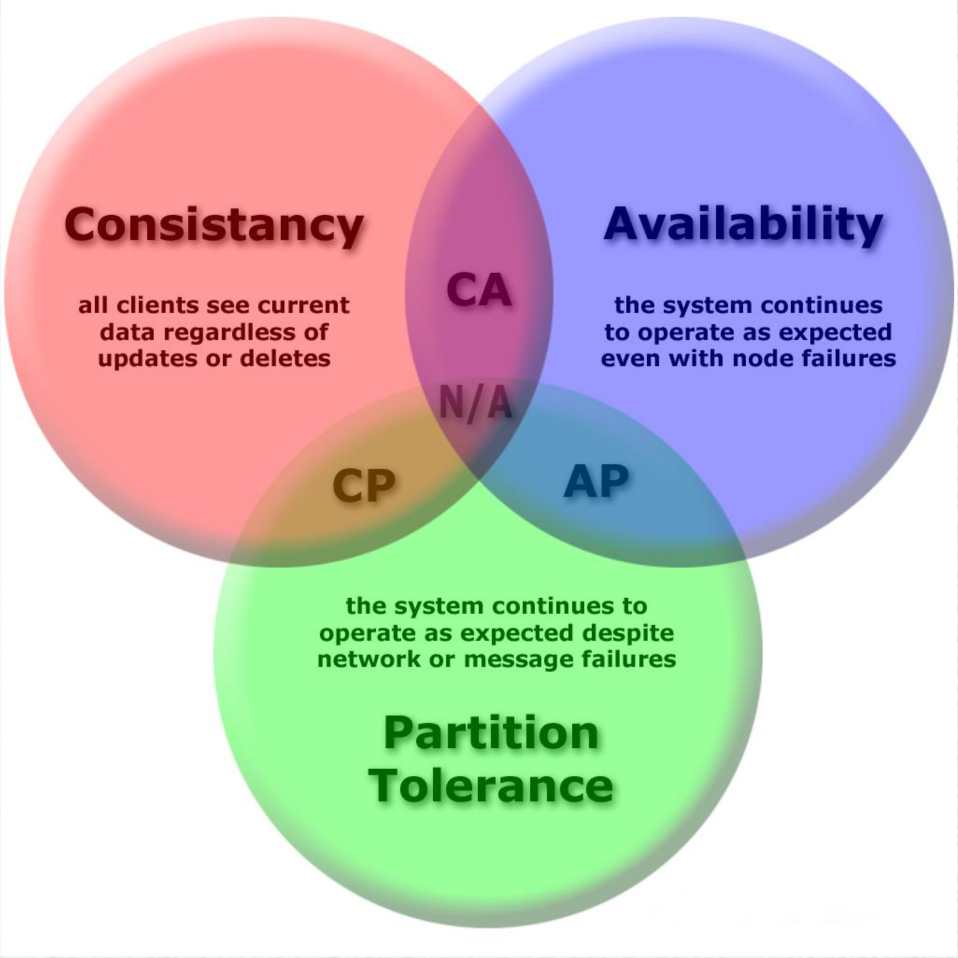
History

- Rackspace Cloud Files V1
  - Distributed Storage
  - Centralized Metadata
  - PostgreSQL DB
- 2009 : Rackspace Cloud Files V2 (SWIFT)
  - Full redesign and rewrite, opensource
  - API compatible with Amazon S3
  - Worked Closely with ops
  - Distributed storage and metadata
  - Logical placement, based on Algorithm

