Cassandra Architecture is inspired by

amazon DynamoDB

Google BigTable

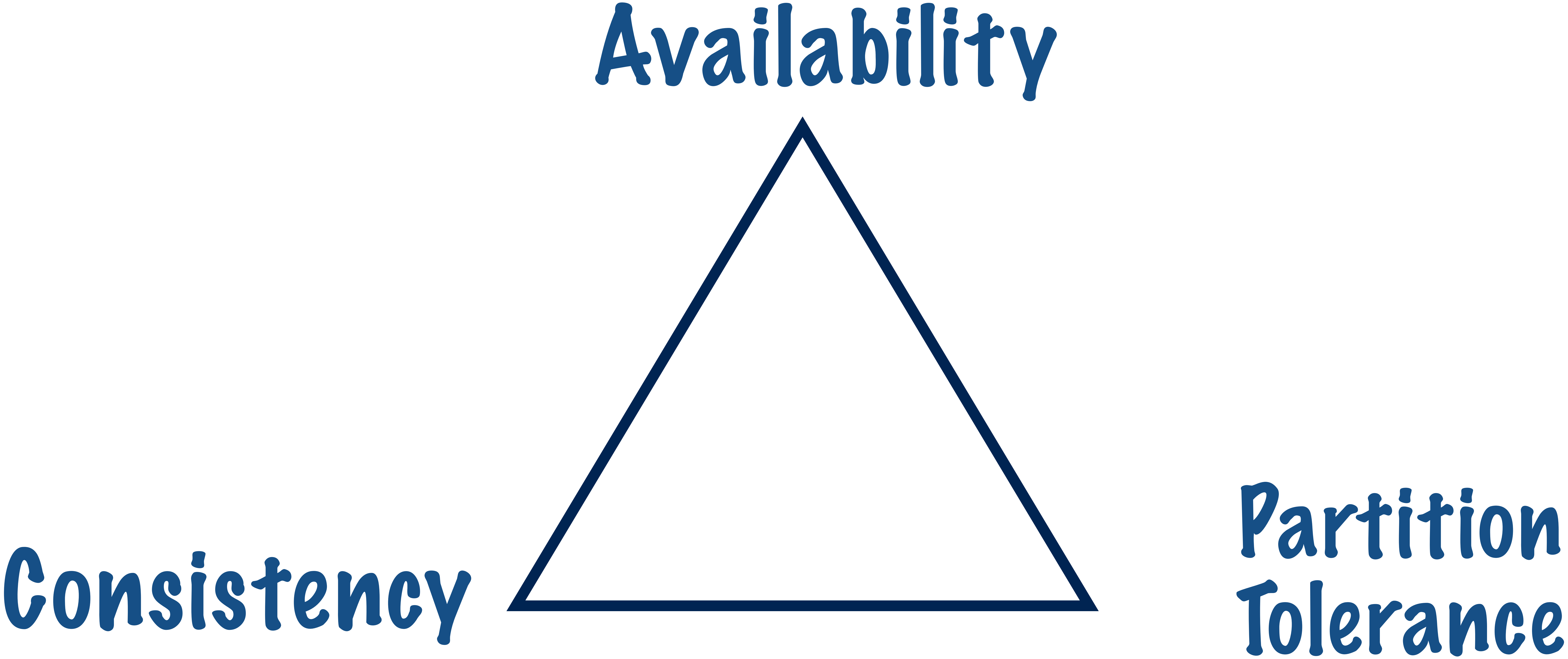
Cassandra

Log Structured ColumnFamily Data - Based on Google Big table

Partitioning and Replication - Based on Amazon DynamoDB

Cassandra

According to CAP theorem, a distributed system can only fulfil 2 of the following 3 properties



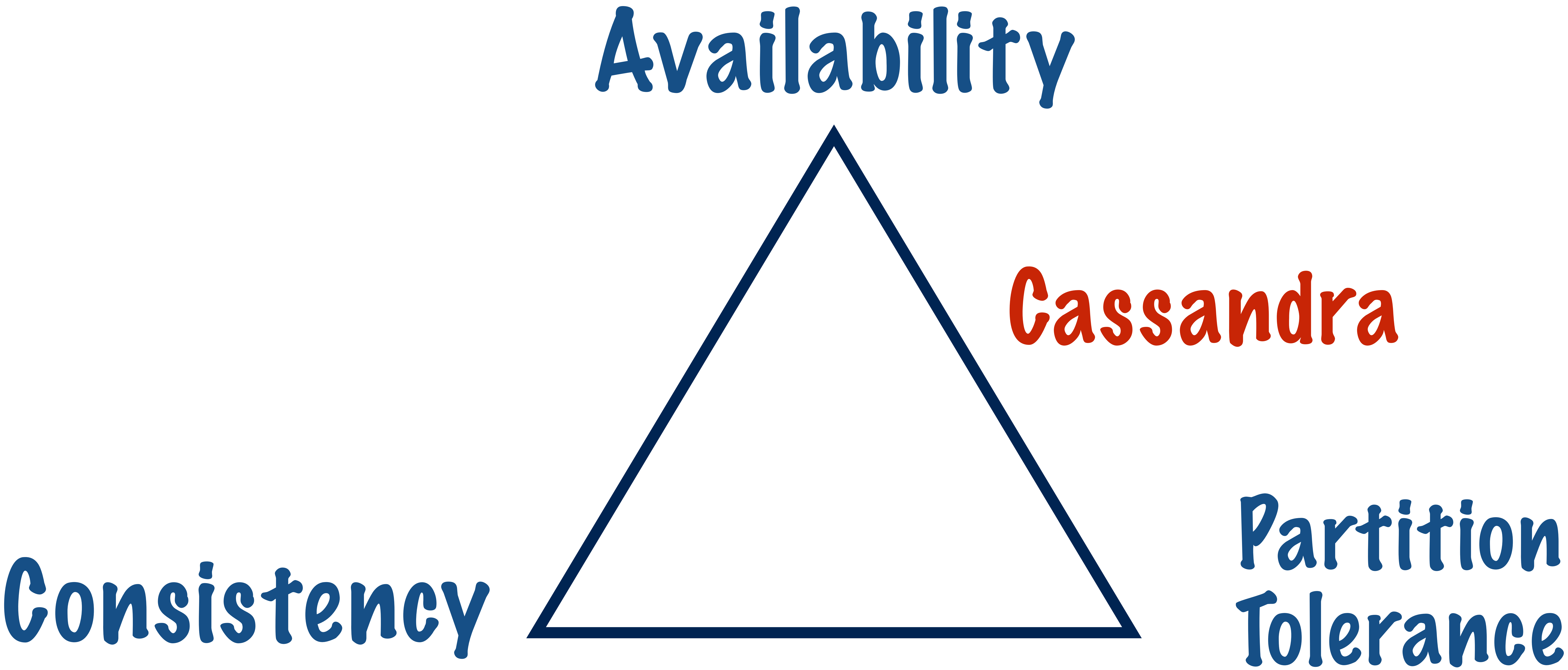
Availability

Consistency

Partition Tolerance

Cassandra

Cassandra is available and partition tolerant.
But it trades off consistency with performance



Availability

Cassandra

Consistency

Partition Tolerance

Let's understand consistency in Cassandra

CONSISTENCY

Consistency ensures that data read from any node in the cluster is the same i.e consistent

CONSISTENCY

Can configure separate consistency levels for READ and WRITE

CONSISTENCY

WRITE

number of replica nodes on which the write must succeed before returning success to the client

READ

number of replica nodes to check before returning data to client

CONSISTENCY

WRITE

ONE

ALL

QUORUM

LOCAL_QUORUM

CONSISTENCY

WRITE

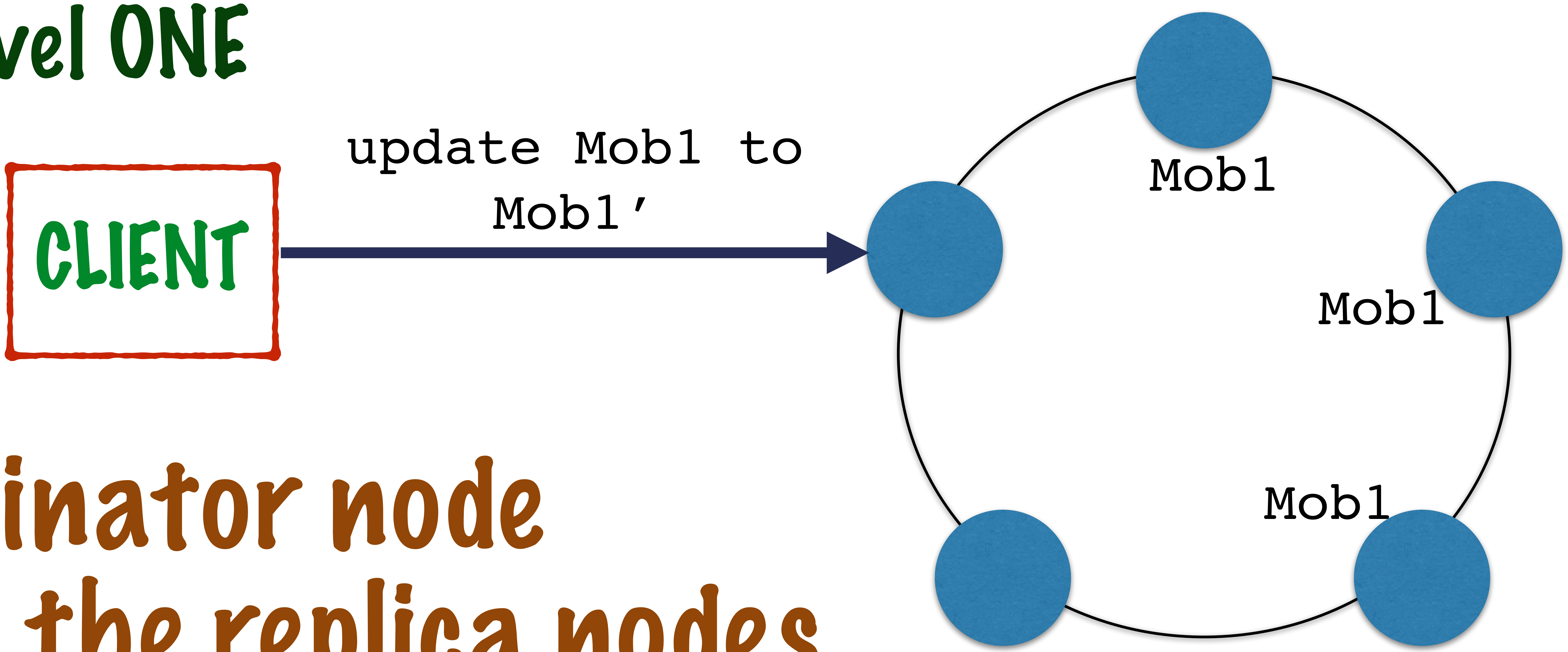
ONE

Only one replica needs to be updated and then the write operation returns a success

CONSISTENCY

WRITE

Consistency Level ONE

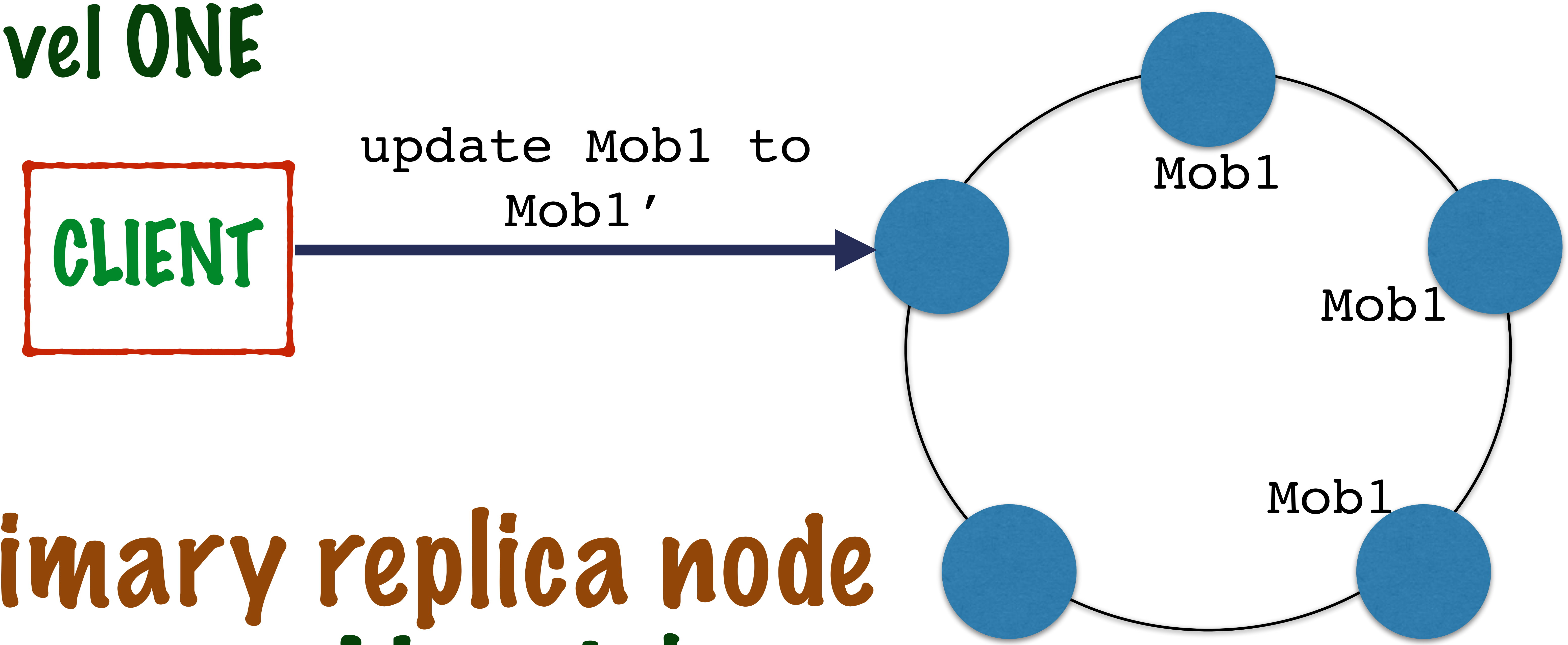


coordinator node determines the replica nodes with the use of token and ReplicaPlacementStrategy

