QUORUMS AND FAULT TOLERANCE

Module 4
Fall 2020

George Porter

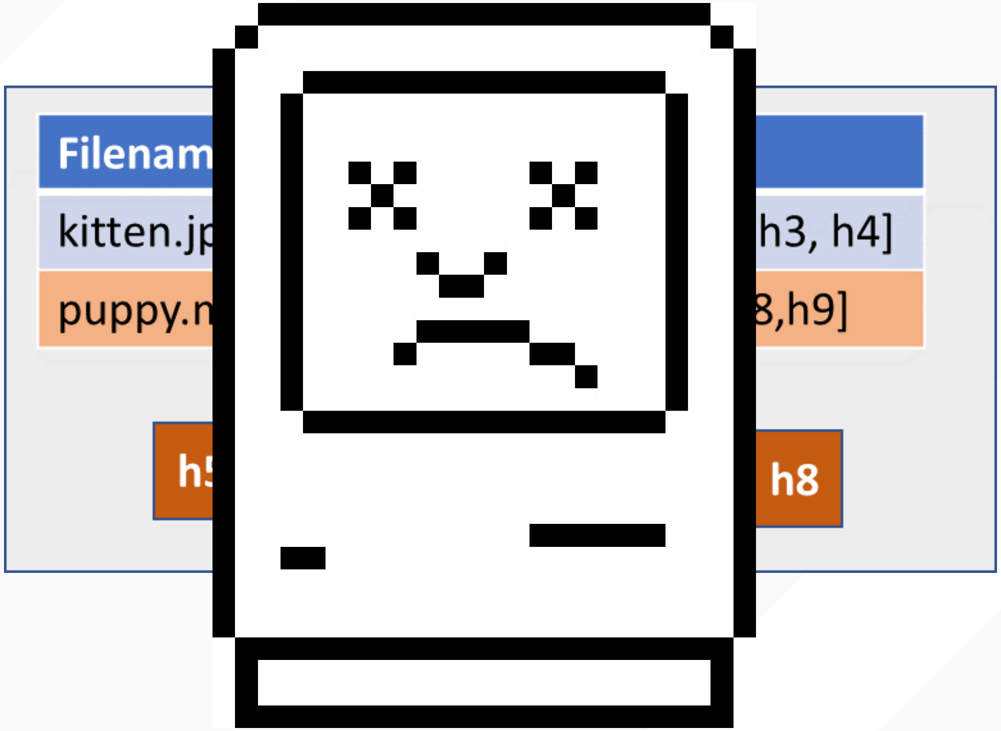
UC San Diego

ATTRIBUTION

- These slides are released under an Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0) Creative Commons license
- These slides incorporate material from:
  - Tanenbaum and Van Steen, Dist. Systems: Principles and Paradigms
  - Kyle Jamieson, Princeton University (also under a CC BY-NC-SA 3.0 Creative Commons license)