BRAC UNIVERSITY
Inspiring Excellence

CAP Theorem

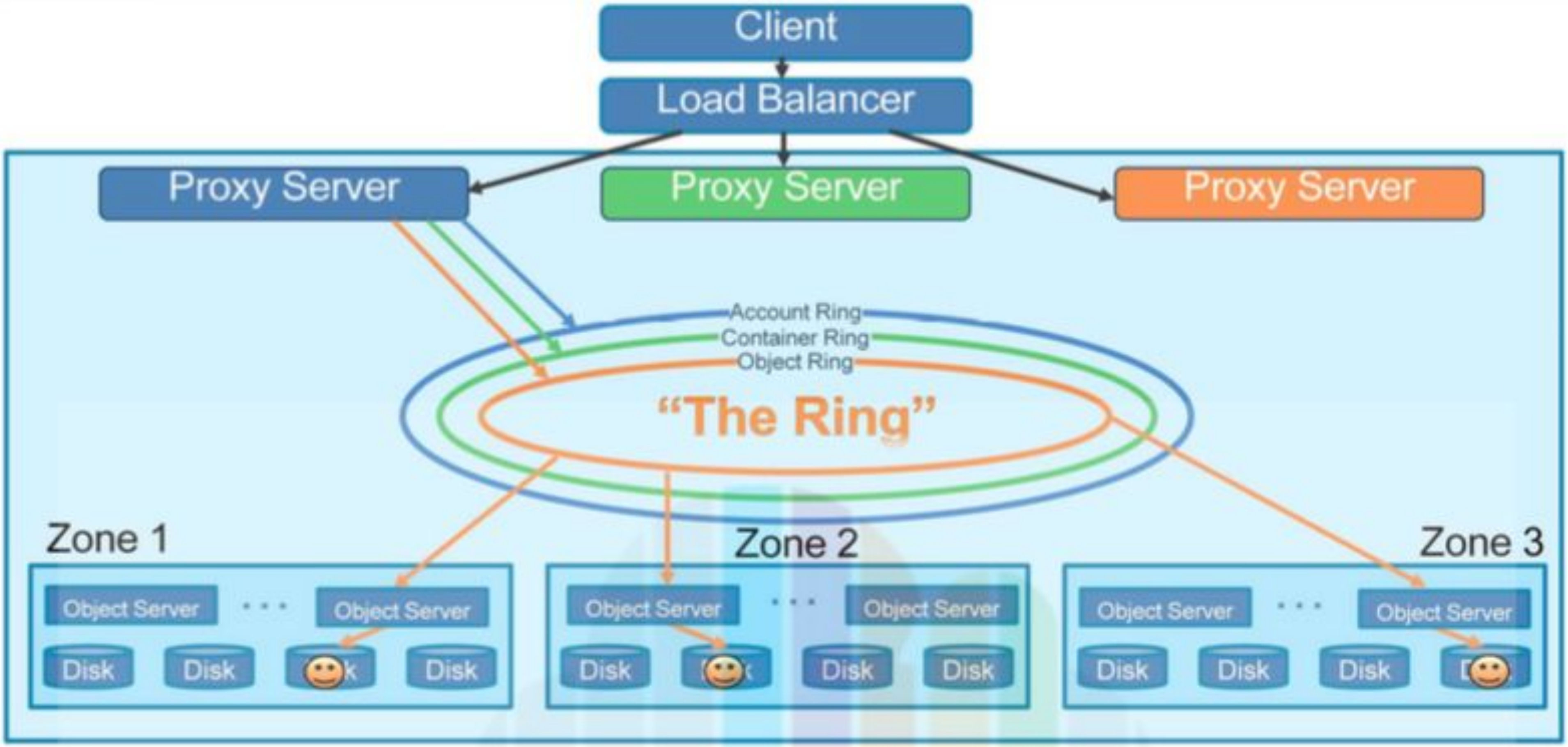
It is impossible for a distributed computer system to simultaneously provide all three of C, A, P.



In 2012 Brewer clarified some of his positions, including why the oft-used "two out of three" concept can be misleading or misapplied, and the different definition of consistency used in CAP relative to the one used in ACID.