CONSISTENCY

WRITE

Consistency Level ONE



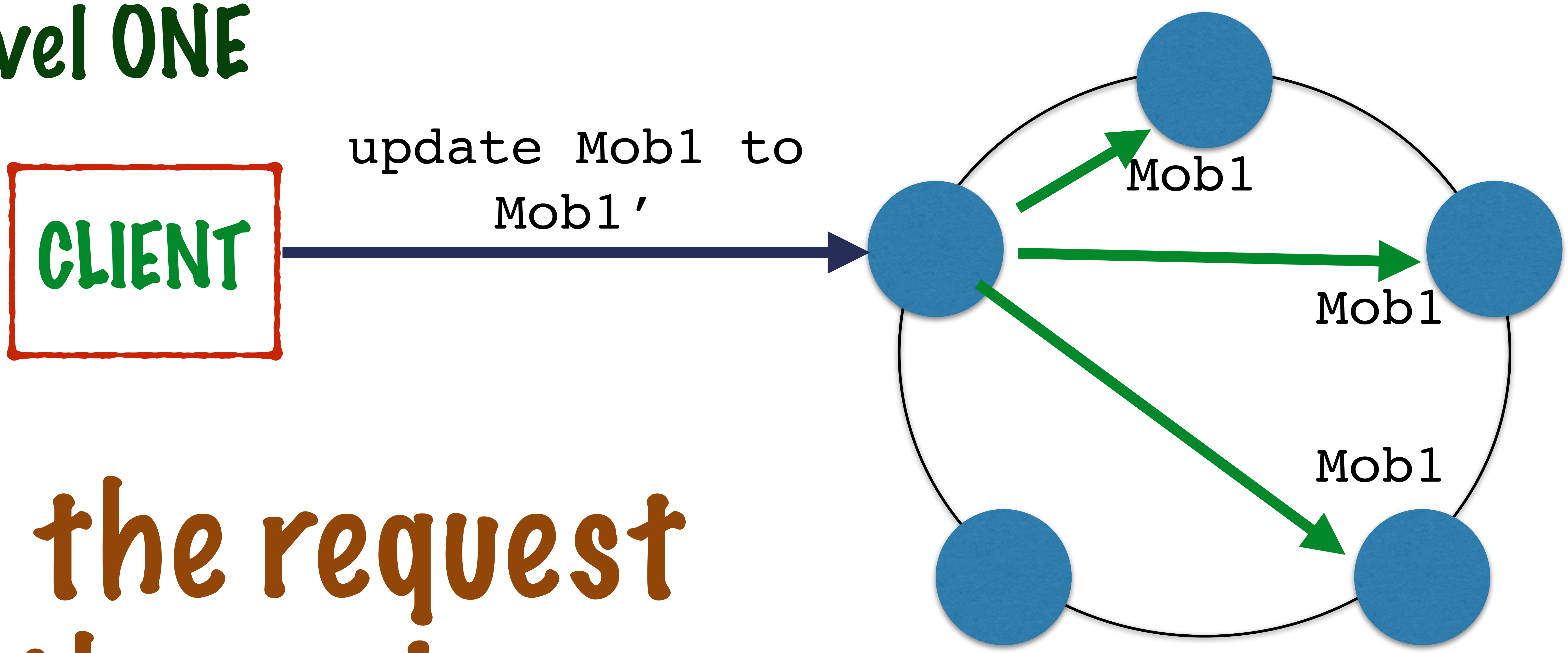
token - primary replica node

replicaPlacementAlgorithm - remaining replica nodes

CONSISTENCY

WRITE

Consistency Level ONE

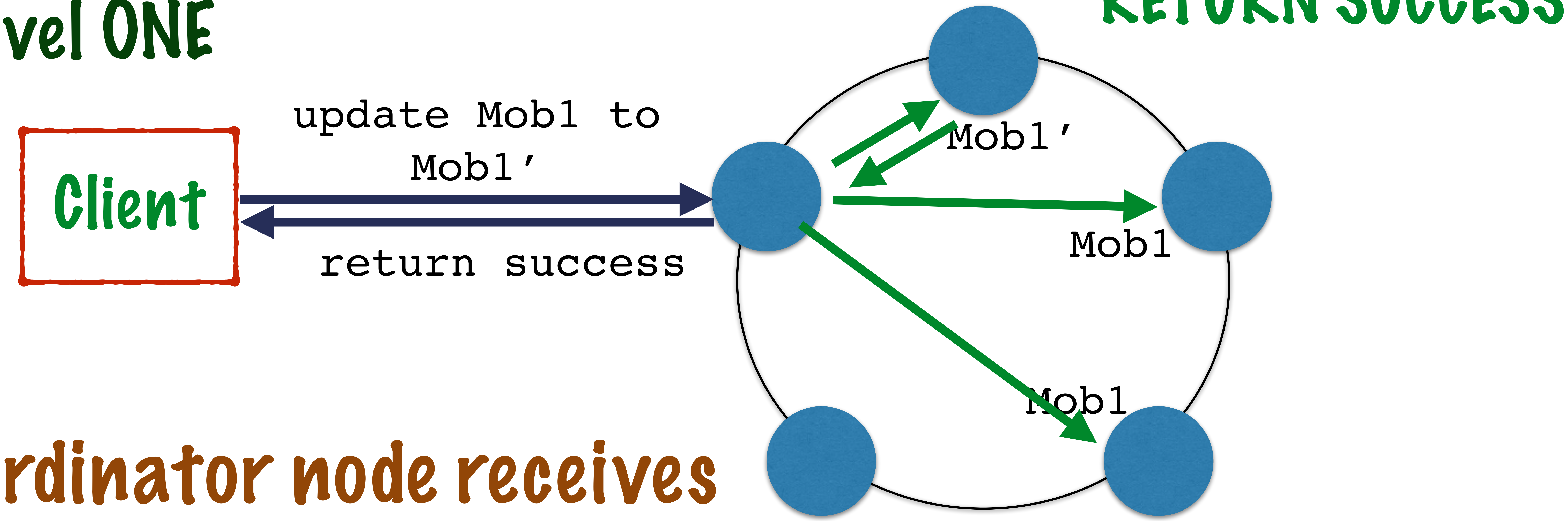


It sends the request to all the nodes simultaneously

CONSISTENCY

WRITE

Consistency Level ONE



As soon as coordinator node receives success response from **ONE NODE**, it returns success to the client

CONSISTENCY

WRITE

ONE

ALL

QUORUM

LOCAL_QUORUM

CONSISTENCY

WRITE

ALL

All replicas need to be updated and then the write operation returns a success

CONSISTENCY

WRITE

ONE

ALL

QUORUM

LOCAL_QUORUM

CONSISTENCY

WRITE

QUORUM

A **minimum** number of replicas (a **quorum**) needs to be updated for the write operation to return a success

CONSISTENCY

WRITE

Consistency Level QUORUM

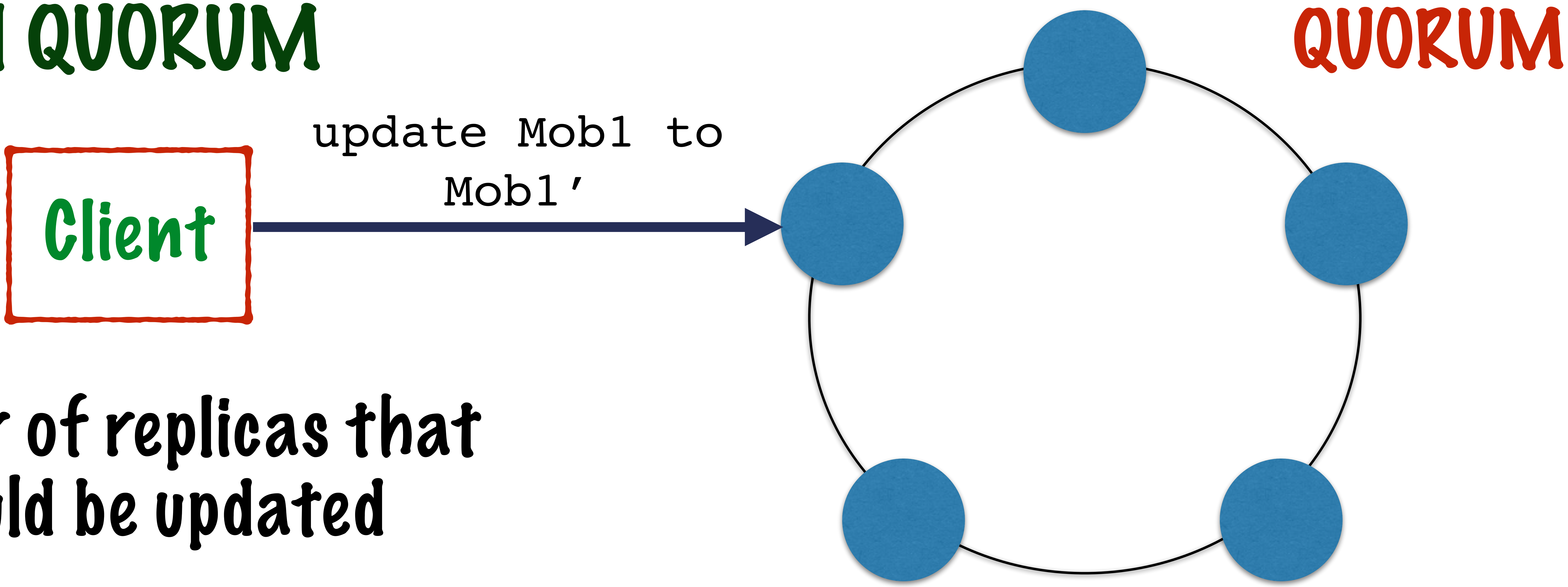
minimum number of people required to attend the meeting so the meeting can be conducted

QUORUM

CONSISTENCY

WRITE

Consistency Level QUORUM



Number of replicas that should be updated

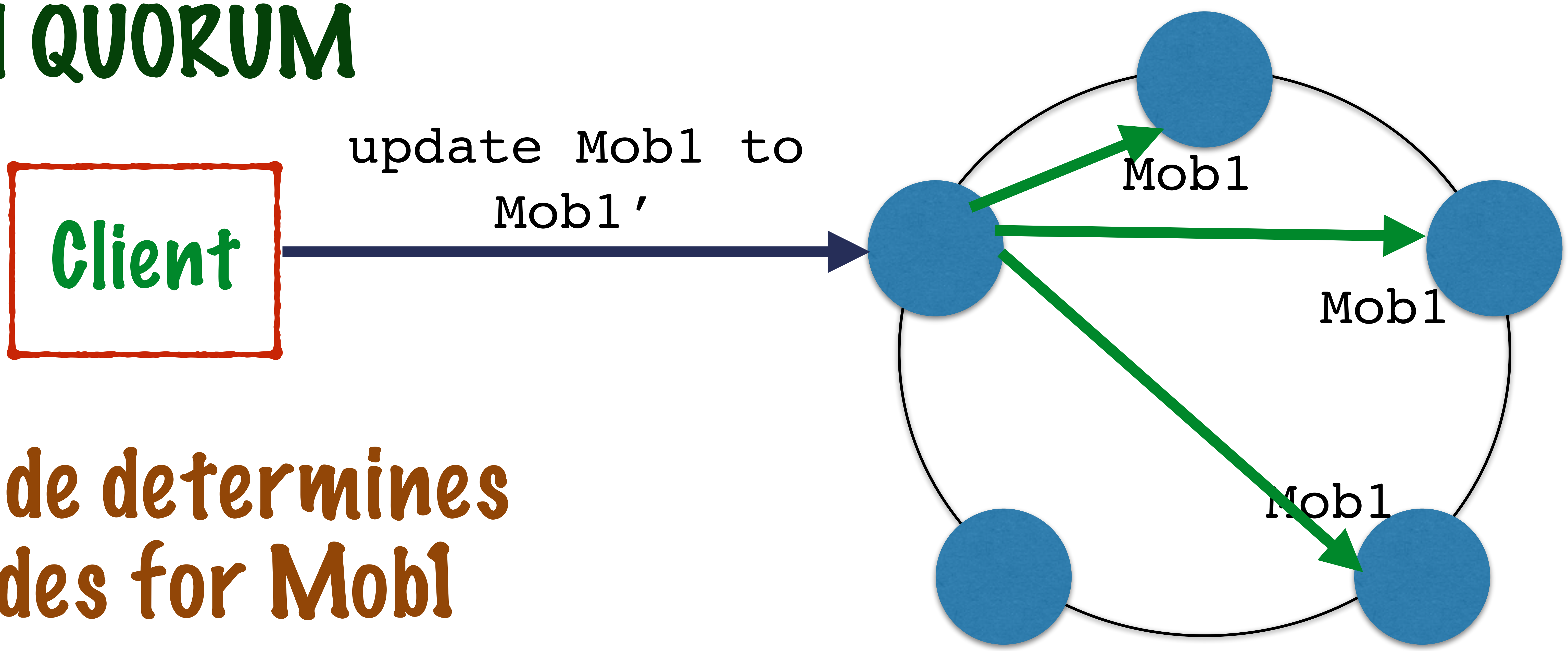
after which we return success to client

CONSISTENCY

WRITE

Consistency Level QUORUM

quorum = 2



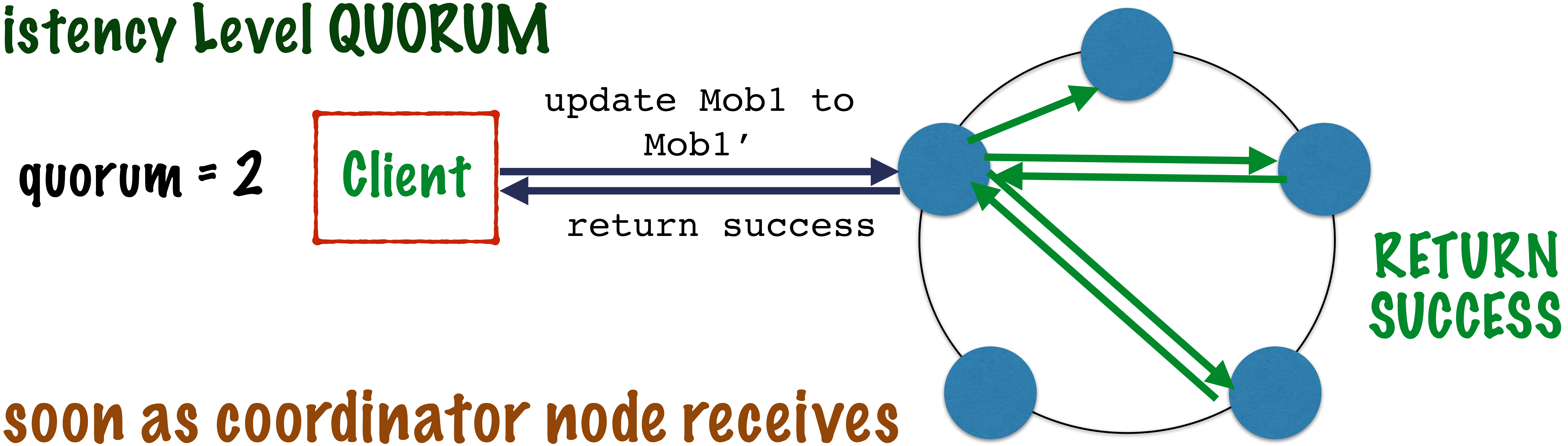
Coordinator node determines the replica nodes for Mob1

sends the request to all the nodes simultaneously

CONSISTENCY

WRITE

Consistency Level QUORUM

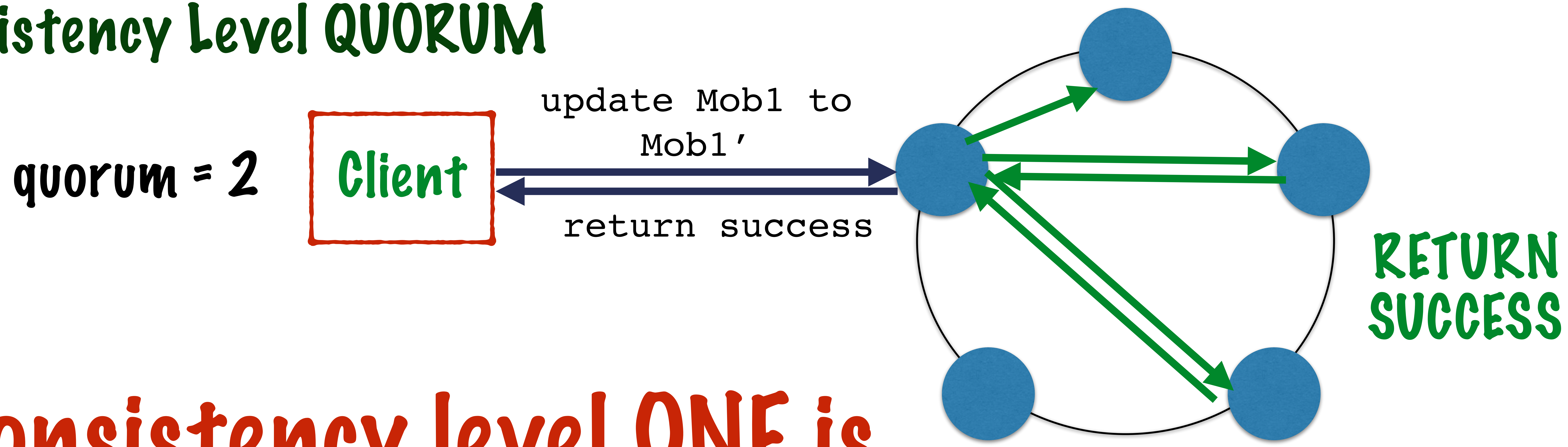


As soon as coordinator node receives response from the quorum (2) nodes, it returns a response to the client

CONSISTENCY

WRITE

Consistency Level QUORUM



Consistency level ONE is simply a quorum of 1!

CONSISTENCY

WRITE

ONE

ALL

QUORUM

LOCAL_QUORUM

CONSISTENCY

WRITE

LOCAL_QUORUM

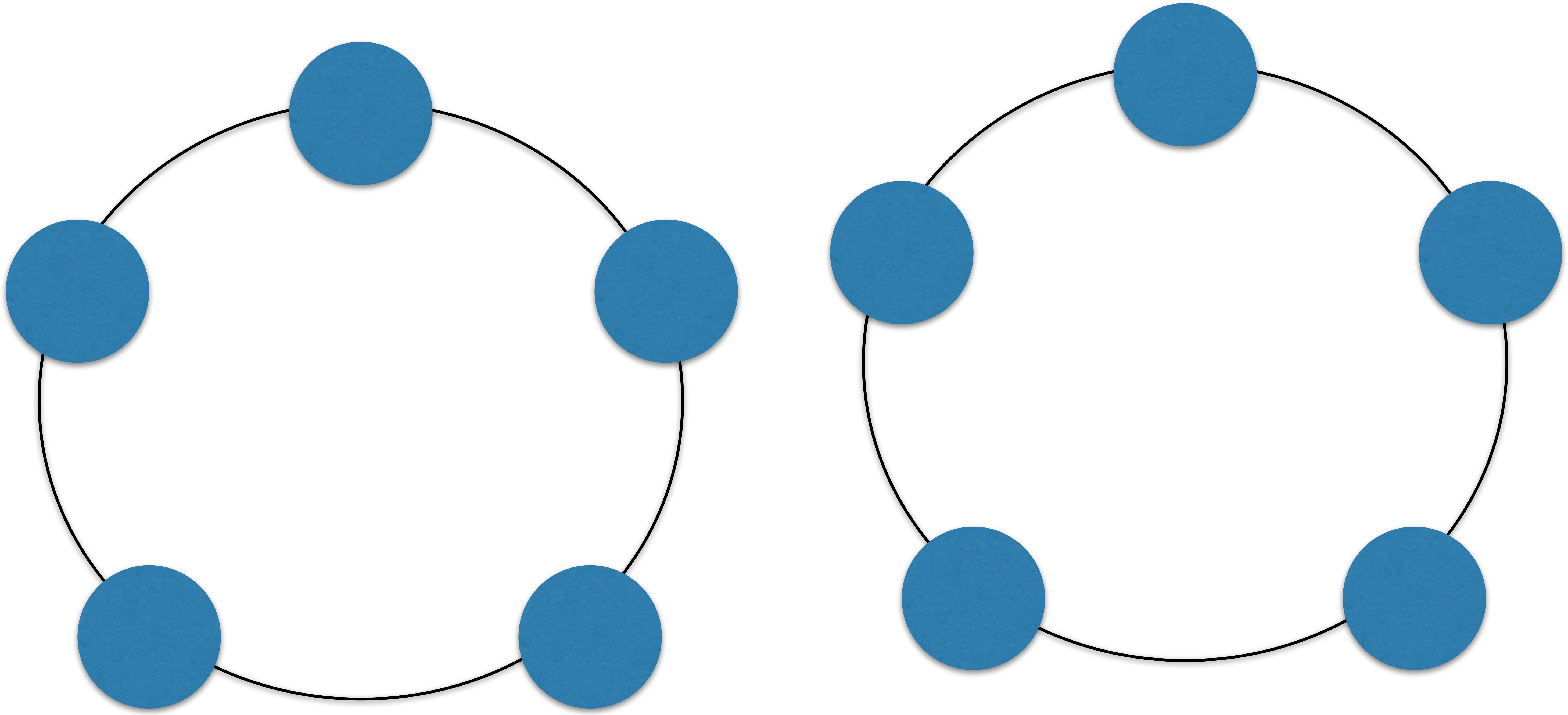
A **minimum** number of replicas (a **quorum**) needs to be updated **per datacenter** for the write operation to return a success