SURFSTORE METADATA SERVER PROBLEM

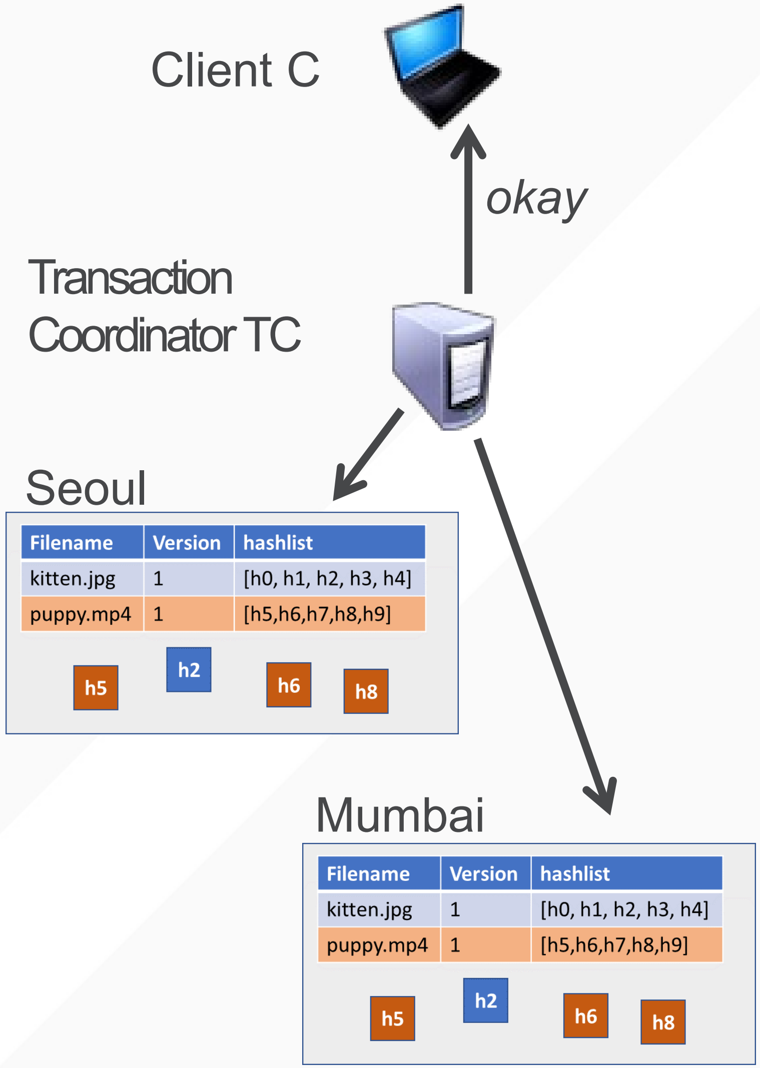
```
UpdateFile(
  file="kitten.jpg",
  ver=2,
  hashlist = {h3,h4}
);
```



All data is lost!

Surfstore Client

IDEA 1: ADAPT TWO-PHASE COMMIT TO SAVE DATA



Seoul

Filename	Version	hashlist
kitten.jpg	1	[h0, h1, h2, h3, h4]
puppy.mp4	1	[h5,h6,h7,h8,h9]

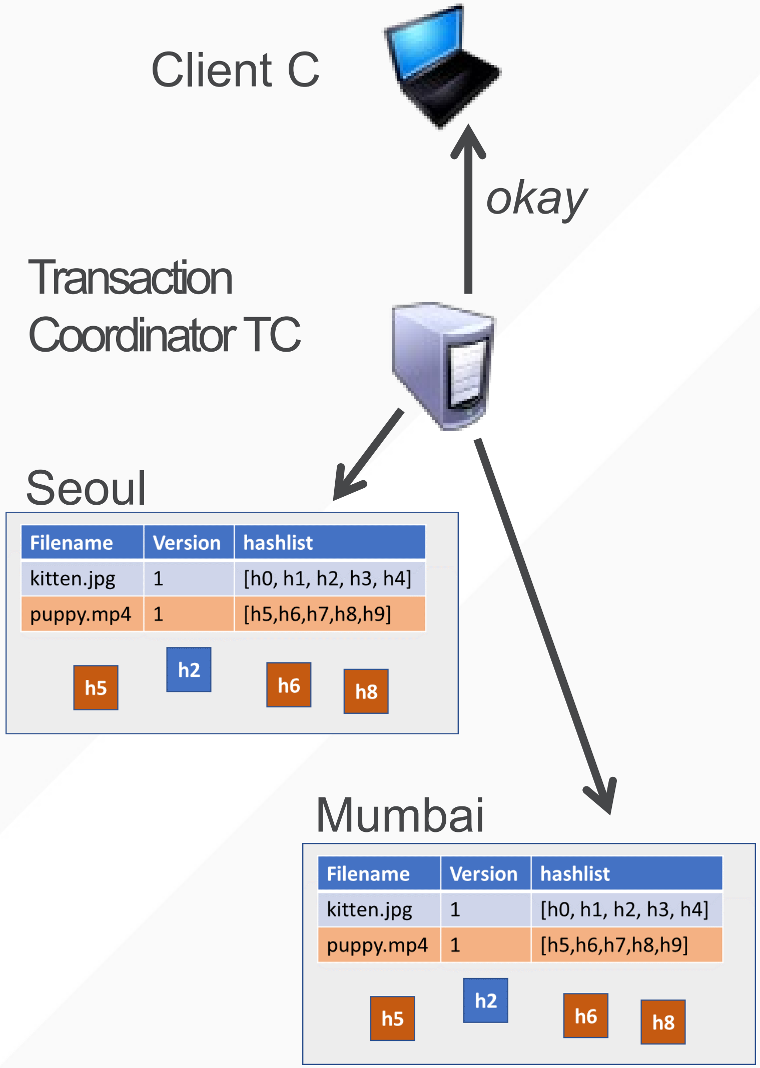
Mumbai

Filename	Version	hashlist
kitten.jpg	1	[h0, h1, h2, h3, h4]
puppy.mp4	1	[h5,h6,h7,h8,h9]