BRAC UNIVERSITY
Inspiring Excellence

Swift Architecture



BRAC UNIVERSITY
Inspiring Excellence

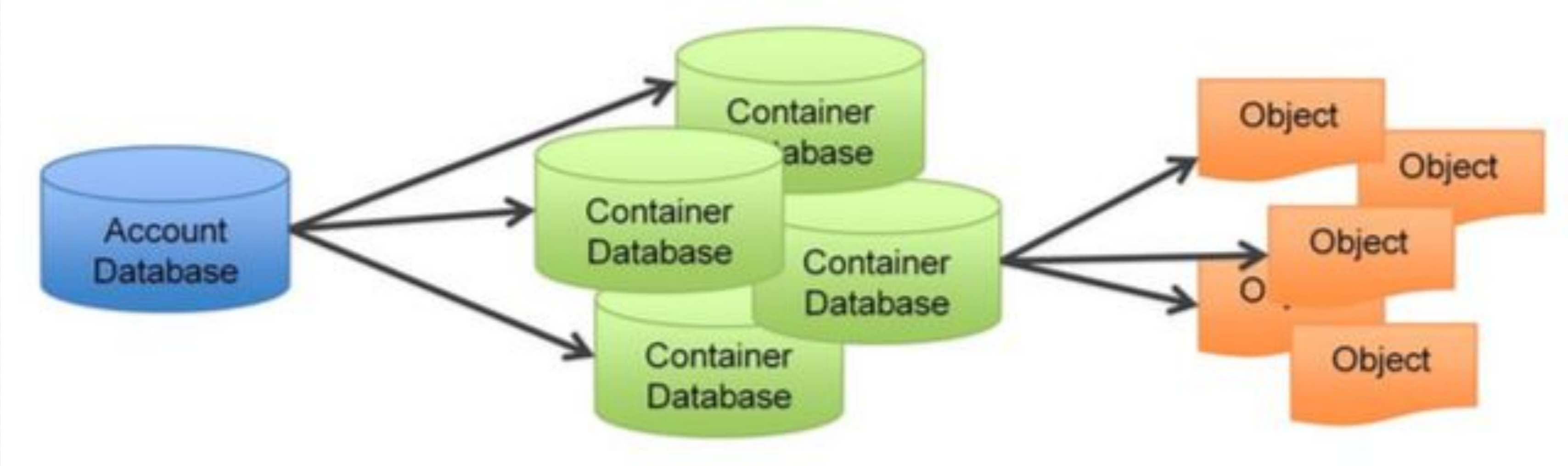
Swift Architecture contd.
Components

- **Proxy Servers**: handle all incoming API Requests.
- **Rings**: mapping between the logical location of data to locations on particular disks.
- **Zones**: represent a location that can isolate data. This could be a drive, a server, a cabinet, a switch, or even a datacenter.
- **Partition**: store Objects, Account databases and Container databases.

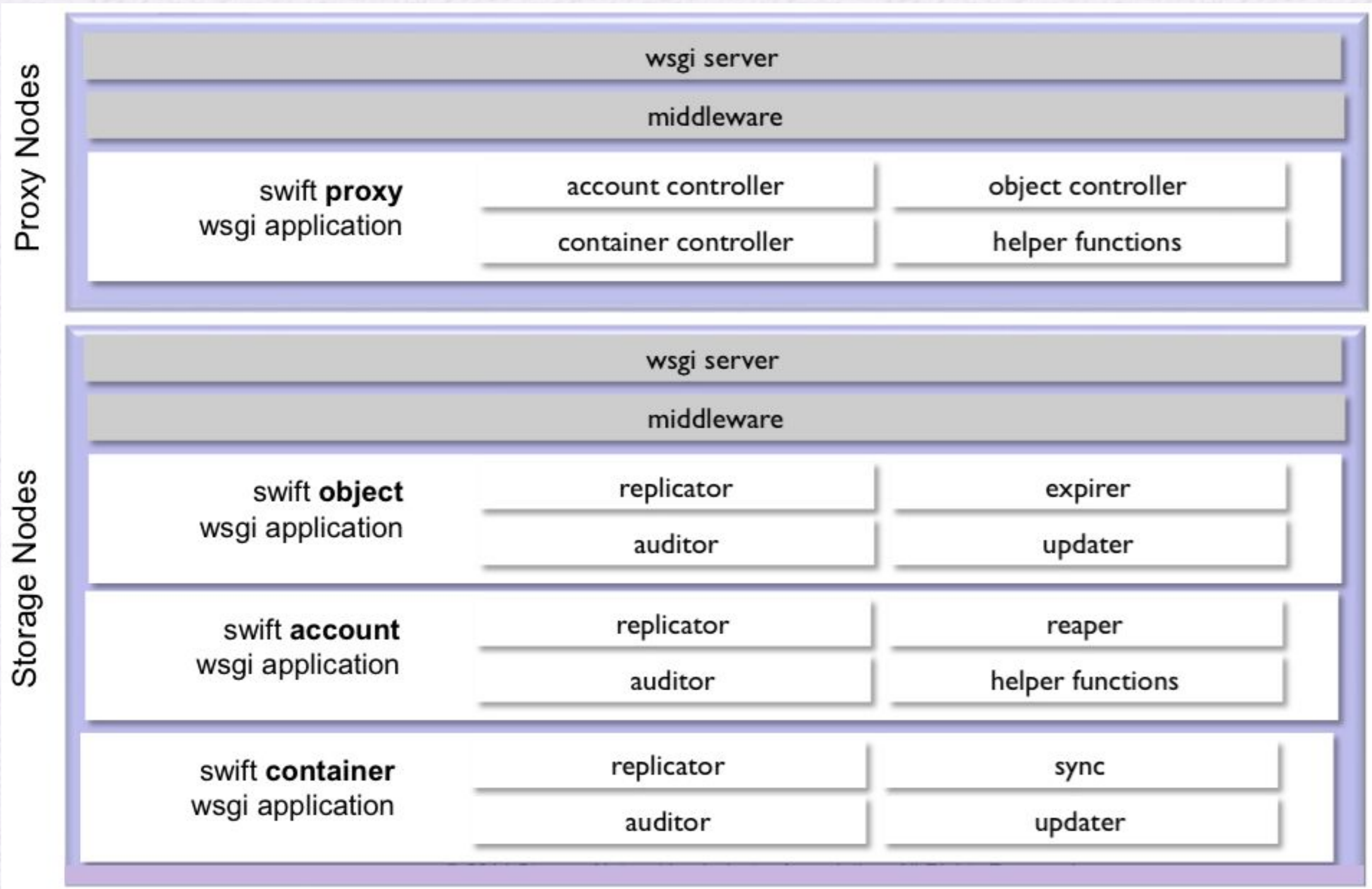
BRAC UNIVERSITY
Inspiring Excellence

Swift Architecture contd.
Components contd.

