MongoDB Replication

CS185C: Introduction to NoSQL Databases

Suneuy Kim

References

[1] The Definitive Guide to MongoDB: A Complete Guide to Dealing with Big Data using MongoDB, Third Edition by David Howes, Peter Membrey, Eelco Plugge and Tim Hawkins, December 16, 2015

[2] MongoDB: The Definitive Guide, 2nd Edition, Powerful and Scalable Data Storage by Kristina Chodorow, May 2013

[3] MongoDB Cookbook - 2nd , Chapter 4. Administration, Understanding and analyzing oplogs

[4] https://docs.mongodb.com/manual/replication/

[5] https://www.mongodb.com/presentations/replication-election-and-consensus-algorithm-refinements-for-mongodb-3-2

What is replication in MongoDB

- Replication is a way of keeping identical copies of your data on multiple servers.
- A replica set is a group of servers with one primary and multiple secondaries.

Primary

- Source of truth at a given moment for the replica set.
- Only server in a replica set to which data can be directly written by your application.

Secondary

- A data-carrying non primary member that replicates its data from another member of its set in as fast as possible.
- To read from a secondary, the connection should be explicit about it by rs.slaveOk(). If you are using driver, set a read preference in the application.
  - This enforcement is due to that fact that any read to non-primary may read old data.
- A secondary replicates from the primary or from one to another using replication chaining.

oplog (operation log)

- A capped collection storing a log of the changes that a primary makes to its databases. (window on the recent activity of your primary)
- Each secondary own copies of all databases. Replaying `oplog` to a secondary should ensure that the databases in each member of the set are identical.
- Each secondary maintains its own oplog and queries the primary or other more up-to-date secondary's via replication chaining for new entries to apply to their own copies of all databases.

Replication using oplog

- Primary writes to an oplog all the contents that need to be replicated. Thus, any create, update, and delete operations as well as any reconfigurations on the replica sets would be written to the oplog
- Secondary would tail (continuously read the contents of the oplog being added to it, similar to a tail with -f option command in Unix) the collection to get documents written by the primary.
- If the secondary has a slaveDelay configured, it will not read documents more than the maximum time minus the slaveDelay time from the oplog.