1. **C → TC:** *"go!"*
2. **TC → Seoul (S), Mumbai (M):** *"prepare!"*
3. **S, M → P:** *"yes"* or *"wrong_version"*
4. **TC → S, M:** *"commit!"* or *"abort!"*
   - **TC** sends ***commit*** if **both** say *yes*
   - **TC** sends ***abort*** if **either** say *no*
5. **TC → C:** *"okay" or "failed"*

- **S, M** commit on receipt of commit message

IDEA 2: ASSUME TC DOESN’T FAIL (FOR NOW)



Seoul

Filename	Version	hashlist
kitten.jpg	1	[h0, h1, h2, h3, h4]
puppy.mp4	1	[h5,h6,h7,h8,h9]

Mumbai

Filename	Version	hashlist
kitten.jpg	1	[h0, h1, h2, h3, h4]
puppy.mp4	1	[h5,h6,h7,h8,h9]

1. **C → TC:** *“go!”*
2. **TC → Seoul (S), Mumbai (M):** *“prepare!”*
3. **S, M → P:** *“yes” [why always yes?]*
4. **TC → S, M:** *“commit!”*
   - **TC** sends ***commit***
5. **TC → C:** *“okay”*

- **S, M** commit on receipt of commit message
- ***Why do we still need the commit?***