- **Account:** determines rights to **Containers and Objects.**
- **Account Database**: a list of all Containers spread throughout the cluster that are usable by an Account.
- **Container**: a logical storage space.
- **Container Database:** handles listings of what objects are in which container.
- **Object**: the data to be stored, such as documents or images.



Swift Architecture contd.
Services



Swift Architecture contd.
Consistency Process

- ***Auditor:***
  - Scan the disks on their node to ensure that the stored data has not suffered file system corruption.

- ***Replicator:***
  - Ensure that enough copies of the most recent version of the data are stored where they should be in the cluster.
  - Handle object and container deletions

- ***Account reaper:***
  - When the account reaper locates an account marked as deleted, it begins stripping out all objects and containers associated with the account, ultimately removing the account record itself.
  - To provide a buffer against error, the reaper can be configured with a delay so that it will wait for a specified period of time before it starts deleting data.

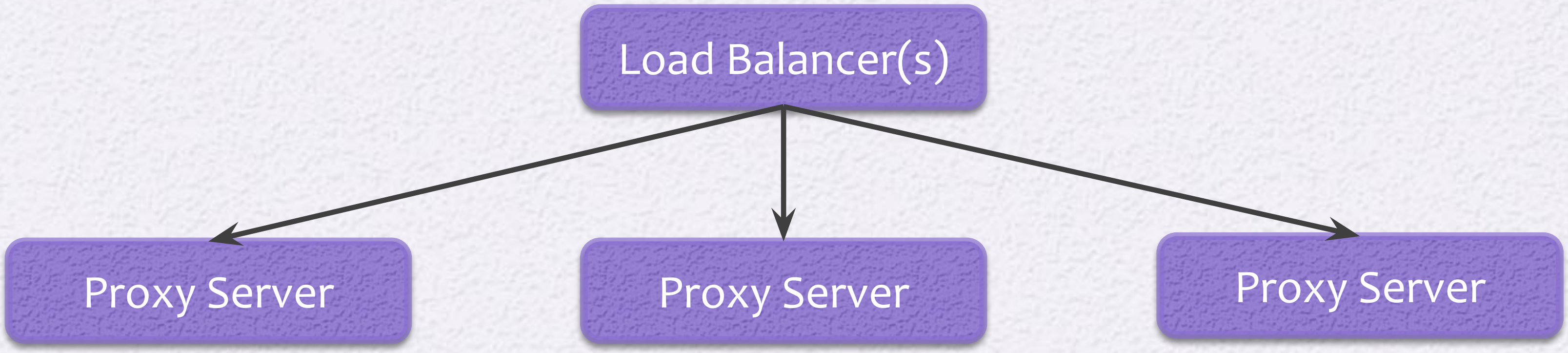
- ***Container and object updaters:***
  - Container updater consistency process is responsible for keeping the container listings in the accounts up-to-date. Additionally, it updates the object count, container count, and bytes used in the account metadata.
  - Object updater updates the container listing as well as the object count and bytes used in the container metadata.

Object expirer:

- Allows designated objects to be automatically deleted at a certain time.