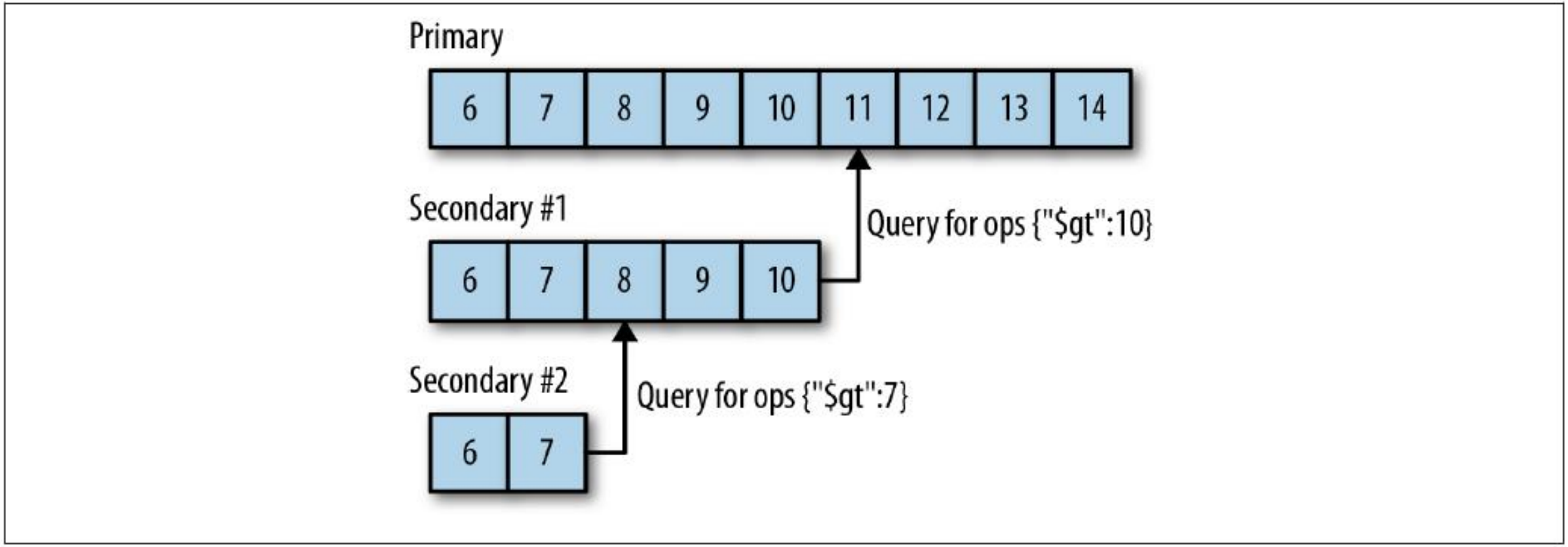


Figure 10-1. Oplog keep an ordered list of write operations that have occurred. Each member has its own copy of the oplog, which should be identical to the primary's (modulo some lag).

oplog size

- The oplog will use space at approximately the same rate as writes come into the system.
- Example

With writing 1KB/minute on the primary, your oplog is probably going fill up at 1KB/minute

- Exception (cases that fill up oplog more quickly than expected)
  - Operations that effect multiple documents
  - Bulk operations fill up oplog more quickly

Handling Staleness

- If a secondary falls too far behind the actual operations being performed on the sync source, the secondary will go stale.
- A stale secondary is unable to continue catch up because every operation in the sync source‘s oplog is too far ahead: it would be skipping operations if it continued to sync.
- Happens the slave has down time or has more writes to the oplog than it can handle
- A stale secondary will attempt to replicate from each member of the set in turn to see if there is along-enough oplog to catch up.
- If it fails, it should be fully resynced from a most recent backup.

Write and Read with Replica Sets

Write:

1. MongoDB applies write operations on the primary
2. MongoDB records the operations to the primary's oplog
3. Secondary members replicate oplog + apply the operations to their data sets

Read: All members of the replica set can accept read operations

- By default, an application directs its read operations to the primary member
  - Guaranties the latest version of a document
  - Decreases read throughput
- Read preference mode can be set

Member configuration options

- Arbiters
- Priority
- Hidden
- Slave Delay

Arbiter

- A non-data-bearing node that may be used to throw an additional vote to break ties in an election. (Only purpose)
- You do not need an arbiter if you have an odd number of nodes.
  - A replica set with 3 nodes: quorum is 2 (67%)
  - A replica set with 3 nodes + 1 arbiter: quorum is 3 (75%)
- If you have a choice between a data node and an arbiter, choose a data node
  - Three data nodes – if one secondary dies, use the remaining secondary node to bootstrap a new server instead of depending on the primary.
  - Two data nodes + one arbiter - if one secondary dies, the primary is the last remaining good copy of data.

Priority

- How strongly this member wants to become primary.
- Range between 0 to 100 and the default is 1.
- Priority 0 – **passive** member who can never be primary
- The highest-priority will always be elected primary so long as they can reach a majority of the set and have the most up-to-date data.
- If a server with the highest priority is not up to dated, it becomes primary once it caught up with the rest of the set. (In this case, the current primary automatically steps down.)

Hidden

- Clients do not route request to hidden members.
- Hidden members are not preferred as replication sources.
- Hide less powerful or back up servers.
- The most common use of hidden nodes is to support delayed members.
- Hidden members must always be priority 0 members and so cannot become primary.
- Hidden members, however, may vote in elections.