CONSISTENCY

WRITE

Consistency level LOCAL_QUORUM

Used for multiple datacenters

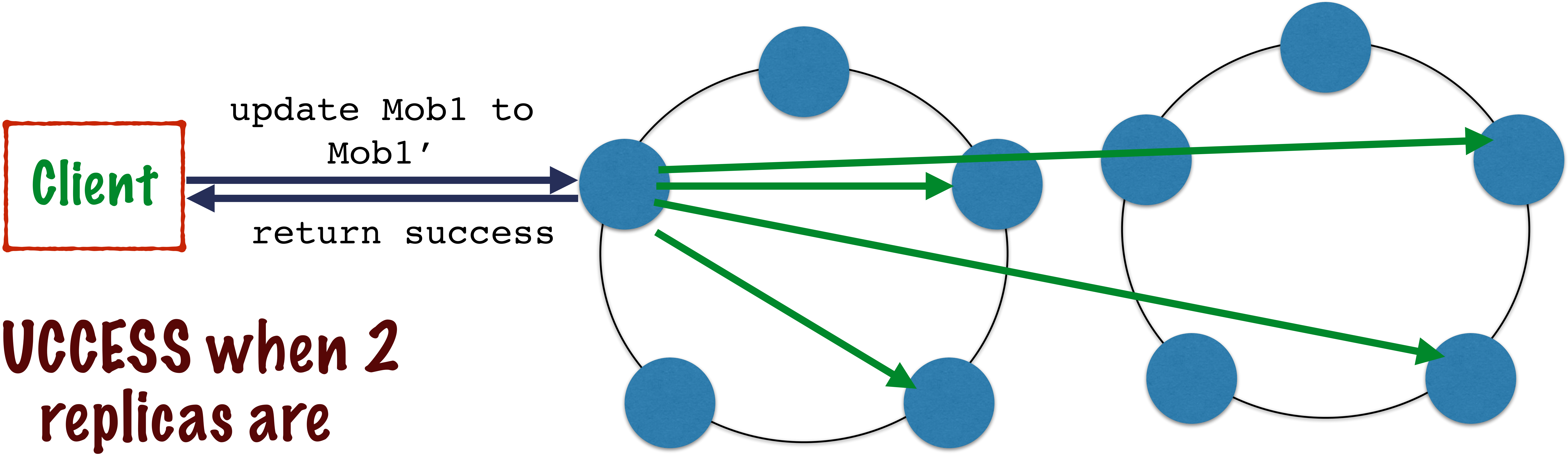


CONSISTENCY

WRITE

Consistency level
LOCAL_QUORUM

quorum = 2



SUCCESS when 2 replicas are updated in each datacenter

CONSISTENCY

WRITE

ONE

ALL

QUORUM

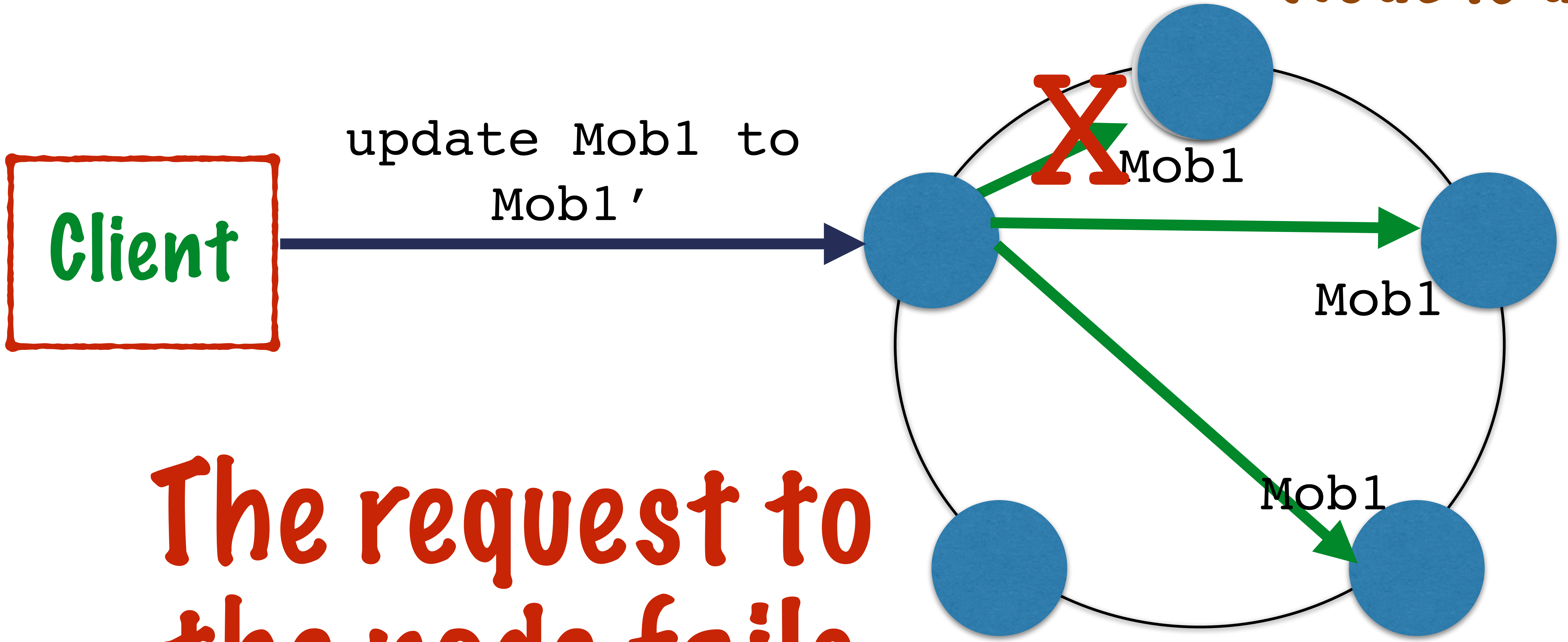
LOCAL_QUORUM

CONSISTENCY

What happens if the replica node is down?

CONSISTENCY

What happens if the replica node is down?



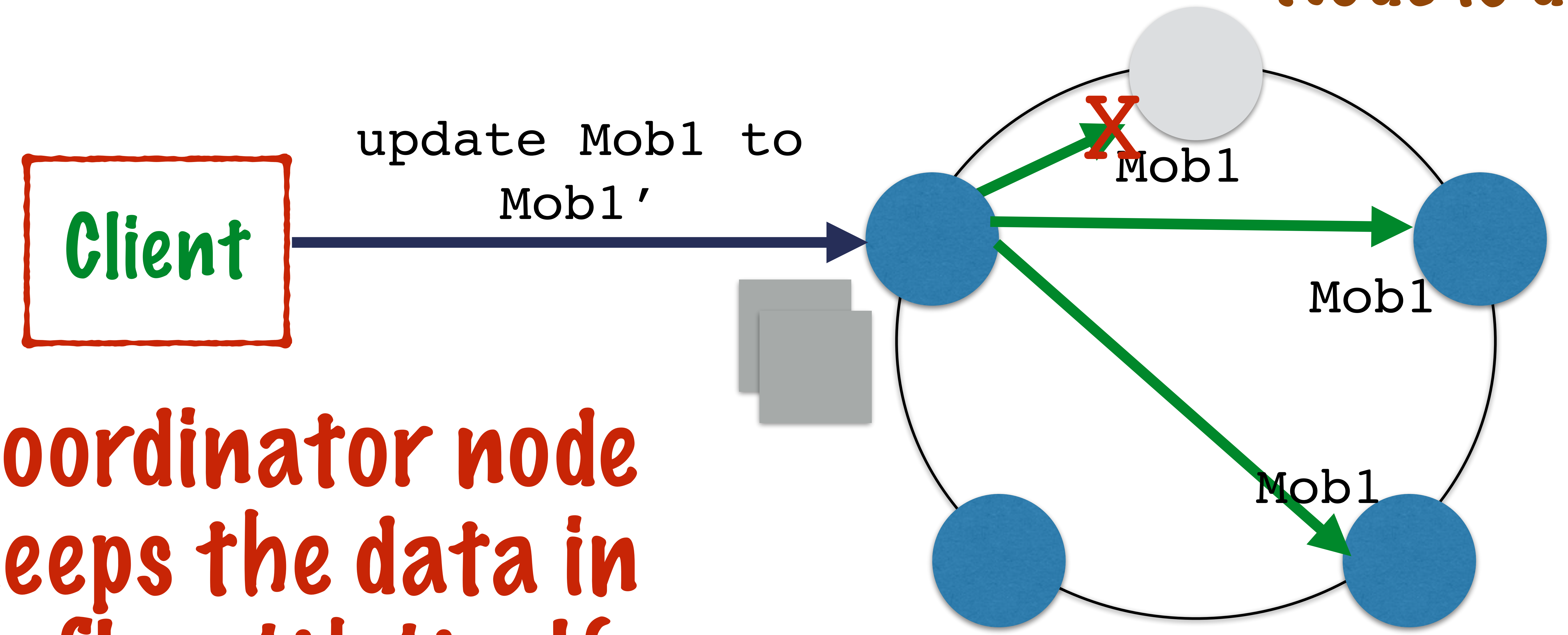
Node is down

The request to the node fails

CONSISTENCY

What happens if the replica node is down?

Node is down



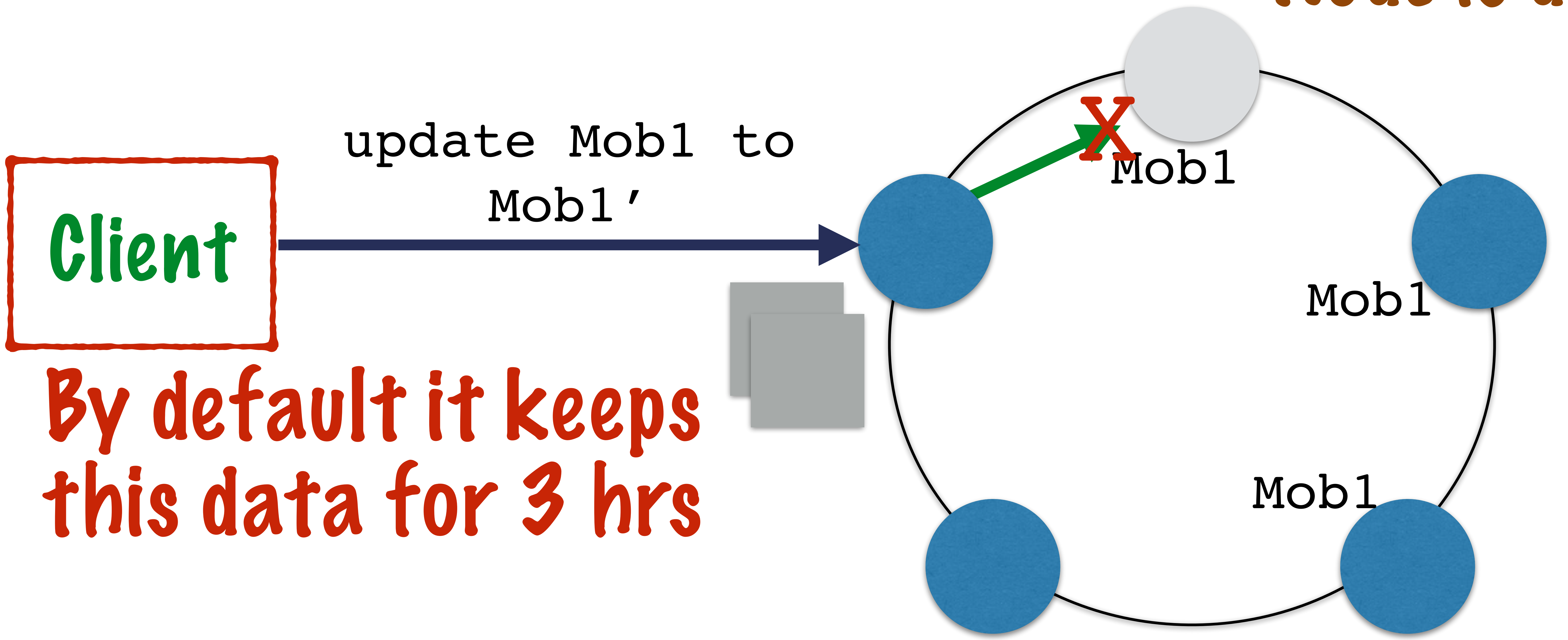
Coordinator node keeps the data in a file with itself

the file is called hint file

CONSISTENCY

What happens if the replica node is down?

Node is down

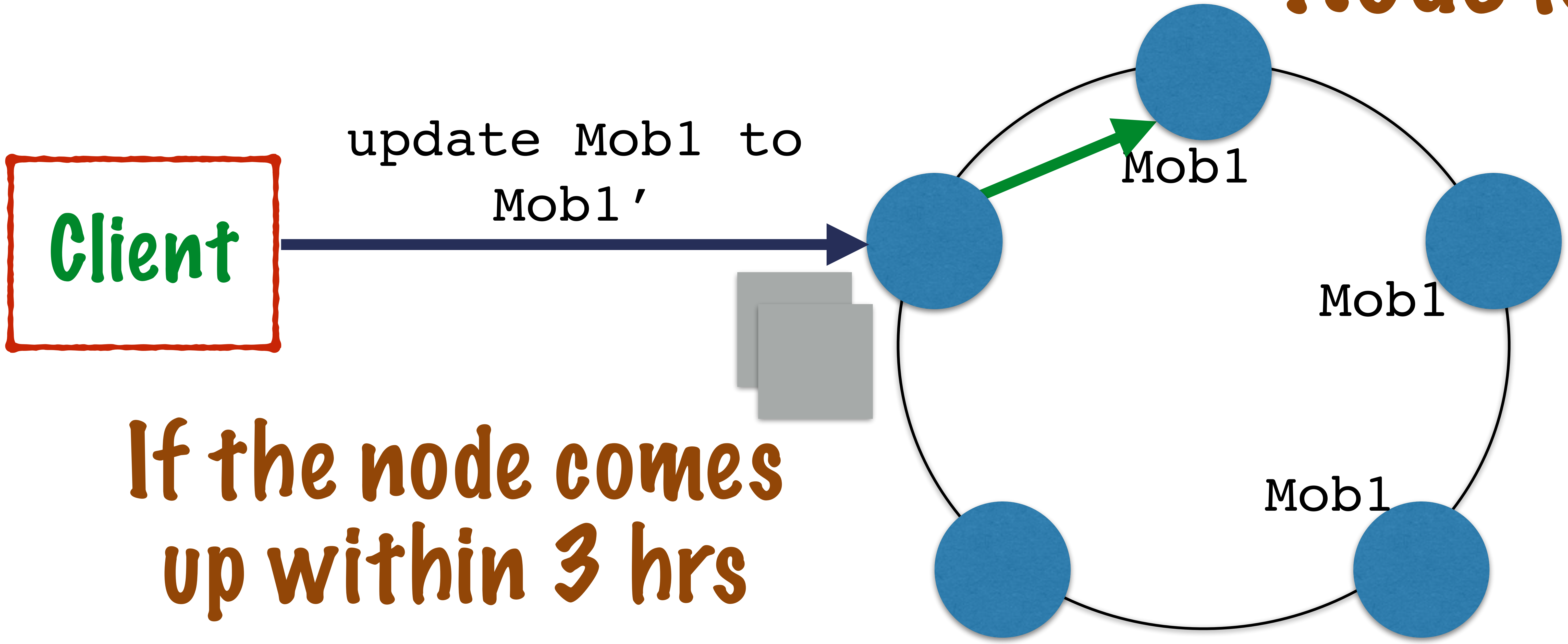


By default it keeps this data for 3 hrs

This time interval is configurable

CONSISTENCY

What happens if the replica node is down?



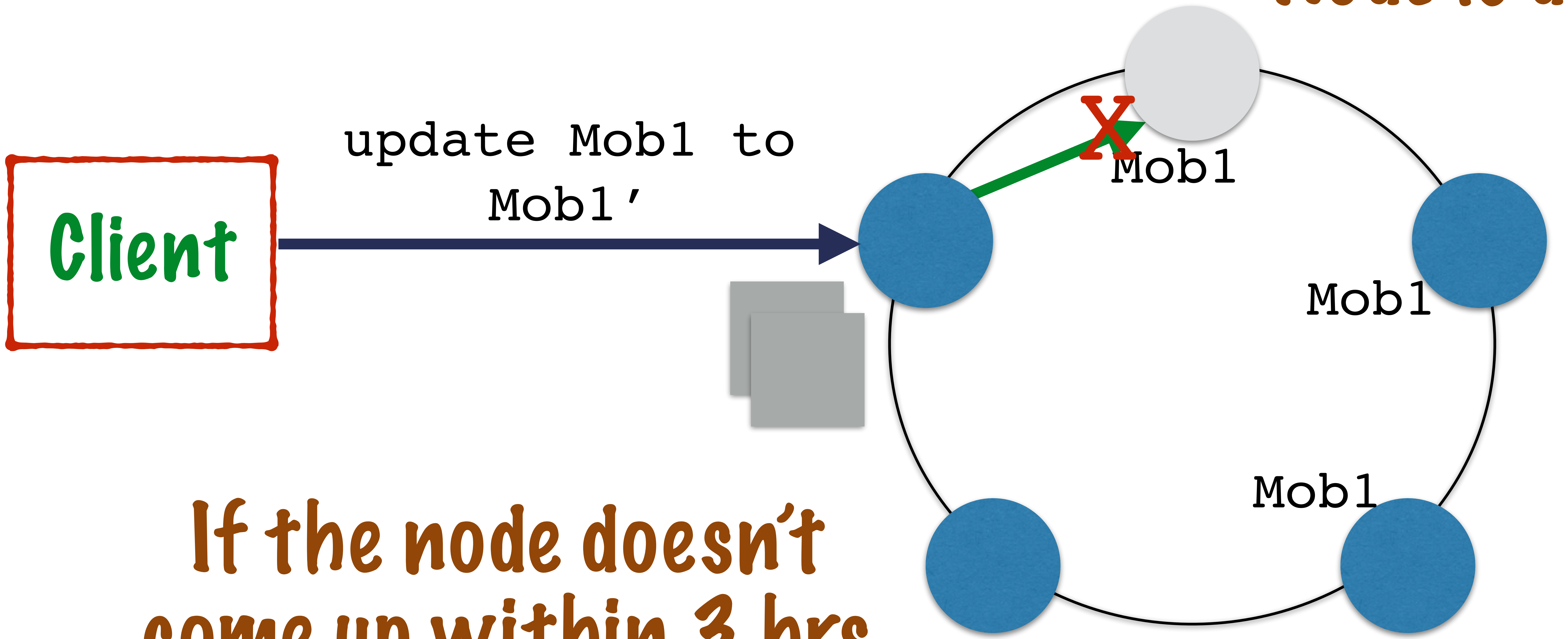
Node is up

If the node comes up within 3 hrs

relay the requests in the file to the node

CONSISTENCY

What happens if the replica node is down?