BRAC UNIVERSITY
Inspiring Excellence

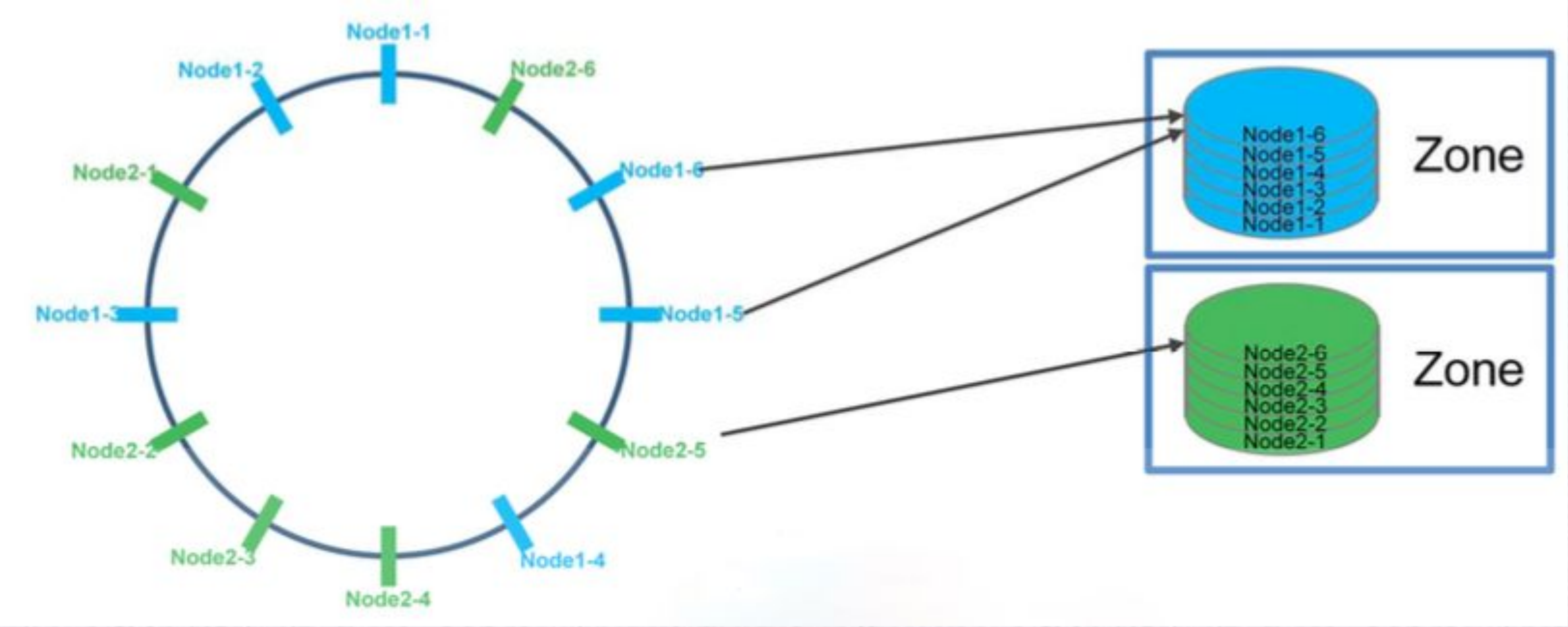
Swift Architecture contd.
Proxy Servers



- Handle all incoming API requests.
- Will determine the storage node based on the URL of the objects.
- Use a shared-nothing architecture and can be scaled as needed based on projected workloads.
- A large number of failures are also handled in the Proxy Server. For example, if a server is unavailable for an object PUT, it will ask the ring for a handoff server and route there instead.
- Also coordinates responses, and coordinates timestamps.