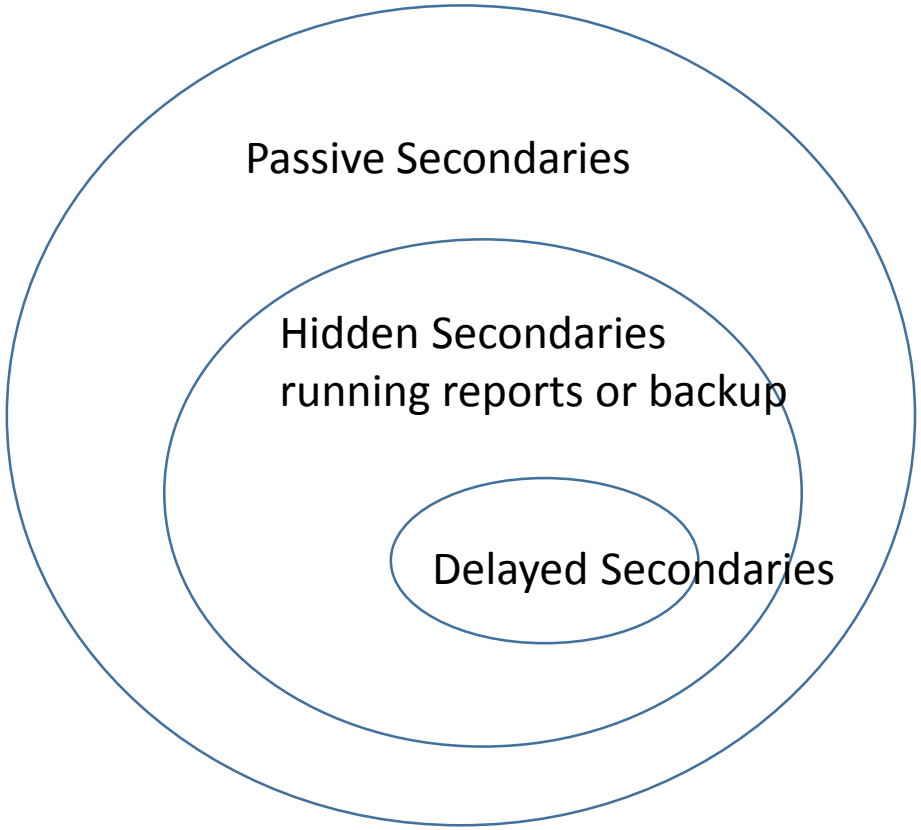
Delayed Secondary

- A delayed secondary delays applying operations a specified number of seconds behind the primary – a running historical snapshot of the replica