Node is down

If the node doesn't come up within 3 hrs

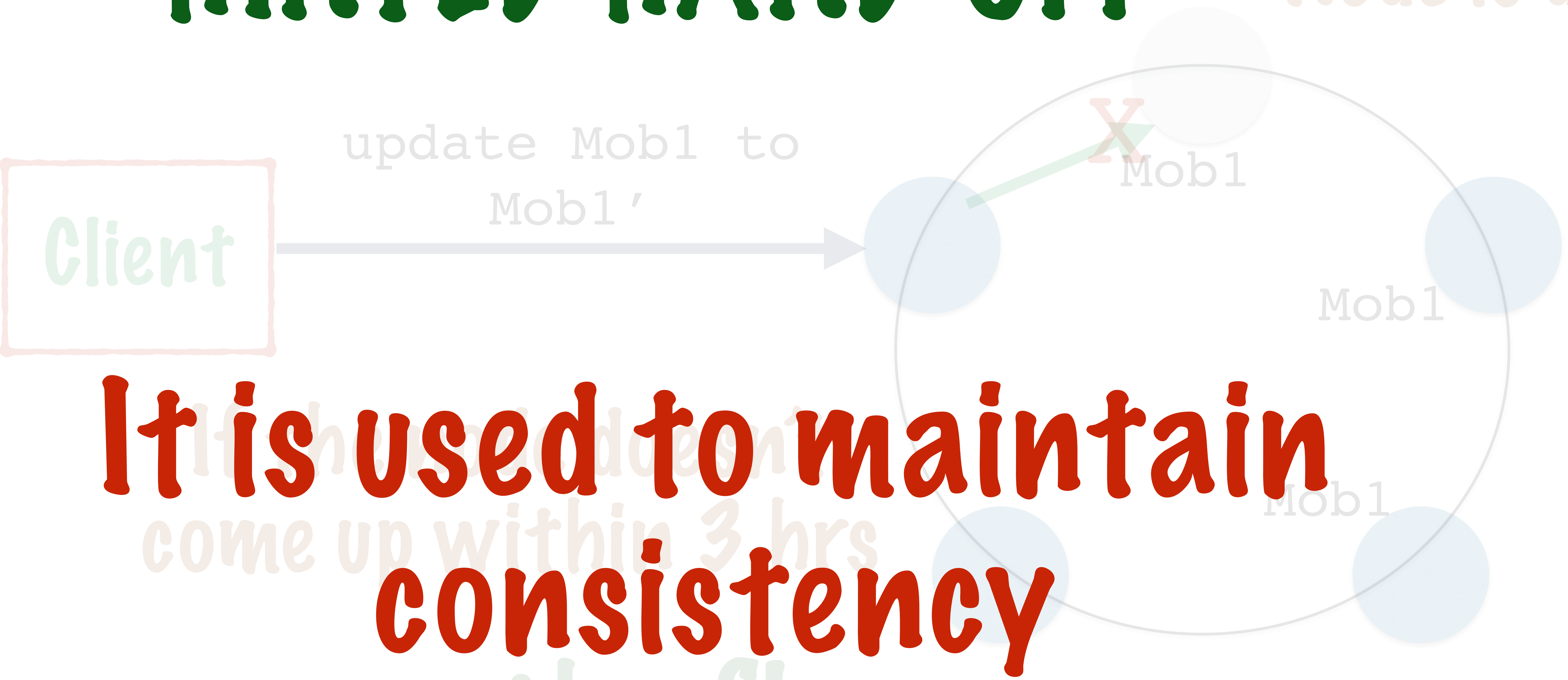
purge the file

CONSISTENCY

This mechanism is called

HINTED HAND OFF

Node is down



It is used to maintain consistency

come up within 3 hrs

purge the file

CONSISTENCY

Depending on the consistency level specified the request will return SUCCESS or ERROR to the user

Consistency level ALL will return failure for the write request

What happens if the replica node is down

Node is down

Client
update Mob1 to Mob1
X
Mob1
Mob1
Mob1

If the node doesn't come up within 3 hrs purge the file

CONSISTENCY

Lets see the read consistency levels

CONSISTENCY

READ

ONE

ALL

QUORUM

LOCAL_QUORUM

CONSISTENCY

READ

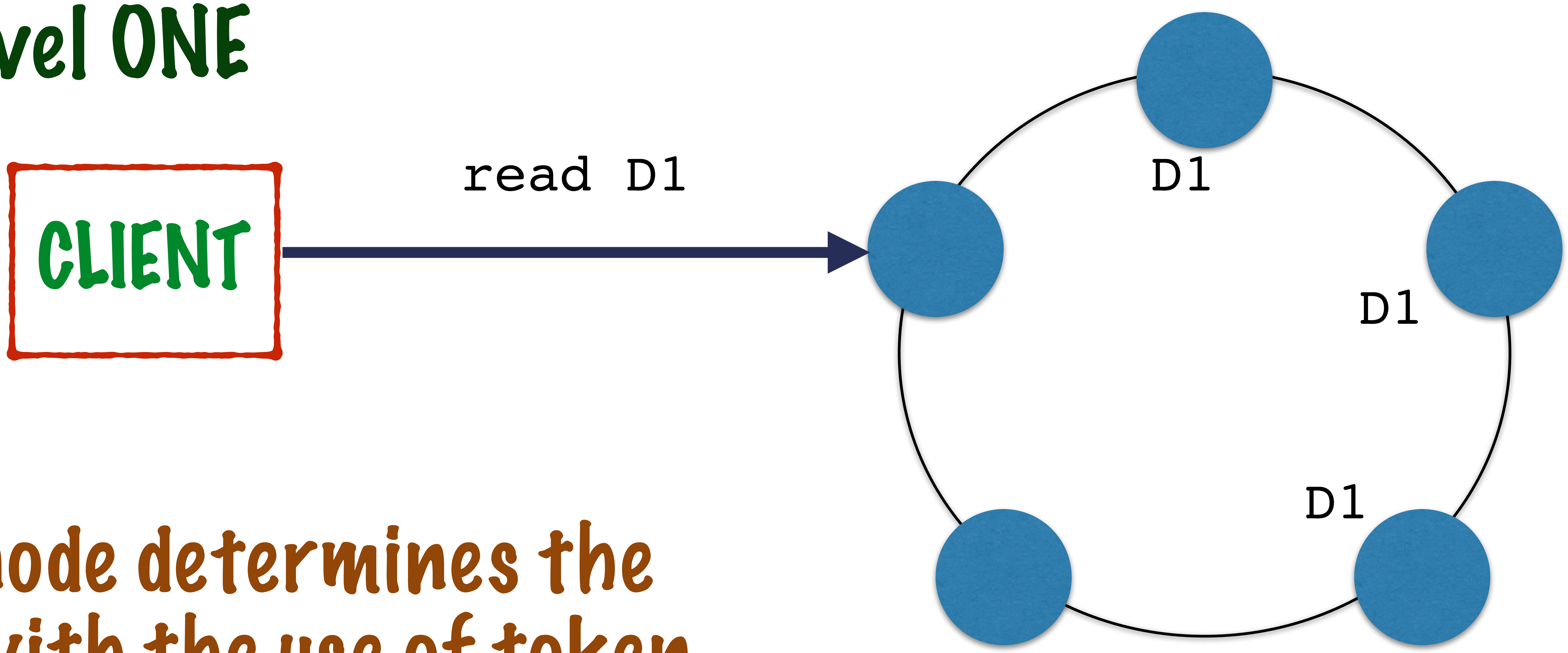
ONE

Only one replica needs to be read and then the read operation returns a success

CONSISTENCY

READ

Consistency Level ONE

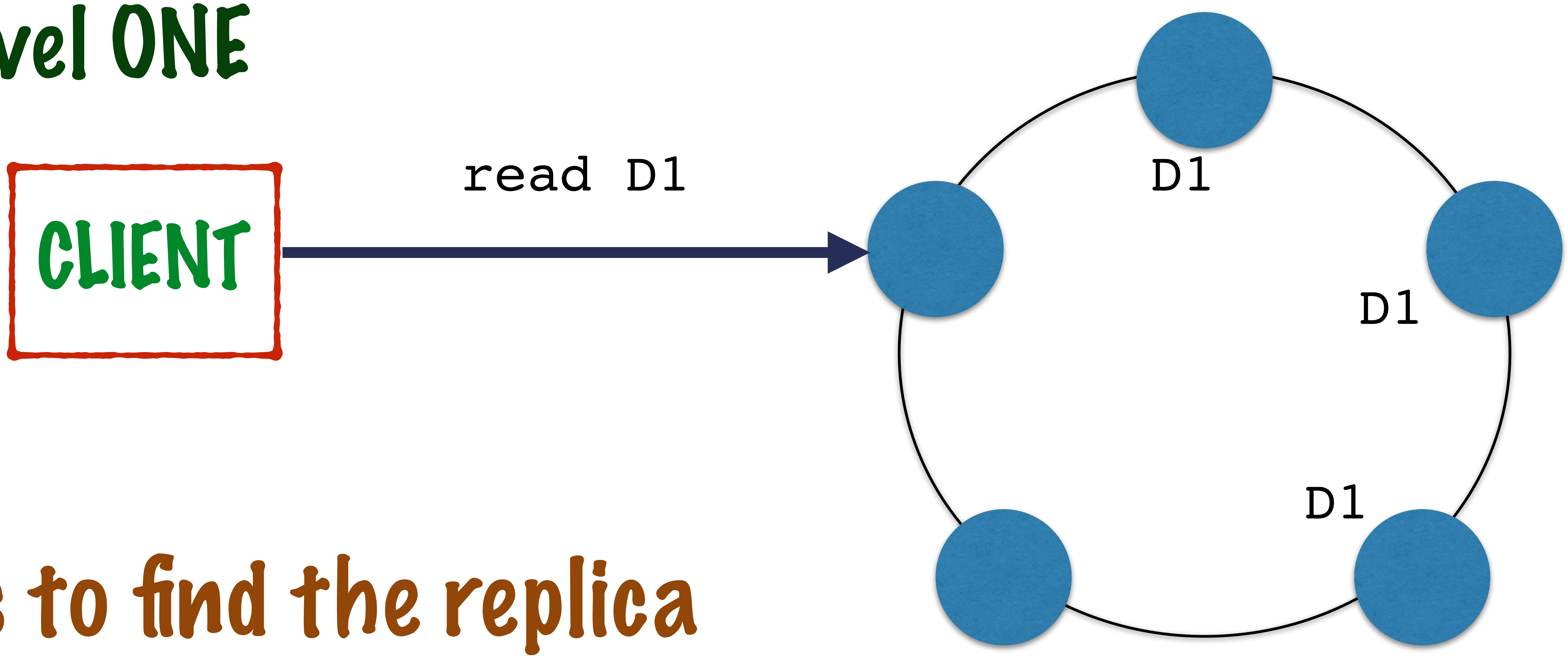


coordinator node determines the replica nodes with the use of token and replicaPlacementAlgorithm

CONSISTENCY

READ

Consistency Level ONE



Now it needs to find the replica node which is the fastest

Snitch is a program which runs on Cassandra and keeps track of a whole bunch of information about the cluster

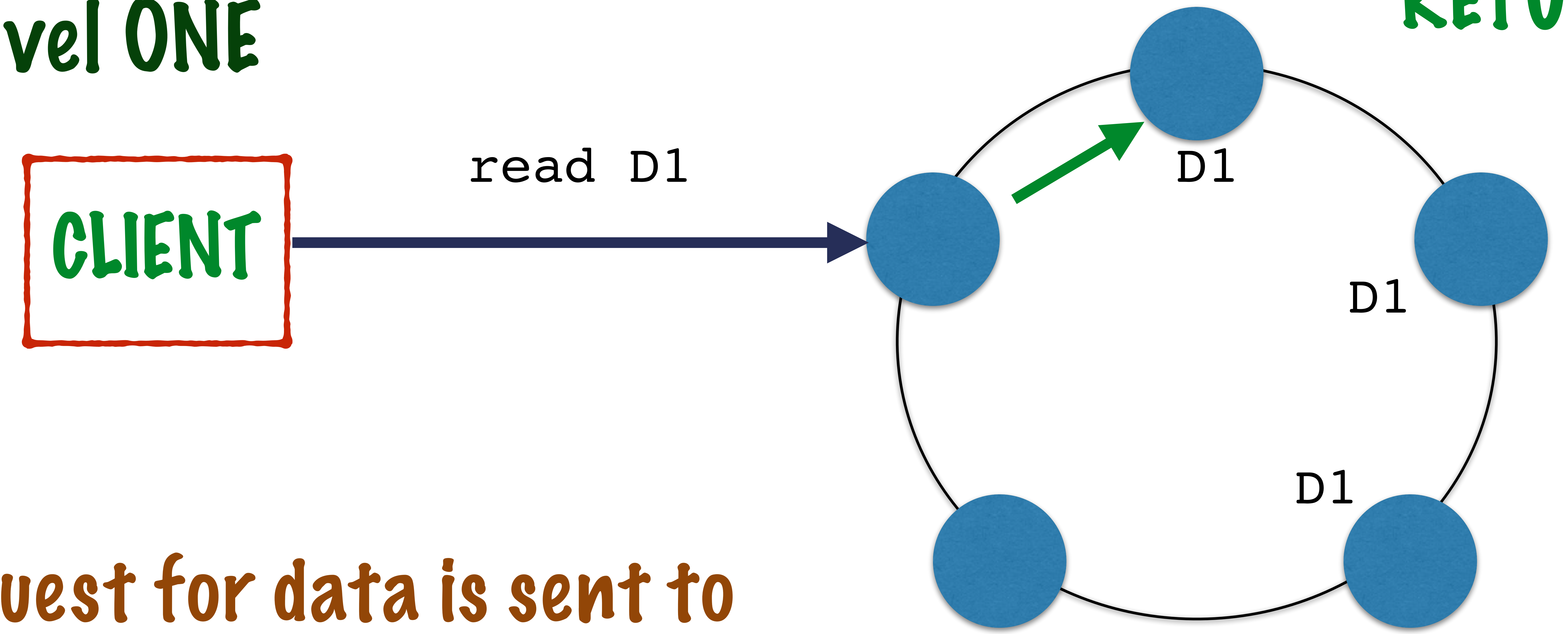
Snitch determines the datacenters and racks each node resides in

It also monitors the network latency between the nodes and maintains this data for each replica