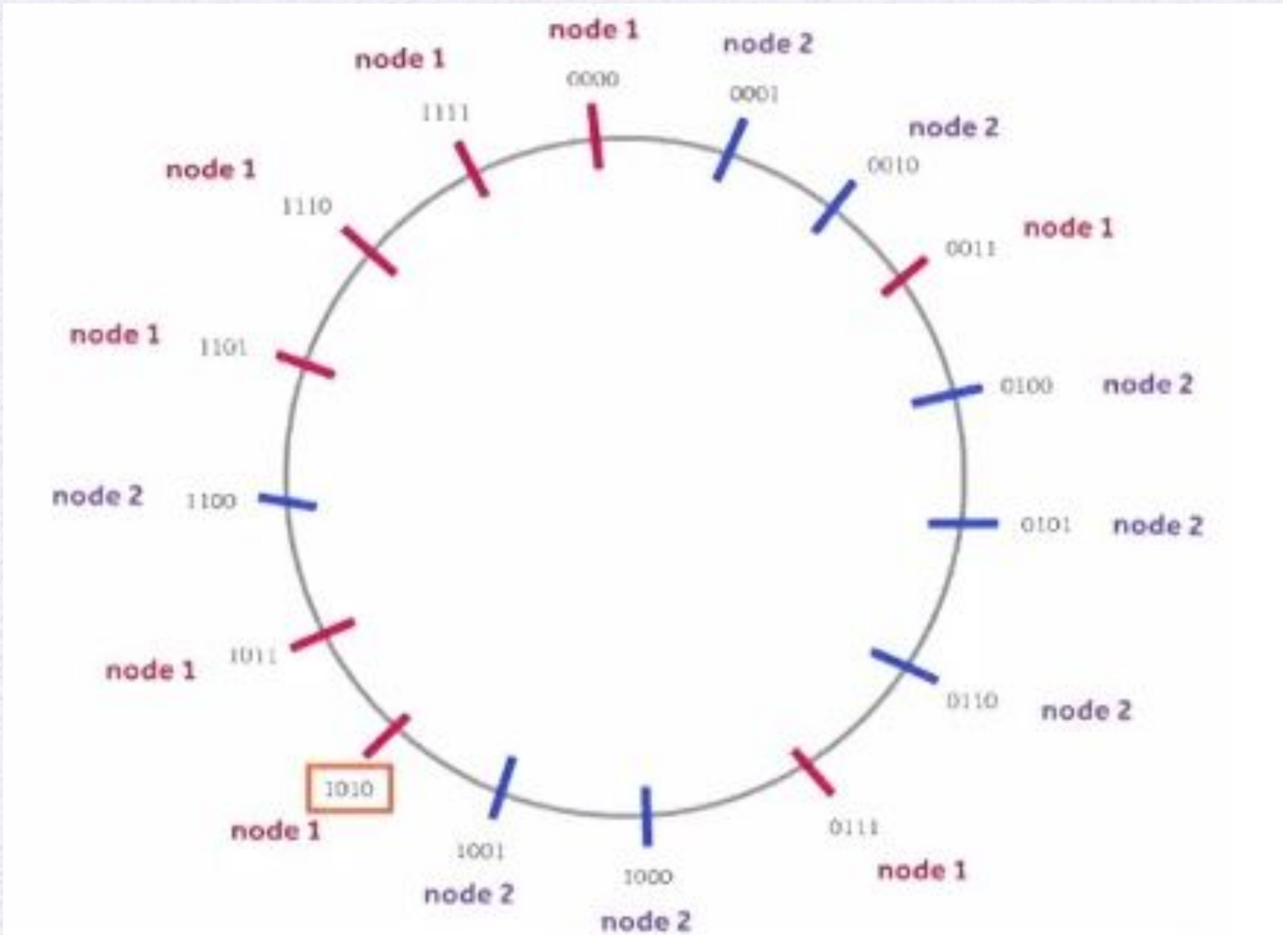
BRAC UNIVERSITY
Inspiring Excellence

Swift Architecture contd.
The Ring



- Mapping between entities stored and their physical locations on disk, an index.
- The Ring maintains this mapping with zones, devices, partitions and replicas.
- Each partition in the Ring is replicated three times across the cluster in different zones.
- Rings are built through consistent hashing algorithm. The ring is used by the Proxy server and several background processes (like replication).
- The ring is also responsible for determining which devices are used for handoff in failure scenarios.

Swift Architecture contd.
The Ring contd.



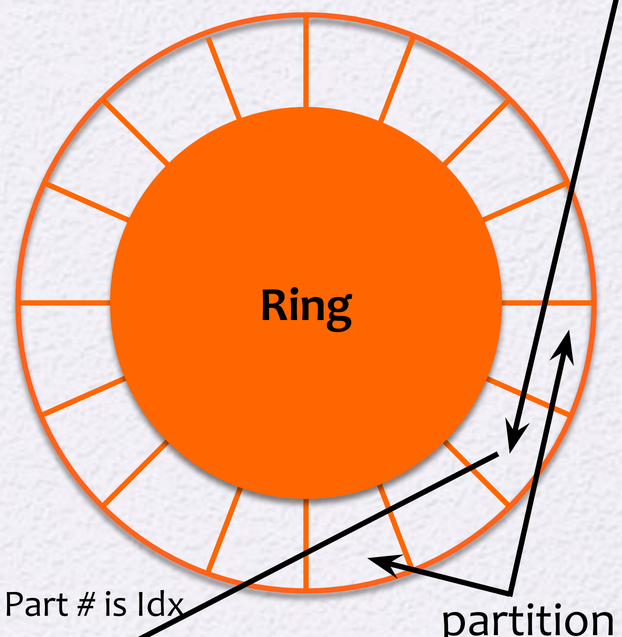
Hash(suffix + “v1/user1/container1/image.jpg” + prefix) = 1010 1011001101 … ➔ node 1

Swift Architecture contd.

The Ring contd.

- The ring is a static data structure maintained external to the cluster (tools provided)
- Devices are assigned to partitions with several policies (regions, zones, etc.) and constraints to assure fault tolerance and load balancing

Md5*(suffix + name + prefix) = index



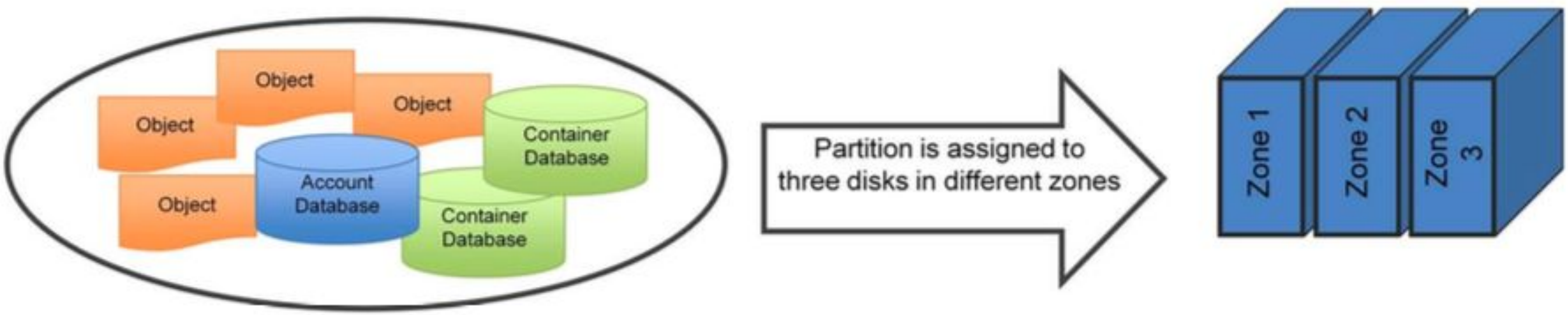
Idx	0	1	2	3	…
Device	Region 1 Zone 2 Weight 1 …	Region 1 Zone 2 Weight 1 …	Region 1 Zone 2 Weight 1 …	Region 1 Zone 2 Weight 1 …	…