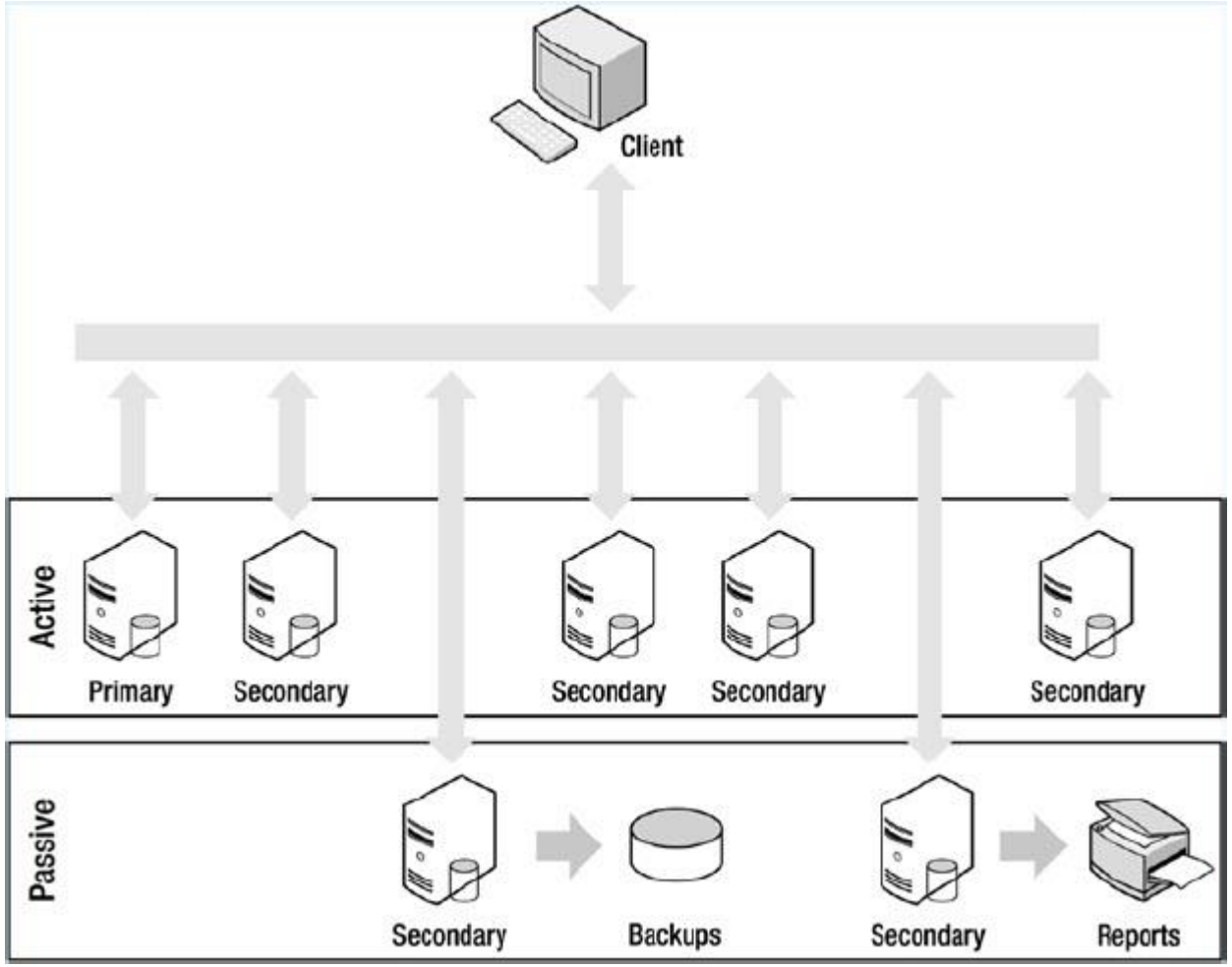
  For example, if the current time is 09:52 and a member has a delay of an hour, the delayed member has no operation more recent than 08:52.

- When a database is corrupted by human error (e.g. accidently dropping main database), you can restore it from identical copy of the data from earlier.
- The amount of delay must be smaller than the capacity of the oplog.

Members with priority = 0



A cluster implemented with a replica set



Replica Set Elections

- Replica set can have at most one primary
- If the current primary becomes unavailable, an election determines a new primary
- Fail over time
  - Time window where a replica set cannot take a write request due to not having a primary
  - A function of time to take a successful election (which might require multiple elections)

Replica Set Elections – Influencing factor

- Heartbeat (ping)
  - Members need to know about other members' states: who's primary, who they can sync from , and who is down.
  - The most important function of hear beats is to let the primary know if it can reach a majority of the set.
  - Every 2s sent to each other
  - No response for 10s node is inaccessible
- Priority: A healthy member with the highest priority is going to be a new primary.
- Connections: A node cannot become primary unless it can connect to a majority of the members

Replica Set Elections

- Replica sets hold an election any time there is no primary:
  - Initiation of a new replica set
  - A secondary loses contact with a primary
  - A primary steps down
- A primary will step down:
  - After receiving the replSetStepDown command which forces a primary to become a secondary
  - If one of the current secondaries is eligible for election and has a higher priority
  - If it cannot contact a majority of the members of the replica set

How Elections Work

- When there is no primary, any eligible member can seek an election.
- The member seeking an election will send out a notice to all of the members it can reach.
- If ever one server vetoes the election, the election is canceled.
  - The member seeking an election is not up to date. It must do synching and call for an election again. (If no one else has become primary in that time.)
  - If the member seeking an election has a lower priority than other member which is eligible for election.
- Members seek election for themselves and neighbors cannot nominate another server.