So snitch knows which is the fastest replica

CONSISTENCY
READ

Consistency Level ONE



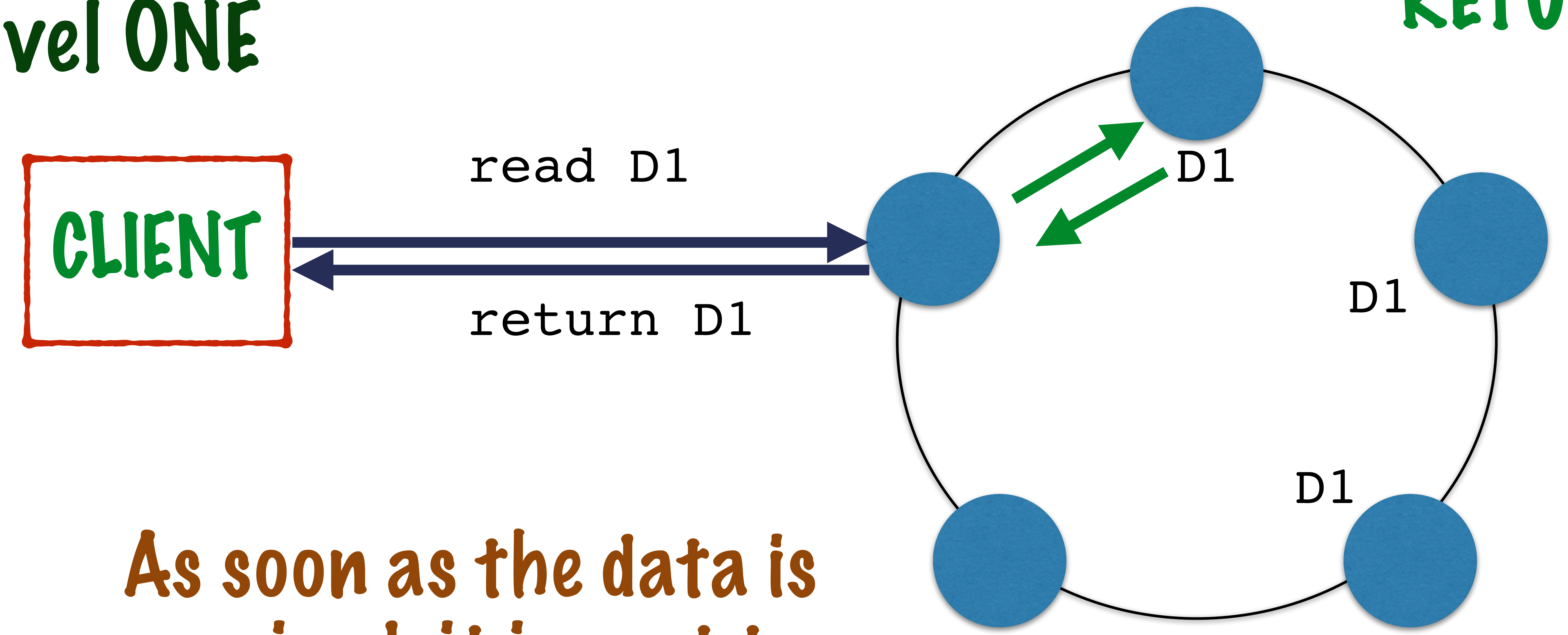
RETURN DATA

request for data is sent to the fastest replica

CONSISTENCY

READ

Consistency Level ONE



RETURN DATA

As soon as the data is received, it is sent to the client

CONSISTENCY

READ

ONE

ALL

QUORUM

LOCAL_QUORUM

CONSISTENCY

READ

QUORUM

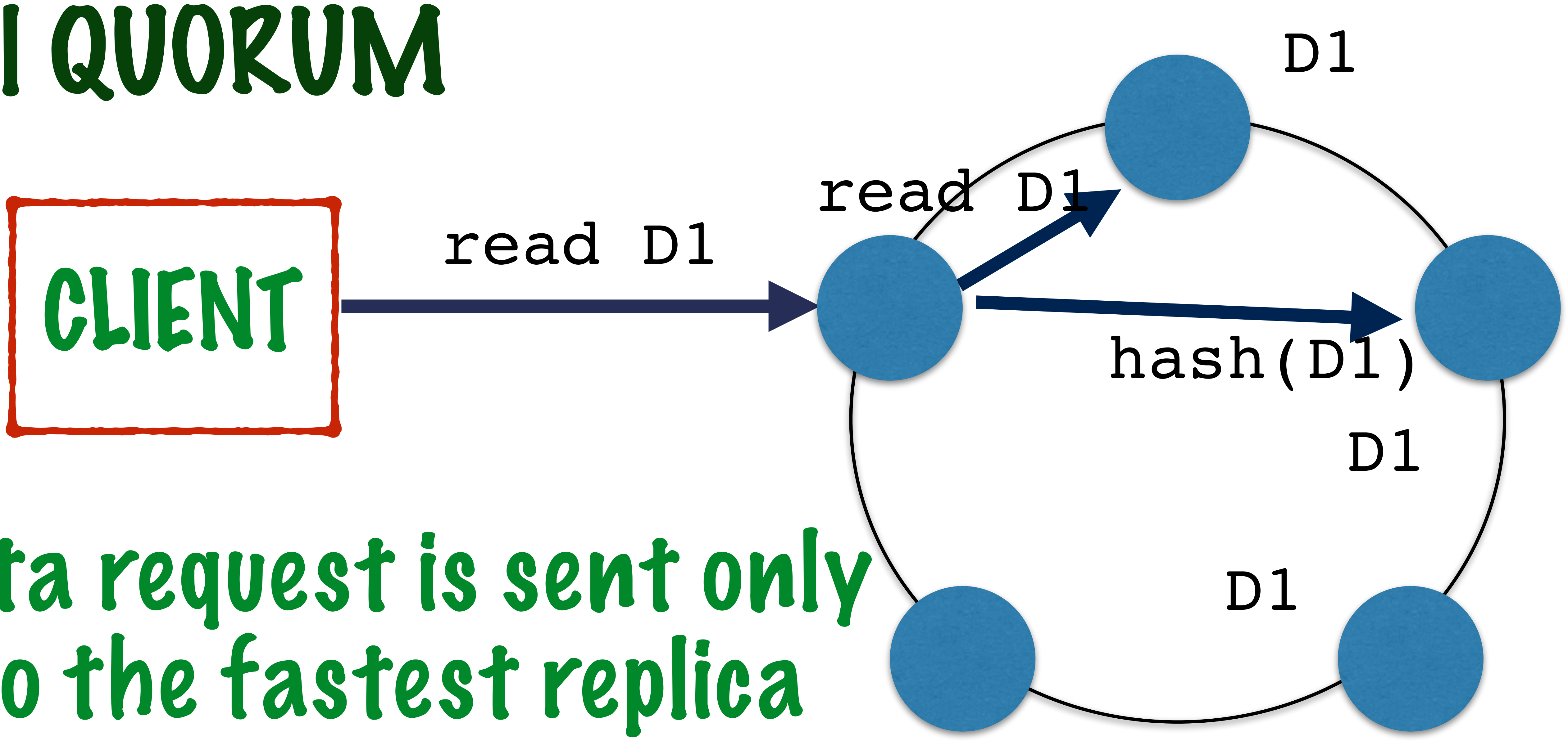
A **minimum** number of replicas (a **quorum**) needs to be read for the read operation to return a success

CONSISTENCY

READ

Consistency Level QUORUM

quorum = 2



data request is sent only to the fastest replica

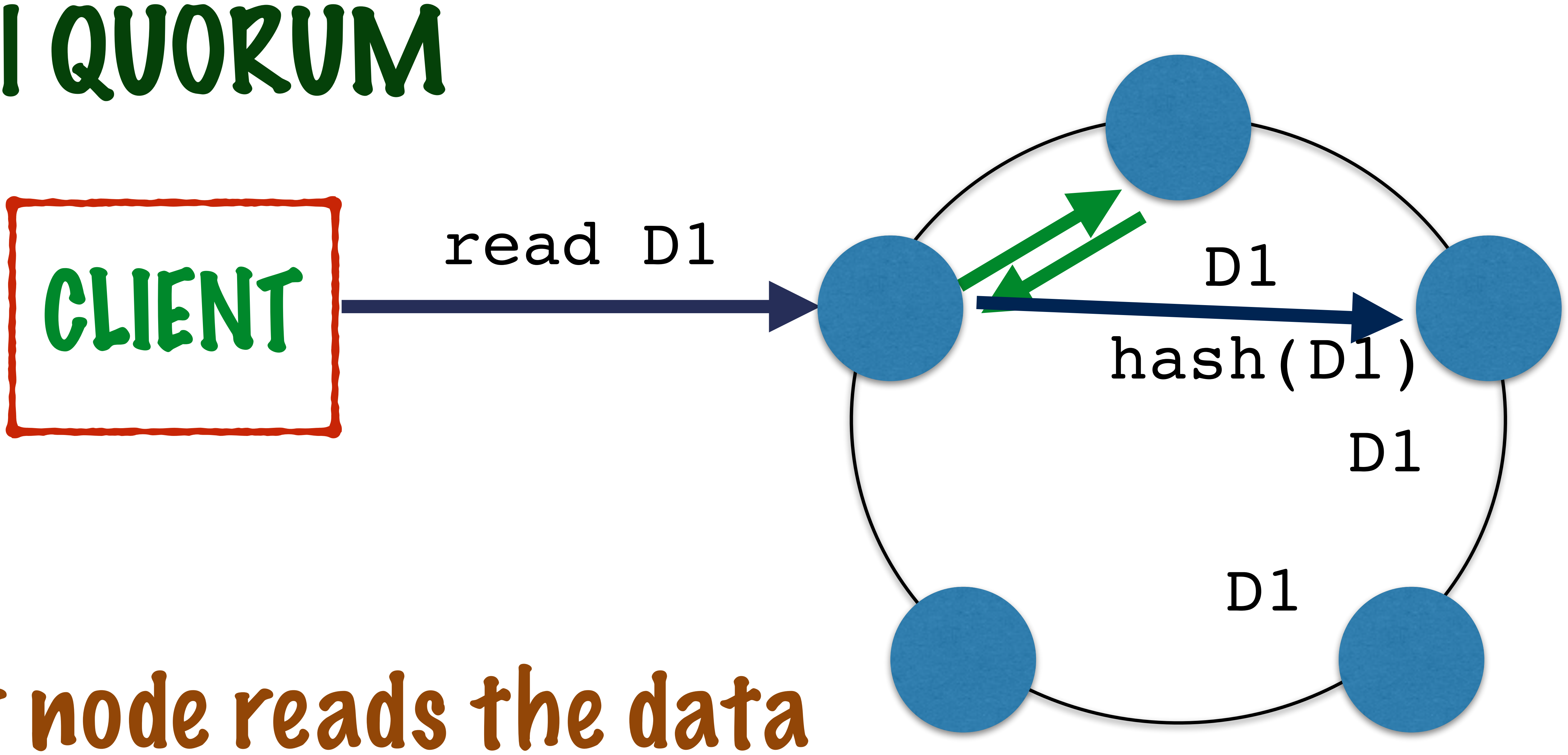
to the other replicas, it sends the request for a hash of the data

CONSISTENCY

READ

Consistency Level QUORUM

quorum = 2

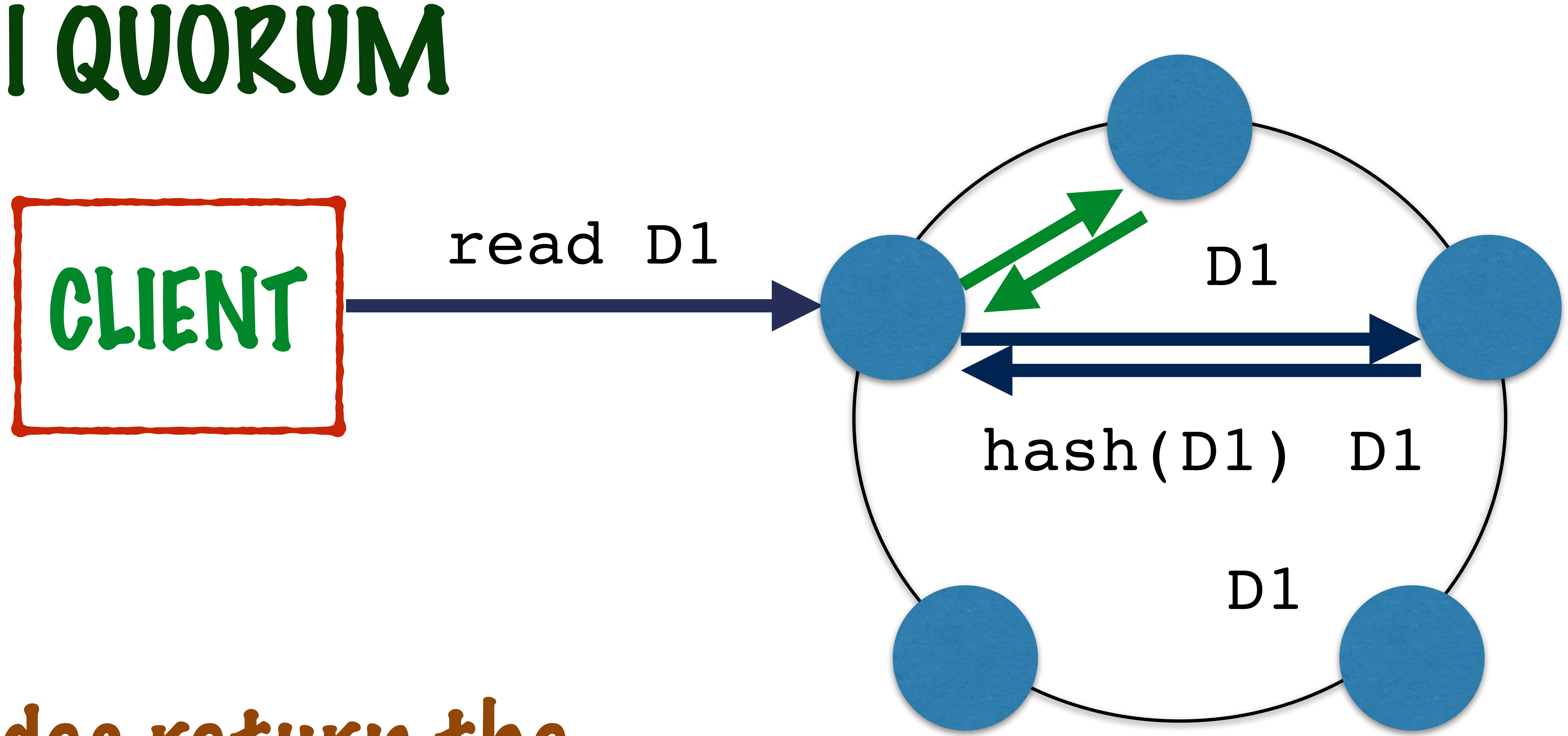


coordinator node reads the data from the first response it receives Lets say it is COPY1

CONSISTENCY

READ

Consistency Level QUORUM



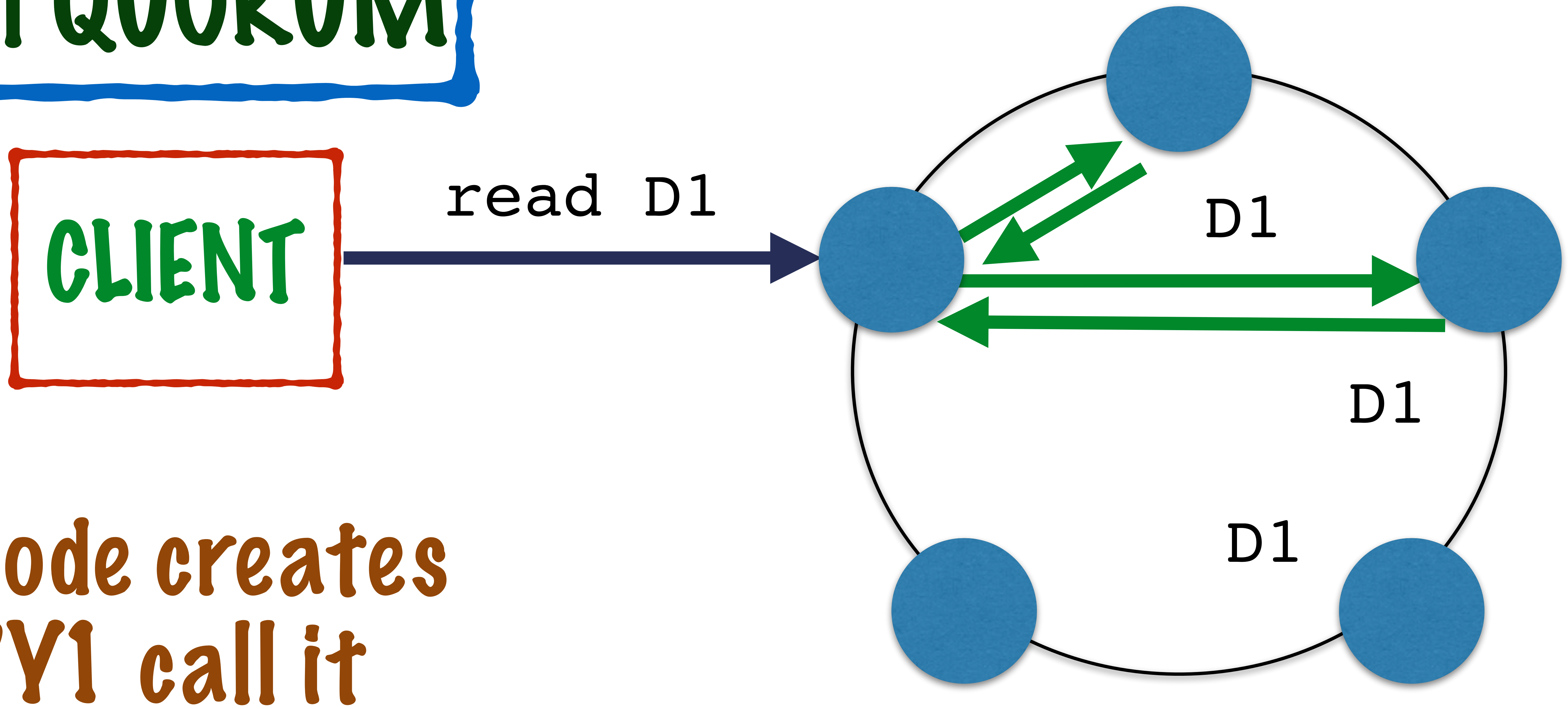
the other nodes return the hash . Lets say it is copy2Hash