lookup val is Idx

Idx	0	1	3	4	5	…
Copy 0	11	21	23	14	20	…
Copy 1	22	89	35	40	54	…
Copy 2	49	71	93	67	33	…

e.g.,
_replica2part2dev[2][7956] =
device of 3^{rd} replica of part 7956

Swift Architecture contd.
Partitions



- A partition is a collection of stored data, including Account, Container & Object Servers.
- The System Replicator and Object upload / download operate on Partitions.
- Num of partitions stored in one node = (num of replicas * total num of partions) / num of nodes.
- Choosing part_power :
  - Immutable once chosen.
  - At least 100 per disk
  - Bigger part_power -> bigger lookup table -> more memory used (but lookups stay fast)

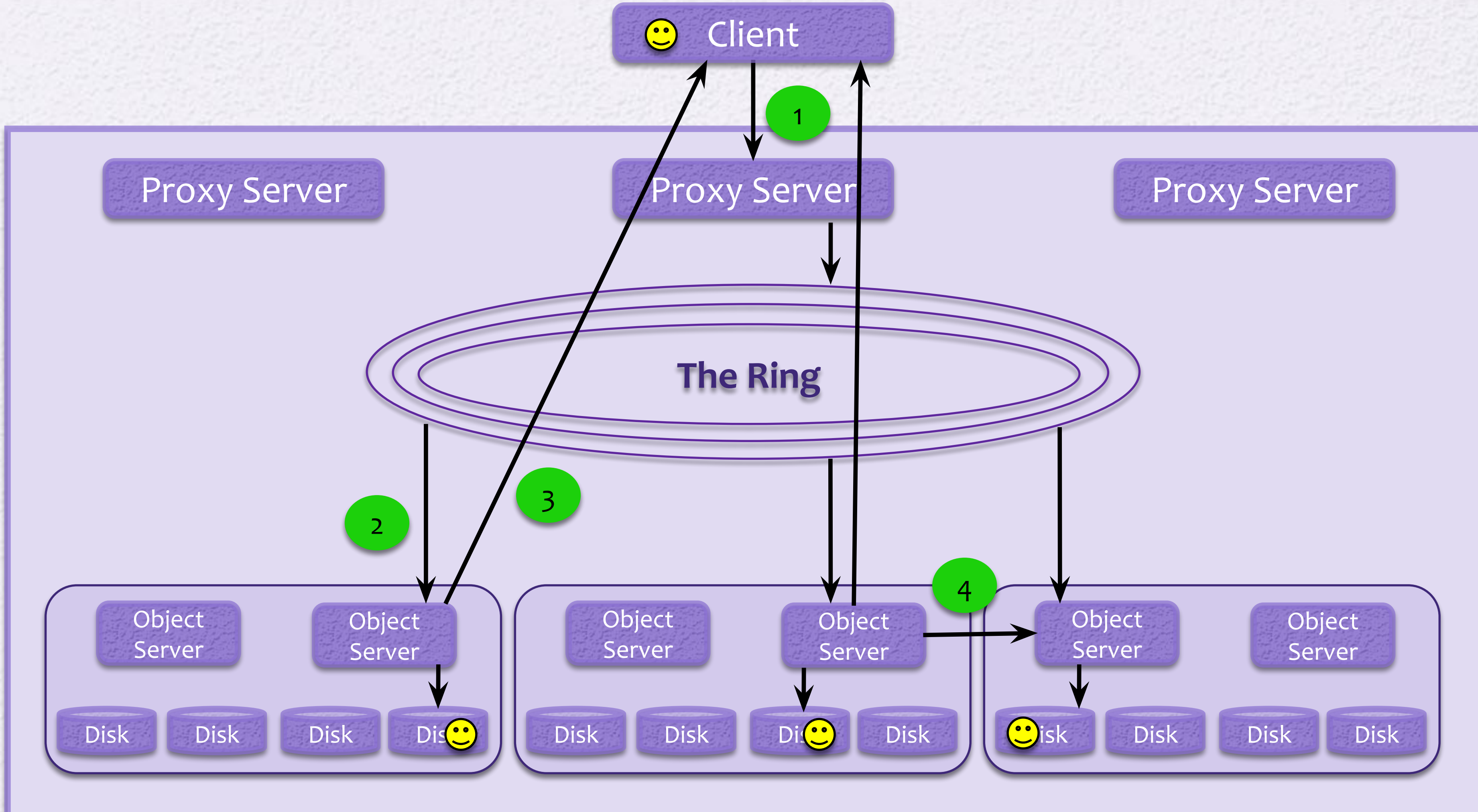
Swift API

- Simple Rest API
  - GET, POST, DELETE, HEAD, OPTIONS
- Simple Response Code
  - 2xx is good
  - 3xx redirect
  - 4xx is bad client
  - 5xx is bad servers etc
- Binding for different languages : python, ruby, java …
- Multiple CLI tools: python-swiftclient, jcloud, log
- Example of metadata (Headers) :