Replica Set Election Criteria

- Priority of every secondary MongoDB server
- Oplog (new primary is up to dated.)
- Up Time (how long the secondary has been up)
- Network Proximity (secondary in San Francisco and New York, which is closer)

→ Healthy node with highest priority will eventually become the primary

https://www.youtube.com/watch?v=WxV3khkca4I#t=2910.003948

Rollback

- If a primary does a successful write and goes down before the secondaries have a chance to replicate it, the next primary elected may not have the write.
- When it resurrects, it cannot find any sync source. Then, it has to roll back by undoing ops that were not replicated before failover.

Rollback Example

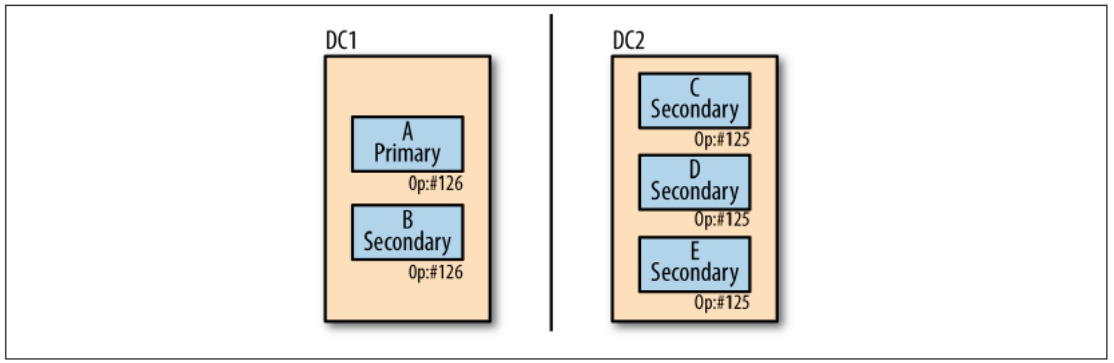


Figure 10-3. Replication across data centers can be slower than within a single data center

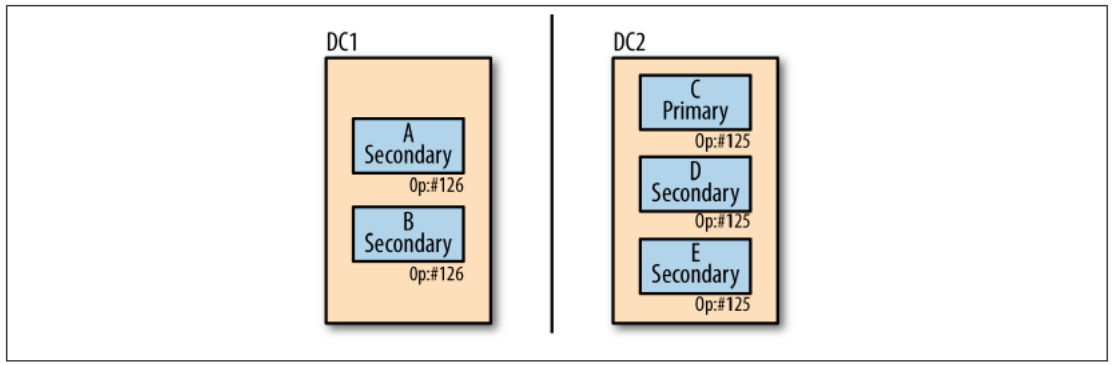


Figure 10-4. Unreplicated writes won't match writes on the other side of a network partition

- In the above example, when the network is repaired, A and B will not be able to find any sync source. And thus, A and B will begin rollback, undoing ops that were not replicated before fail over.
- A and B will copy each document affected by 126 in a rollback directory and will copy the version of that document from the current primary.
- Once rollback is complete, it transitions into recovering state and begins syncing normally again.

More thoughts on Rollback

- In this previous example, is the operation 126 going to be lost?
  - They are written to special rollback files that have to be applied to the current primary.
  - Thus, the write disappears until an admin manually applies the rollback files to the current primary. (MongoDB cannot automatically do it since they may conflict with other writes that have happened since the crash.)
- Writing to a majority prevents rollback – the new primary would have to have a copy of the write to be elected (a member must be up to date to be elected primary.)