CONSISTENCY

READ

quorum = 2

Consistency Level QUORUM



Coordinator node creates hash of COPY1 call it COPY1HASH

It compares the two hashes

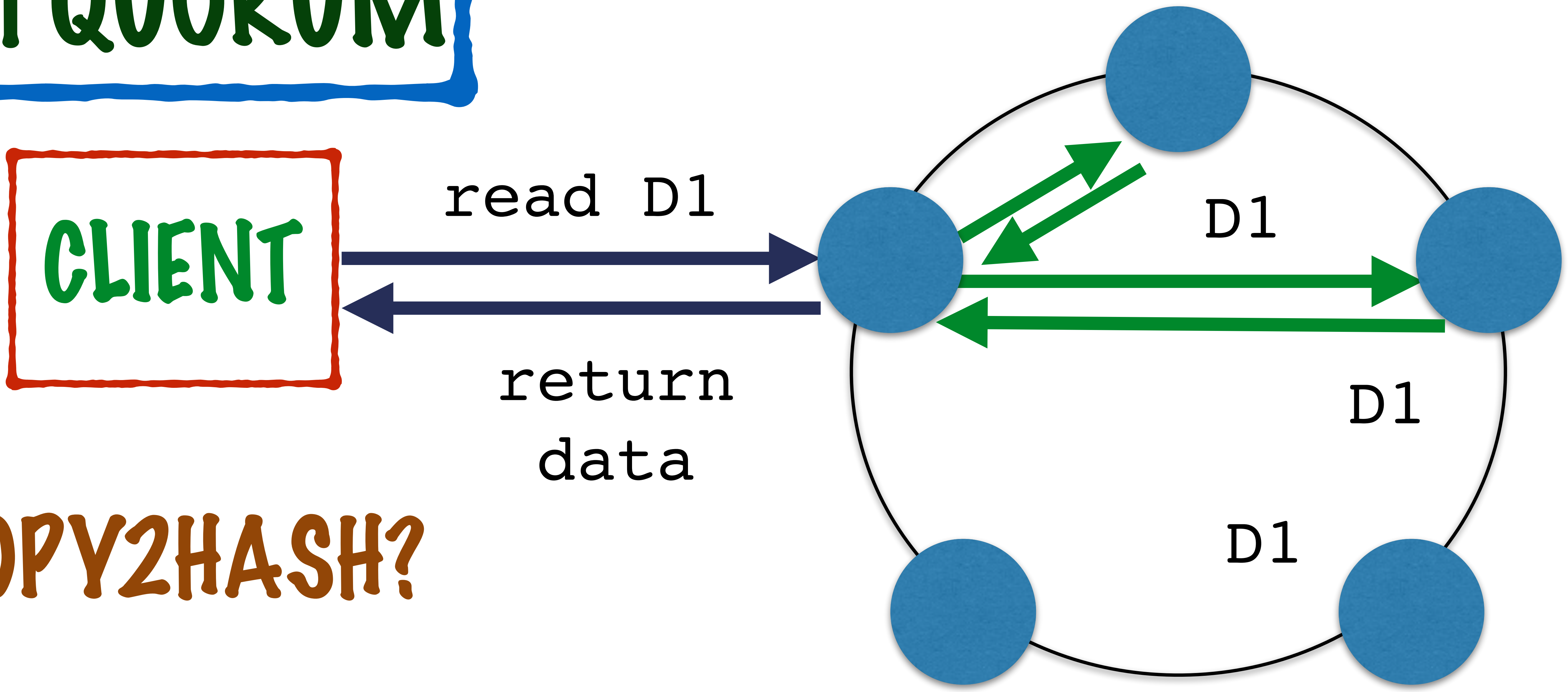
COPY1HASH == COPY2HASH?

CONSISTENCY

READ

quorum = 2

Consistency Level QUORUM



COPY1HASH == COPY2HASH?

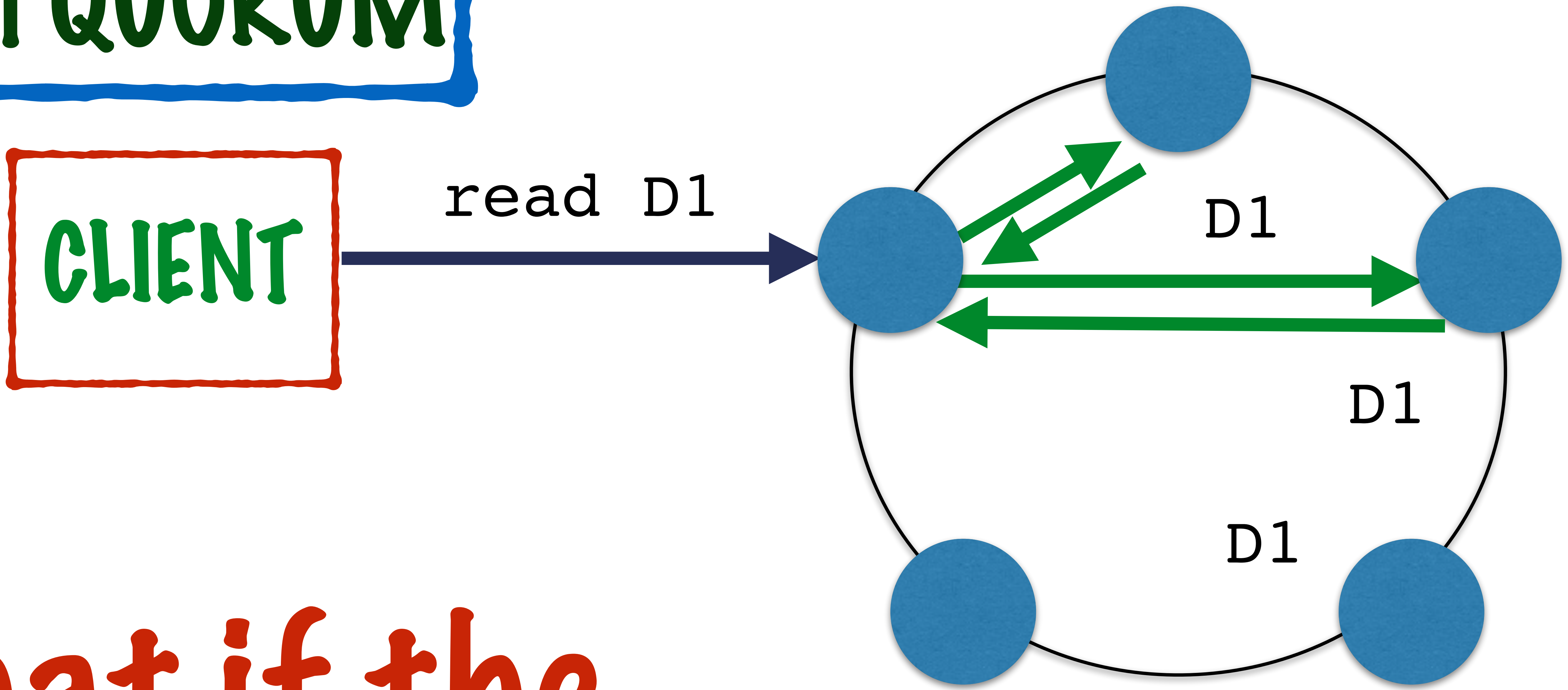
if the hashes match, then return the data to the client

CONSISTENCY

READ

quorum = 2

Consistency Level QUORUM



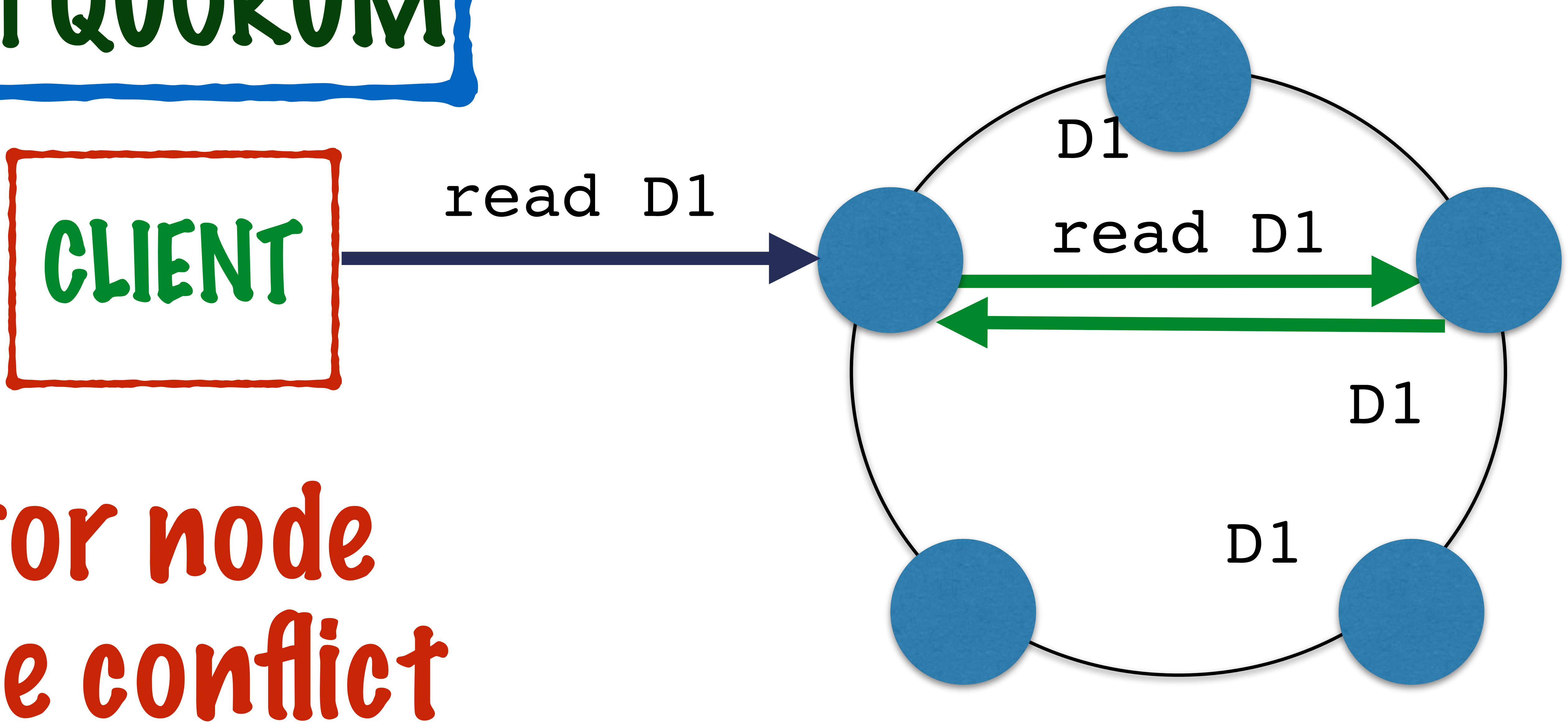
But what if the hashes do not match?

CONSISTENCY

READ

quorum = 2

Consistency Level QUORUM



Coordinator node resolves the conflict

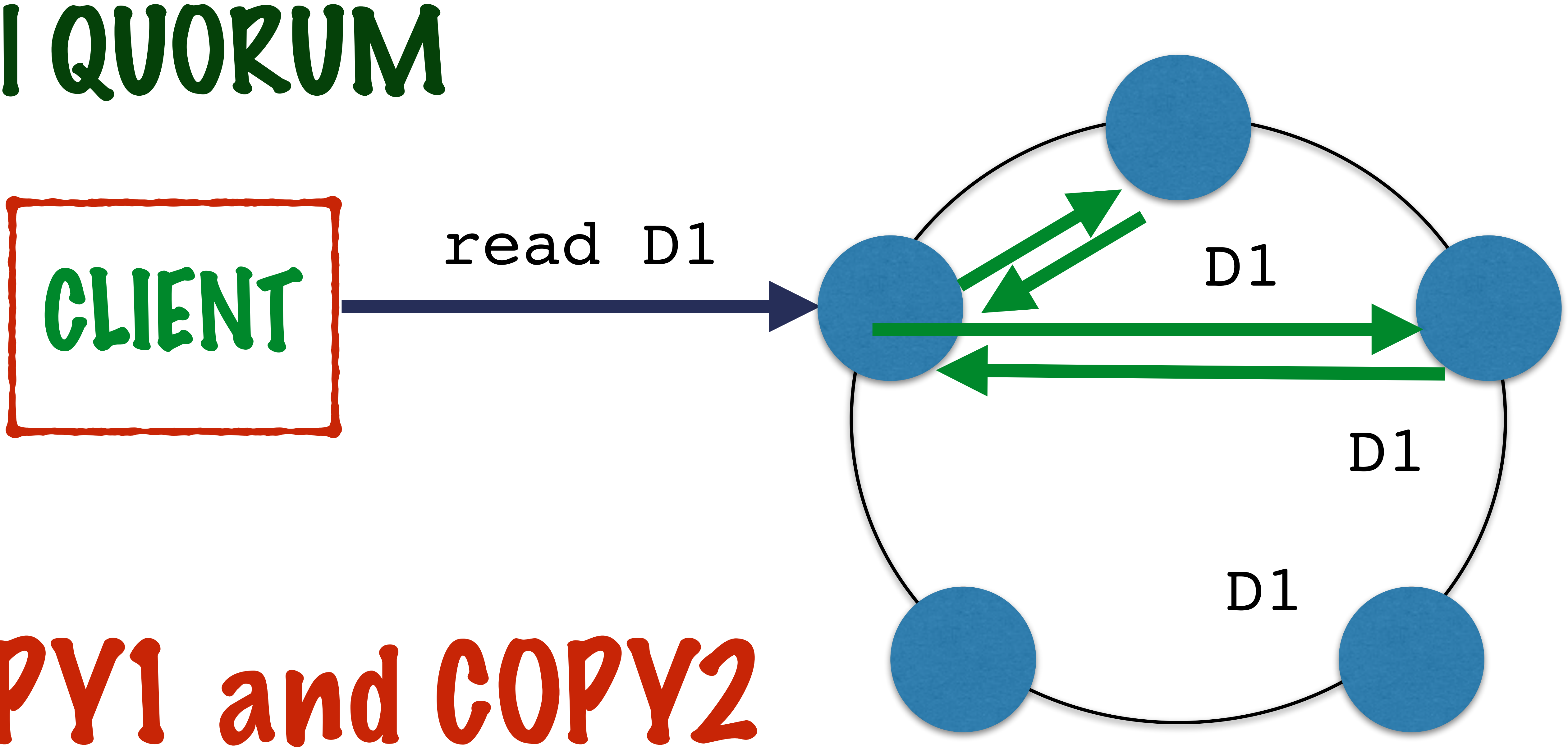
It sends a request to 2nd node to send the data

CONSISTENCY

READ

quorum = 2

Consistency Level QUORUM



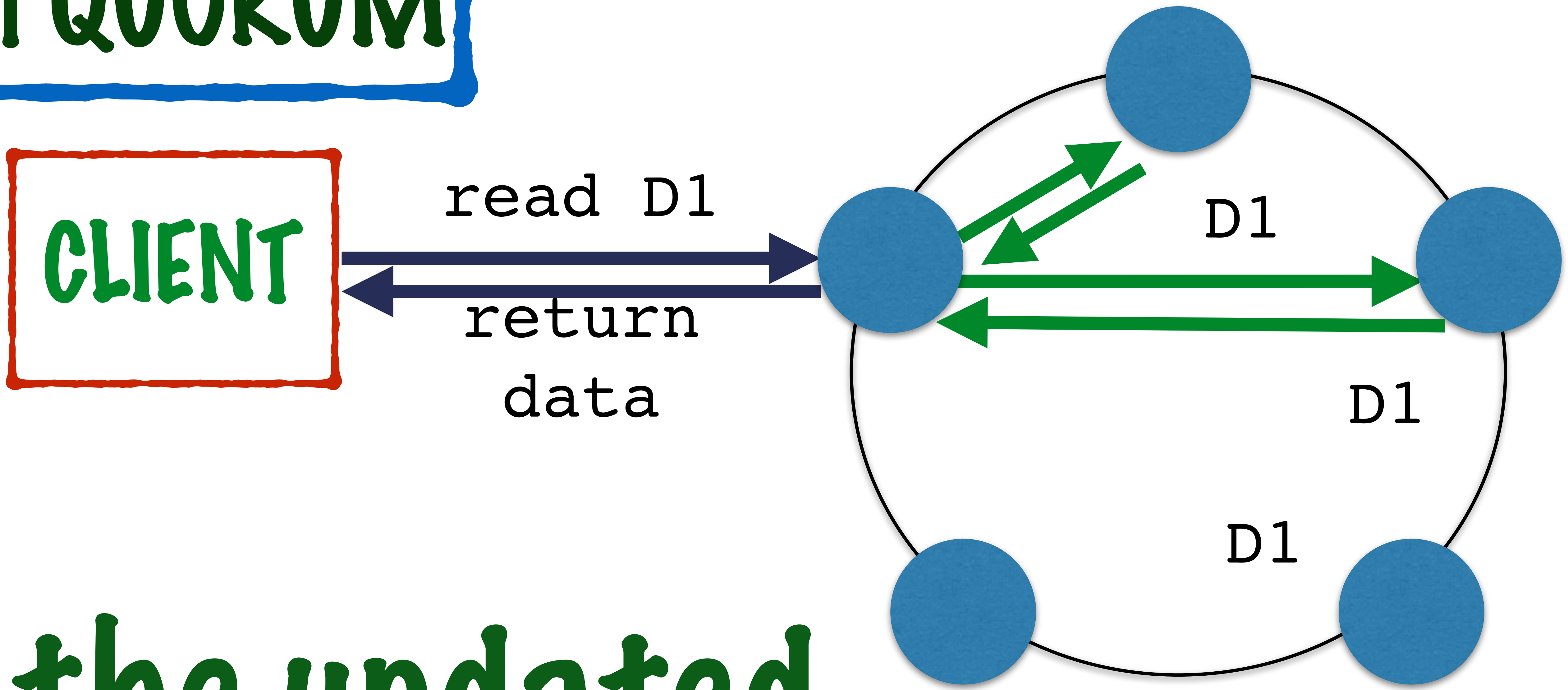
It merges COPY1 and COPY2 by keeping the column data with the latest timestamp