BRAC UNIVERSITY
Inspiring Excellence

Swift Architecture contd.

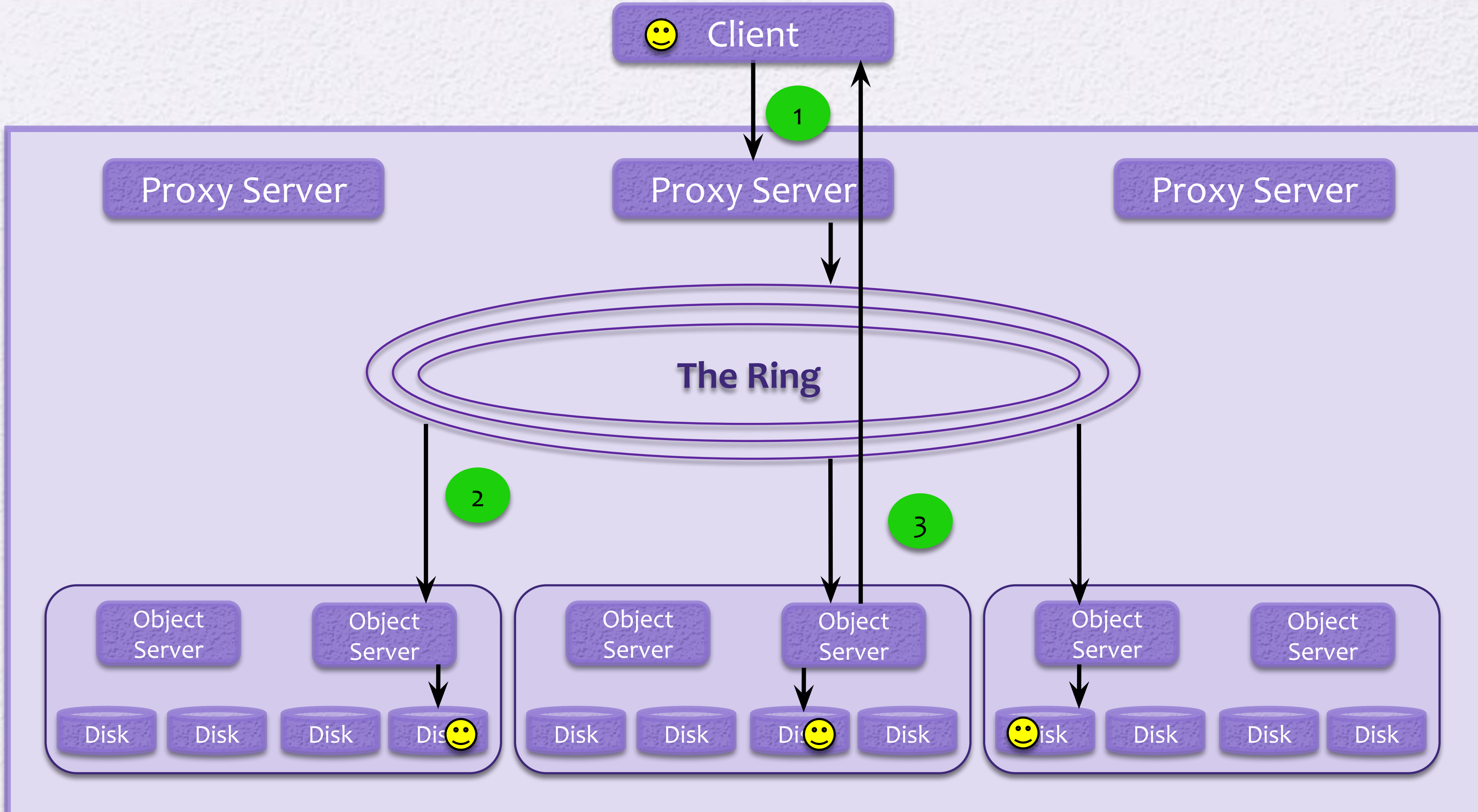
Process for a PUT



Inspiring Excellence

Swift Architecture contd.

Process for a GET



Swift Architecture contd.

Process for a DELETE