Replica set is about majority

- You need a majority of voting members to elect a primary
- A primary can only stay primary so long as It can reach a majority
- A write is safe when it's been replicated to a majority
- What is a majority?
- More than half of all members in the set.
- Majority is based on the set's configuration without considering how many members are down or unavailable.

Table 9-1. What is a majority?

Number of members in the set	Majority of the set
1	1
2	2
3	2
4	3
5	3
6	4
7	4

Why a majority is enforced to elect or stay primary?

- Requiring a majority to stay primary is to avoid ending up with more than one primary in case of network partition.

Why a majority is enforced to elect or stay primary?

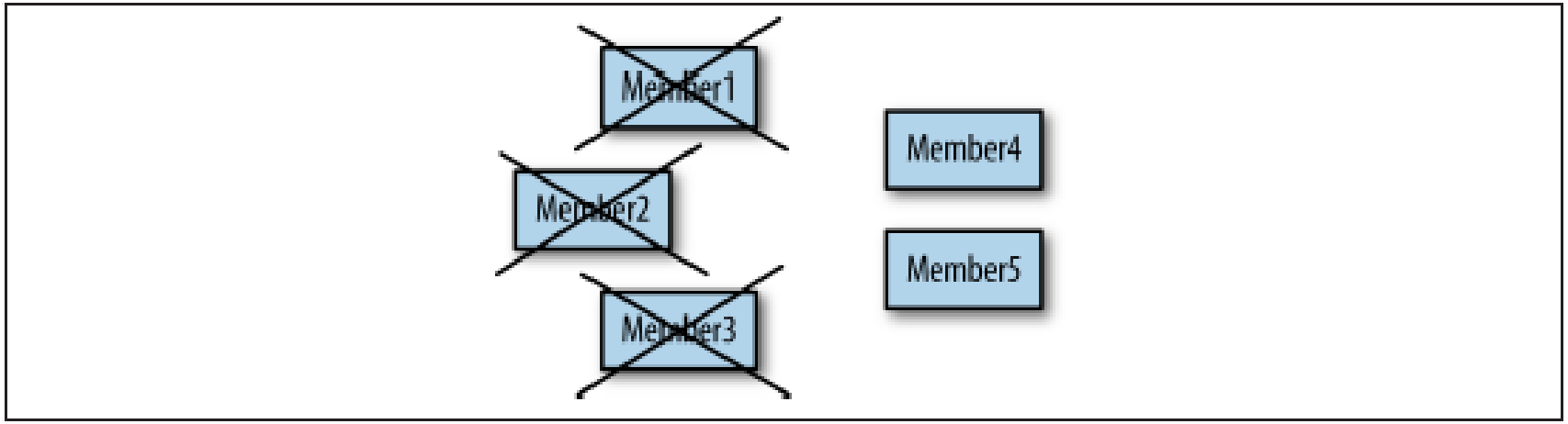


Figure 9-1. With a minority of the set available, all members will be secondaries