CONSISTENCY

READ

quorum = 2

Consistency Level QUORUM



It returns the updated data to the client

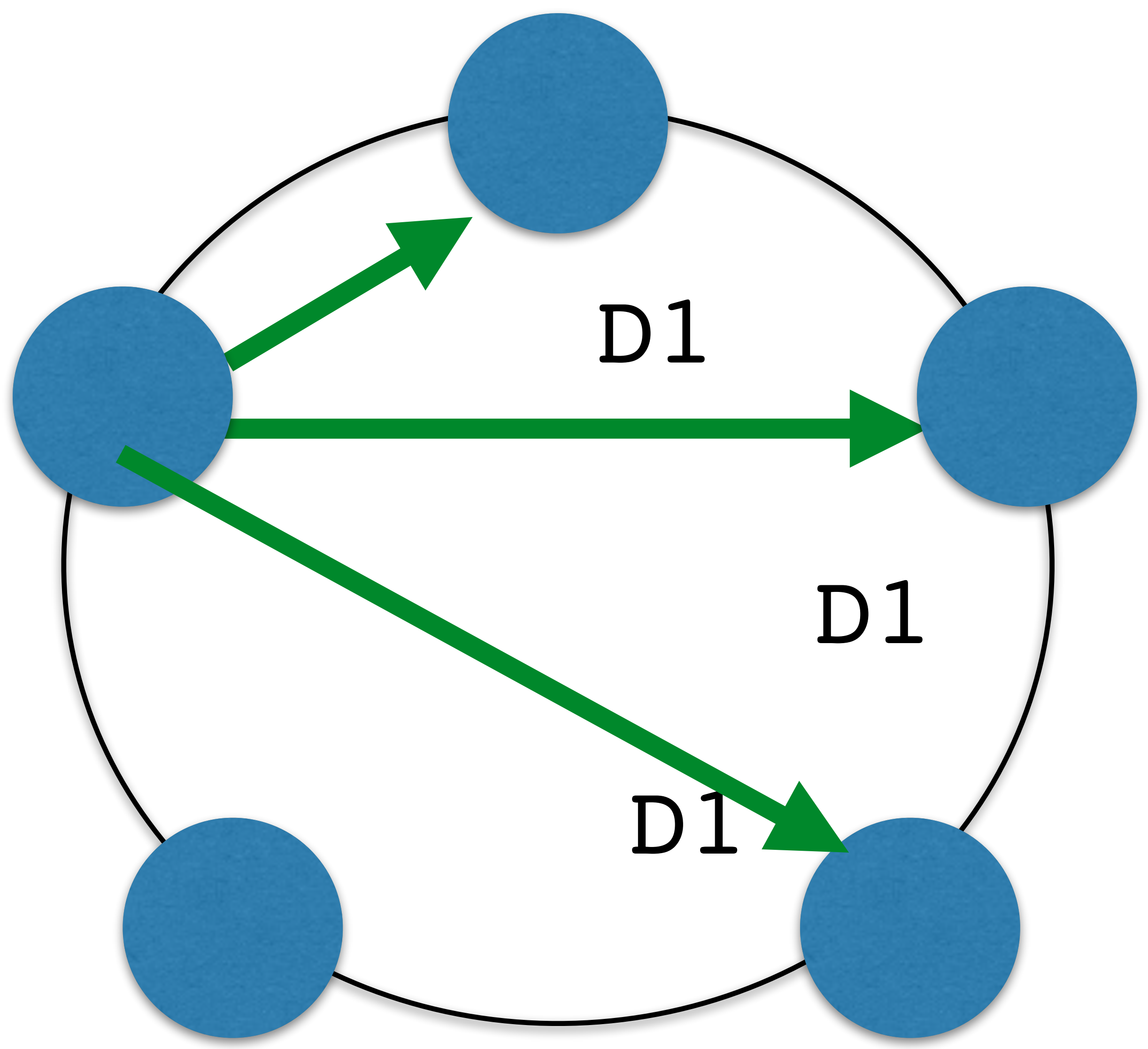
CONSISTENCY

READ

quorum = 2

Consistency Level QUORUM

CLIENT



In the background it sends this updated data to all the replica nodes asking them to update their data

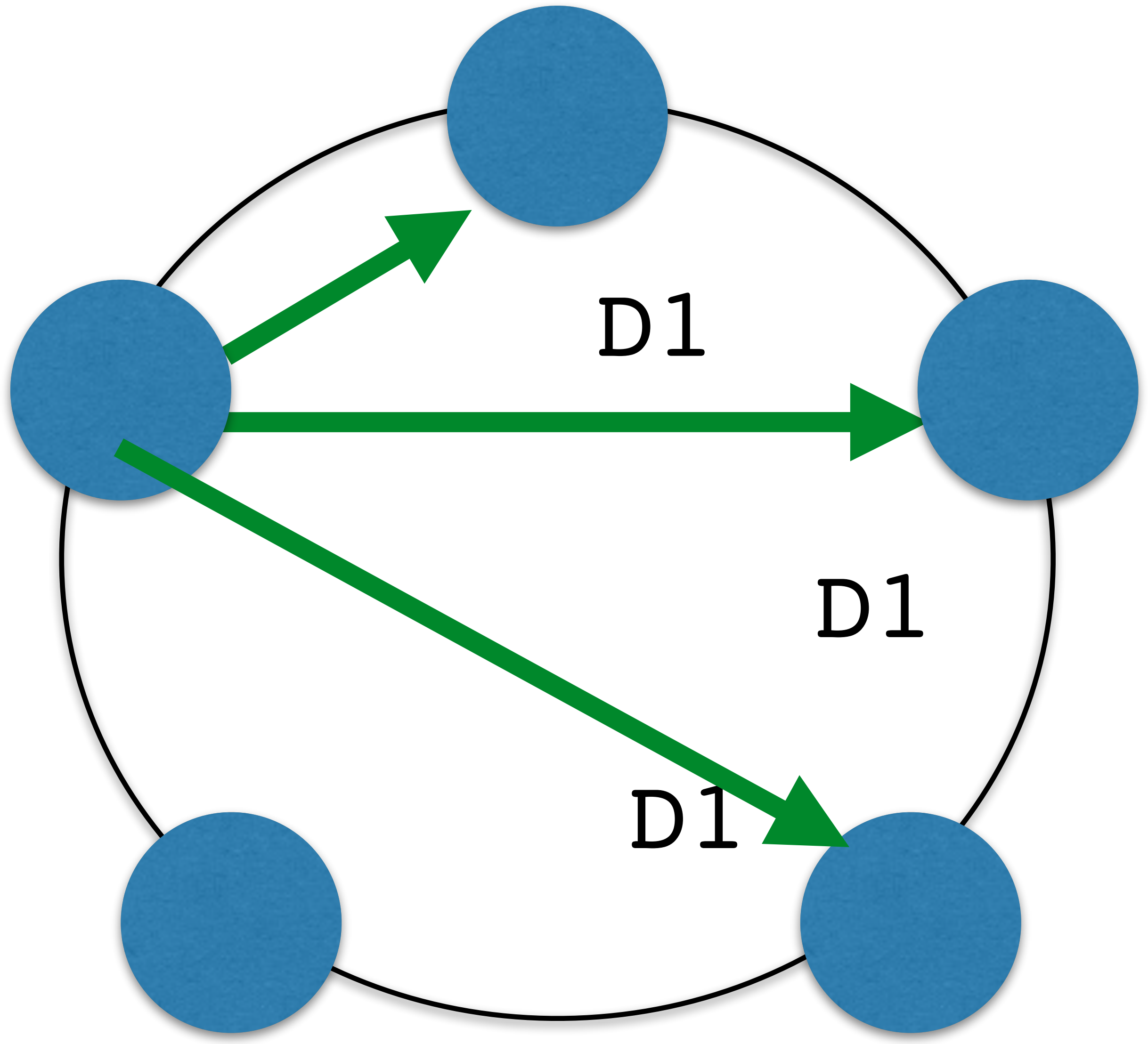
CONSISTENCY

READ

quorum = 2

Consistency Level QUORUM

CLIENT



Now in the background it sends this updated data to all the replica nodes asking them to update their data

This mechanism is called READ REPAIR

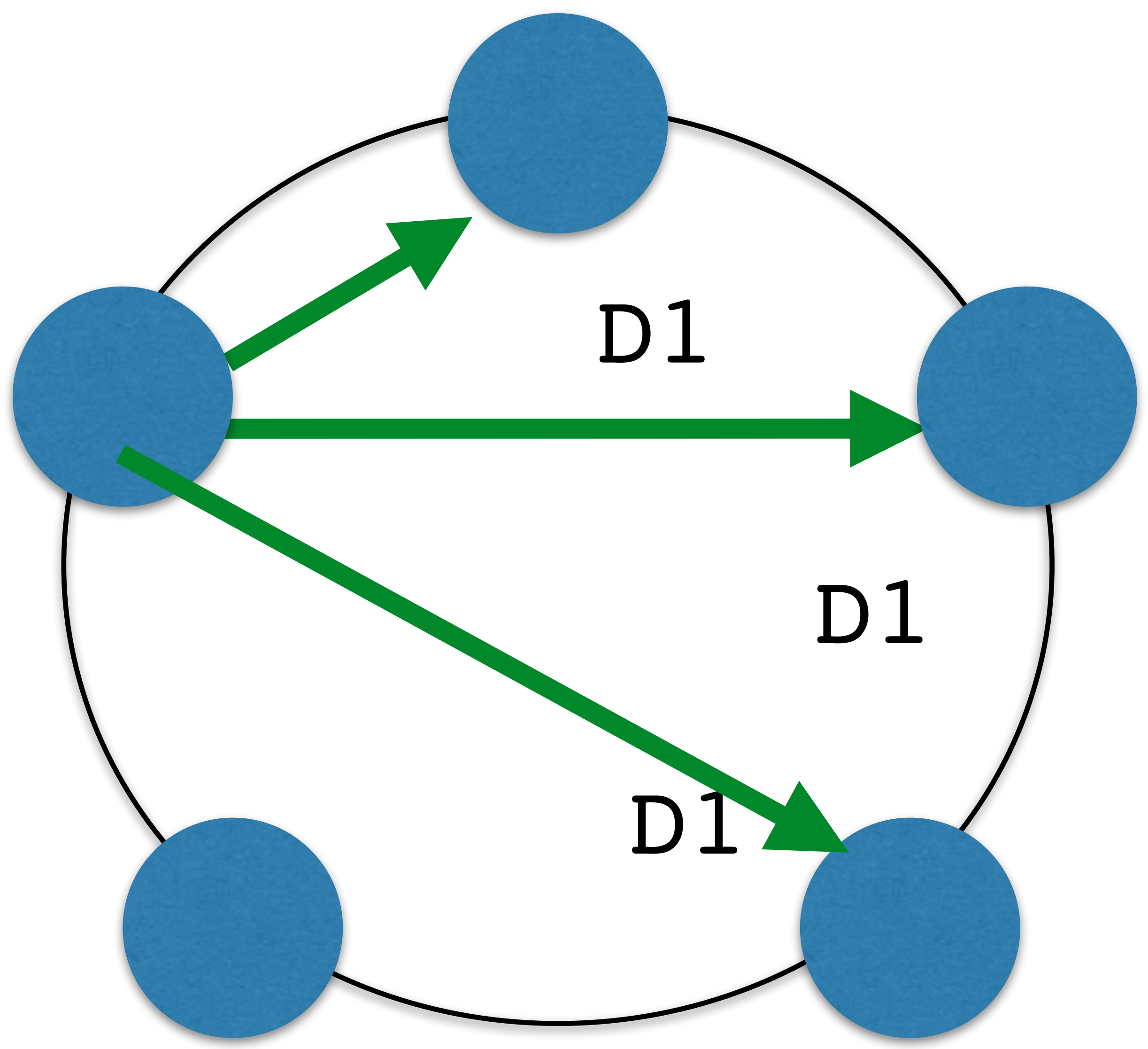
CONSISTENCY

READ

quorum = 2

Consistency Level QUORUM

CLIENT



The frequency at which Read Repair happens is configurable

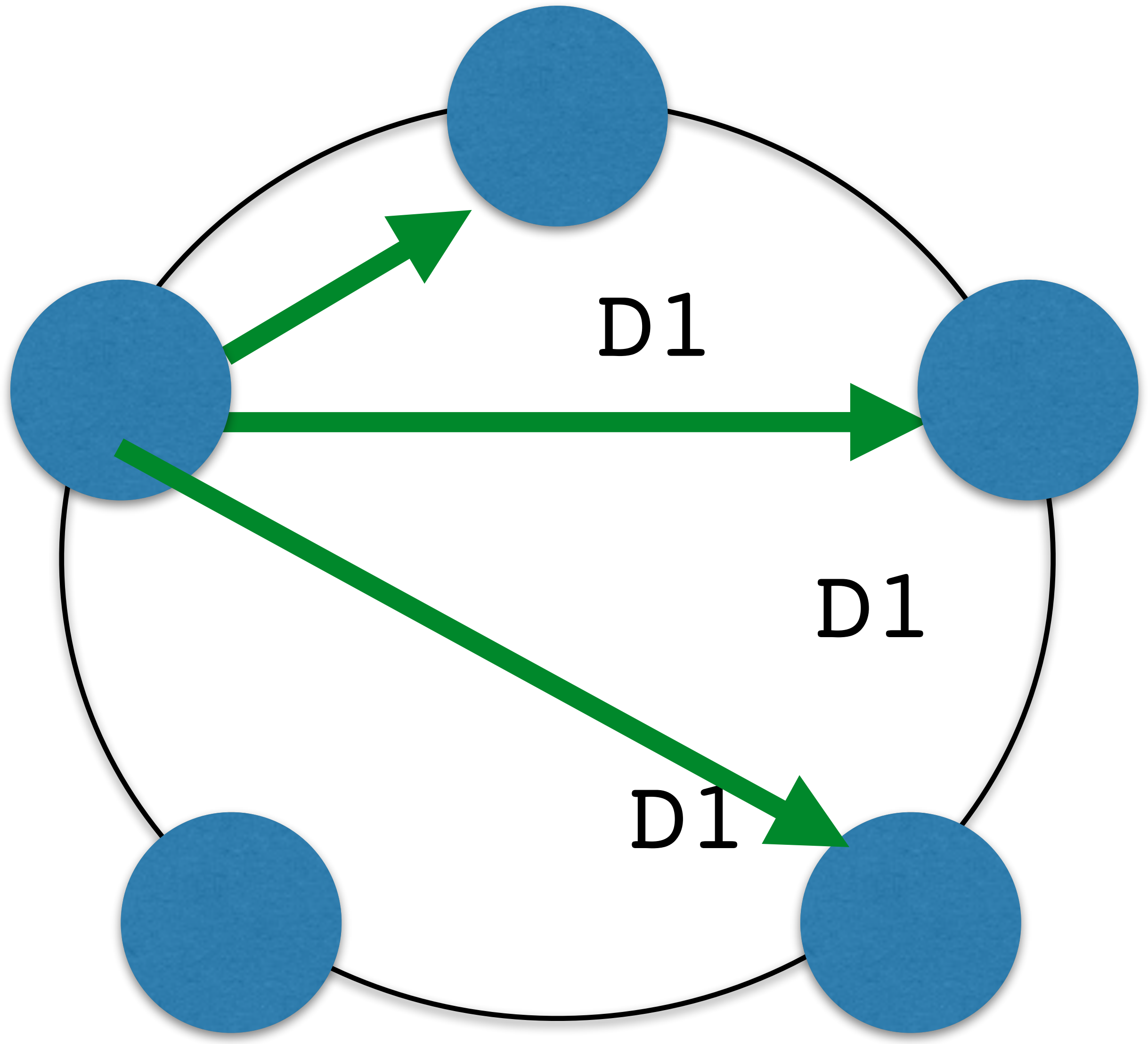
CONSISTENCY

READ

quorum = 2

Consistency Level QUORUM

CLIENT



read_repair_chance value is the probability that read repair will be triggered when there is a mismatch

CONSISTENCY

READ

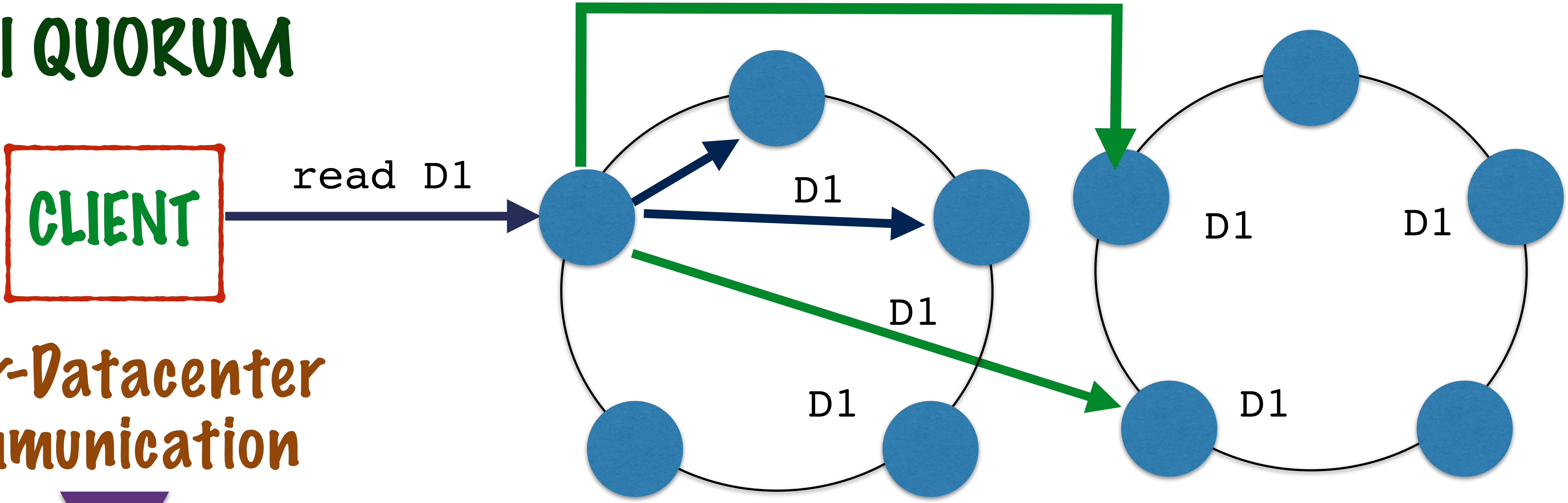
Consistency Level QUORUM

Can we use quorum for a multi datacenter cluster?