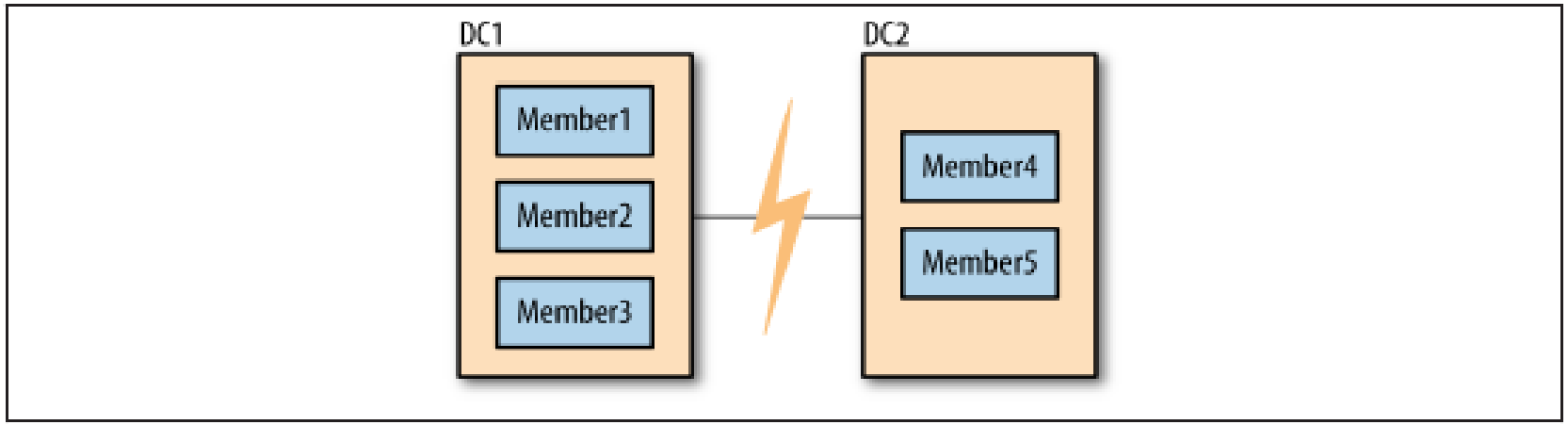


Figure 9-2. For the members, a network partition looks identical to servers on the other side of the partition going down

Without the majority enforcement (in this case 3), both DC1 and DC2 elect a primary ending up with split brain problem.

How to design a replica set

- MongoDB chose to a single primary because write conflicts can occur with multiple primaries.
- By enforcing a single primary to be in a replica set, the development becomes easier but there can be a periods when the replica set is read-only.
- Not recommended: a two member replica set
- Recommended
  - A majority of the set in one center: good as long as the primary data center is healthy. Otherwise, the secondary data center can’t elect a primary.
  - Equal number of servers in each data center plus a tie break server in a third location: involves three separate locations for servers.

Command	Description
rs.help()	
rs.status()	Returns the current state of the replica set
rs.initiate()	Initializes a replica set using default parameters
rs.add(“host:port”)	Add a member server to the replica set
rs.addArb(“host:port”)	Add an arbiter to the replica set
rs.stepDown()	Makes the primary server step down and forces the election of a new primary member.
rs.remove(“host:port”)	Removes a given member from a replica set.
rs.conf()	Displays the configuration structure of the current replica set
db.isMaster()	Not specific to a replica set. Allows an application or driver to determine whether a particular connected instance is the primary server.

Read Preference

An application can set a read preference to select which member of a replica set it wish to read from.

Primary	The default read preference. Reads will only be directed at the primary.
PrimaryPreferred	Reads will be directed at the primary, unless there is no primary available; then reads will be directed at a secondary.
Secondary	Reads will only be directed at secondary nodes. If no secondary is available, this option will generate an exception.
SecondaryPreferred	Reads will be directed at a secondary unless none is available; then reads will be directed at a primary. This corresponds to the behavior of the old "slaveOk" secondary read method.
Nearest	Reads from the nearest node, regardless of whether that is a primary or secondary. Nearest uses network latency to determine which node to use.

Write Concern

- How many nodes these data need to have been safely committed to before it is considered to be complete.

Option	Description
W=0 or Unacknowledged	The write will be sent, but no attempt to acknowledge if it was committed will be made.
W=1 or Acknowledged	A write must be confirmed by the primary. This is the default.
W=N or Replica Set Acknowledged	The primary must confirm the write, and *N*–1 members must replicate this write from the primary. This option is more robust but can cause delays if there is replication lag on members in your replica set or if not enough members are up at the time the write is committed because of an outage or the like.
W=Majority	A write must be written to the primary and replicated by enough members that a majority of members in the set have confirmed the write. As with w=*N*, this can cause problems during outages or if there is replication lag.
J=true or Journaled	Can be used with w= write concerns to specify that the write must be persisted to the journal to be considered confirmed.
Wtimeout=milliseconds	Wtimeout causes an operation to return an error and expire after the given number of milliseconds have expired, even if the operation here would eventually succeed.