CONSISTENCY

READ

Consistency Level QUORUM



Inter-Datacenter communication

▼

HIGHER READ LATENCY

CONSISTENCY

READ

Can we use quorum for a multi datacenter cluster?

NO!

READ OPERATIONS CAN END UP TAKING ARBITRARILY LONG IF THEY ARE CROSS-DATACENTER READS

CONSISTENCY

READ

ONE

ALL

QUORUM

LOCAL_QUORUM

CONSISTENCY

WRITE

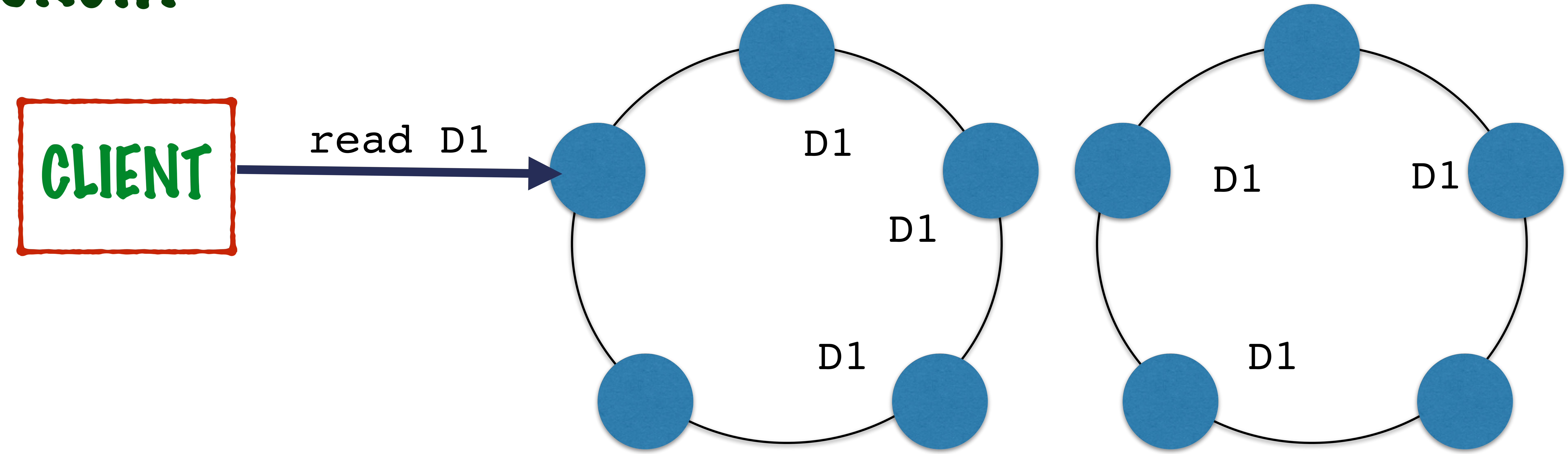
LOCAL_QUORUM

A **minimum** number of replicas (a **quorum**) needs to be read **on one datacenter** for the read operation to return a success

CONSISTENCY

READ

Consistency Level
LOCAL_QUORUM

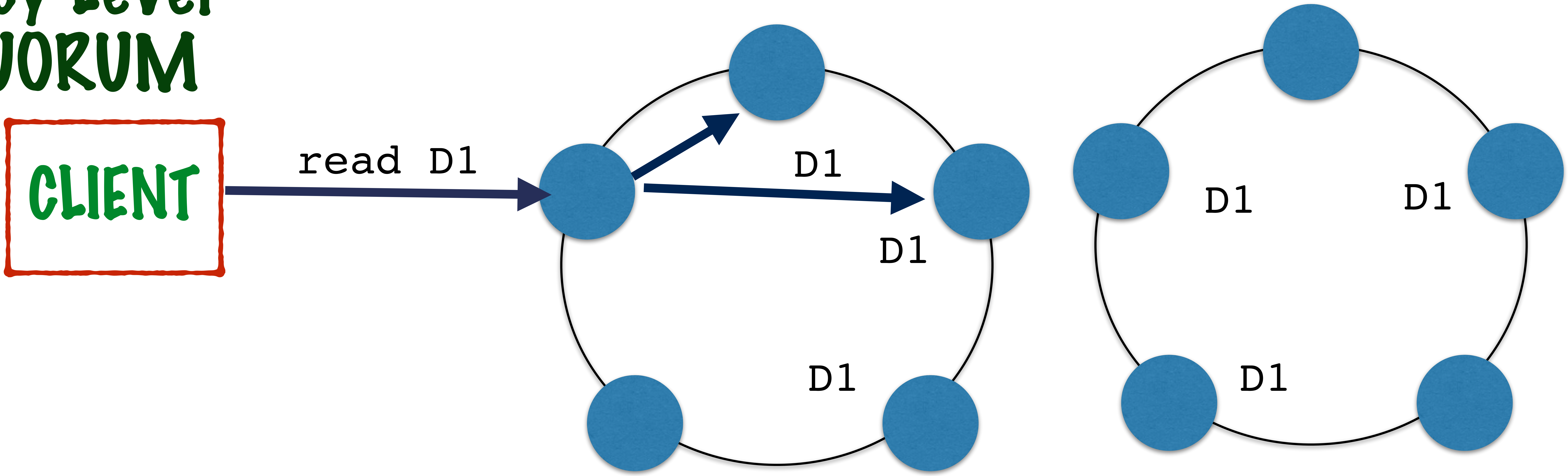


CONSISTENCY

READ

quorum = 2

Consistency Level
LOCAL_QUORUM



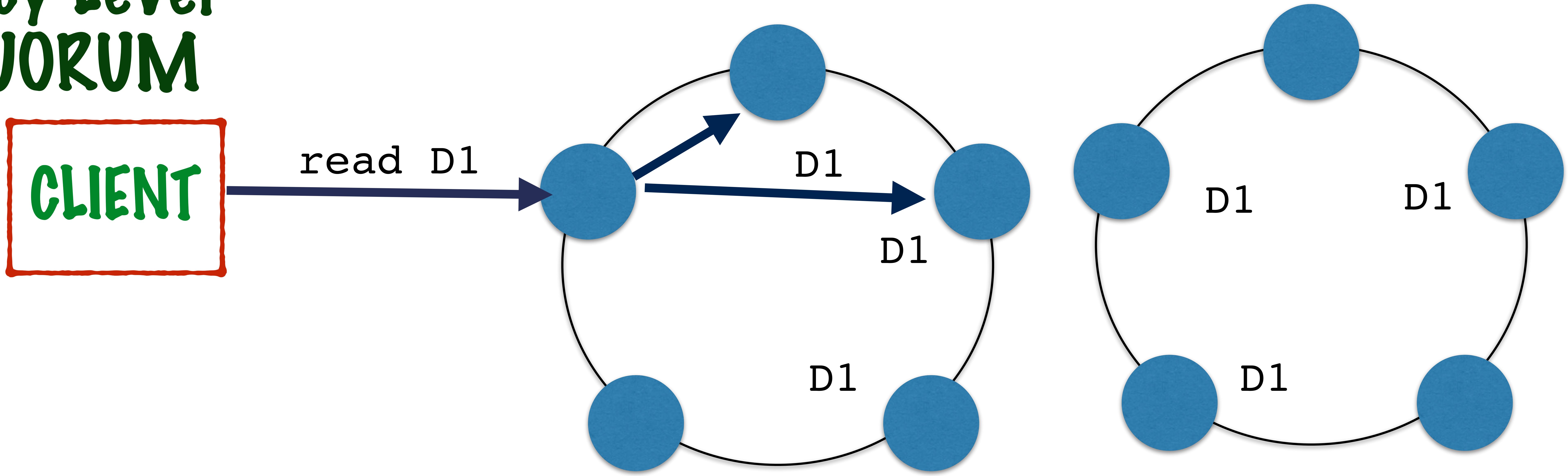
requests data from 2
nodes with D1 replicas
from current datacenter

CONSISTENCY

READ

quorum = 2

Consistency Level
LOCAL_QUORUM



AVOIDS INTER-DATACENTER COMMUNICATION DELAYS

CONSISTENCY

READ

ONE

ALL

QUORUM

LOCAL_QUORUM

CONSISTENCY

WRITE

ALL

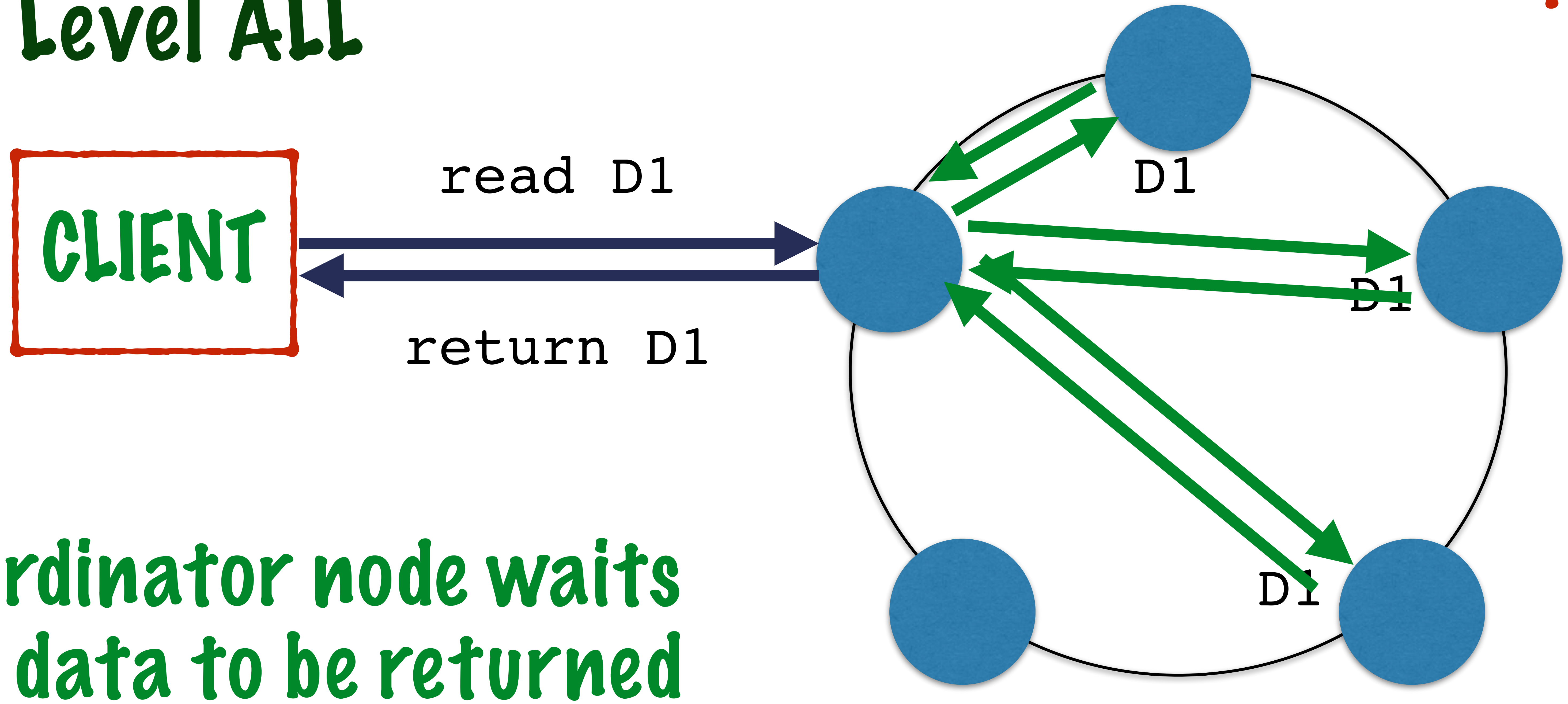
All replicas need to be read and only then does the read operation return success

CONSISTENCY

READ

request fails if even 1 node doesn't send the response

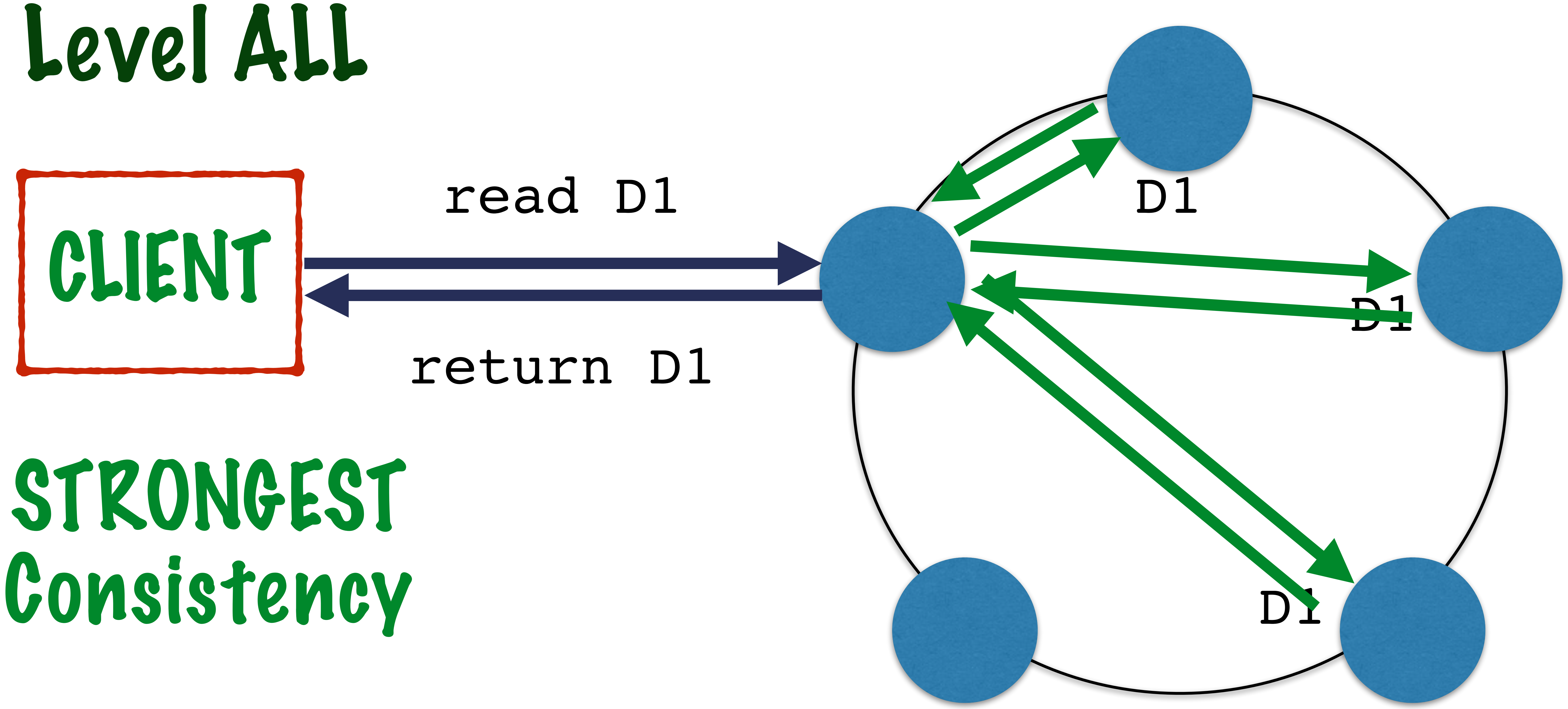
Consistency Level ALL



coordinator node waits for data to be returned from all nodes before returning to client

CONSISTENCY
READ

Consistency Level ALL



STRONGEST Consistency

LOWEST Availability

Quorum value is derived from replication_factor

```
cassandra@cqlsh> CREATE KEYSPACE catalog WITH replication={‘class’:’SimpleStrategy’,
                                                            replication_factor’:’3’};
```

We can set the replication factor of our cluster

How to determine the replication_factor (rf)

rf = 1

Only 1 node has data

Consistency level = one

PROS

faster writes

consistent data

CONS

disk failure → Loss of data

node is down → cluster is unavailable

How to determine the replication_factor (rf)

rf = 1

Only 1 node has data

Consistency level = one

faster writes

consistent data

PROS < CONS

disk fai → Loss of data

node is down → cluster is unavailable

cons outweigh the pros

How to determine the replication_factor (rf)

Lets go to the other extreme

rf = n

consistency level = quorum

PROS

No data loss at all!

CONS

Wastage of disk space

Slower reads and writes

NOT ACCEPTABLE

Relationship between replication factor and quorum

```
rf = replication_factor (configurable)
```

This is per-datacenter

```
quorum = ceiling((sum of all rf + 1)/2)
```

Sum across all datacenters

```
(read_quorum + write_quorum) > rf
```

ensures consistency

Ensuring consistency

```
(read_quorum + write_quorum) > rf
```

write_quorum = rf,
read_quorum = 1

write succeeds only when data is updated on all replicas

read can be from any one replica

Ensuring consistency

```
(read_quorum + write_quorum) > rf
```

write_quorum = 1,
read_quorum = rf

write succeeds when data is updated on 1 replica

During read, all the replica nodes are checked for data

Ensuring consistency

```
(read_quorum + write_quorum) > rf
```

write_quorum = read_quorum = (rf/2)+1

during read operation, there would be atleast 1 replica node where write has been successful

How to determine the replication_factor (rf)

For a 10 node cluster

3 replicas of data ensures fault tolerance

rf = 3
read consistency = quorum (value : 2),
write consistency = quorum (value : 2)

Cluster will be still available, if
1 node has failed

read_quorum + write_quorum = 4 > rf