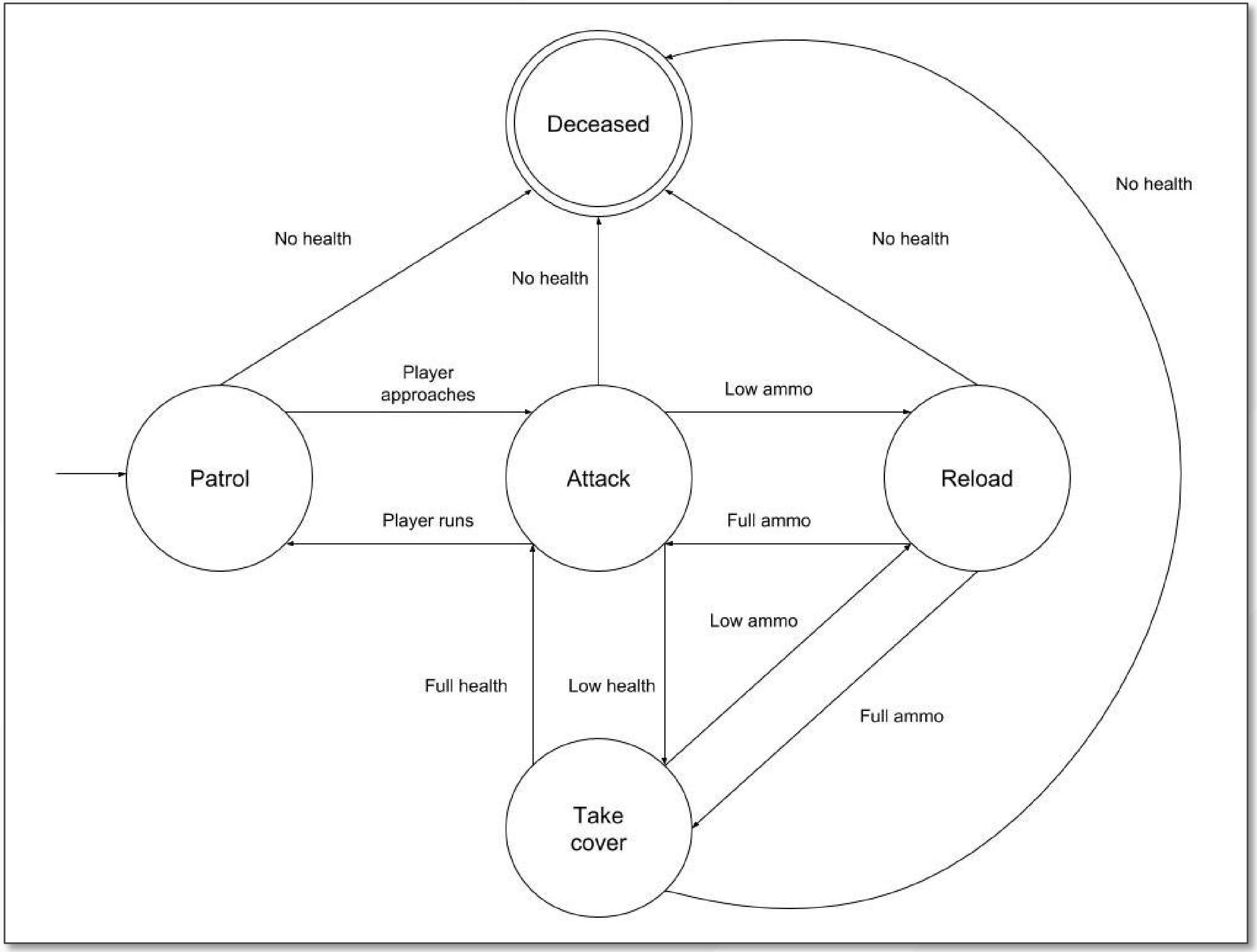
NETWORK PARTITIONS

- Some failure (either network or host) keeps replicas from communicating with one another
- Two-phase commit (even if we assume all replicas agree) only works if all nodes can be contacted
- How to proceed with read/write transactions in case where not all replicas can be contacted?

QUORUM-BASED PROTOCOLS

- Idea: Tell client that a file's version is updated after a majority of SurfStoreServers get the update
- Form a "read quorum" of size N_R
  - Contact N_R servers and read all their versions
  - Select highest version as the "correct" version
- Form a "write quorum" of size N_W
  - Contact N_W servers
  - Increment the highest version from that set
  - Write out that new version to the servers in the write quorum

REPRESENTING A SERVICE AS A FINITE STATE MACHINE

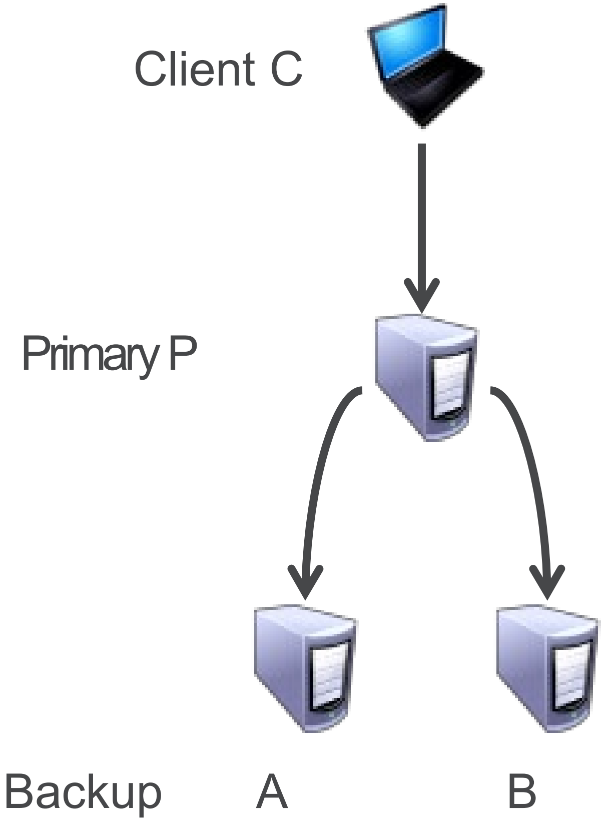


STATE MACHINE REPLICATION

- **Any server** is essentially a ***state machine***
  - Operations **transition** between states

- Need an op to be executed on all replicas, or none at all
  - *i.e.*, we need **distributed** **all-or-nothing atomicity**
  - If op is deterministic, replicas will end in same state

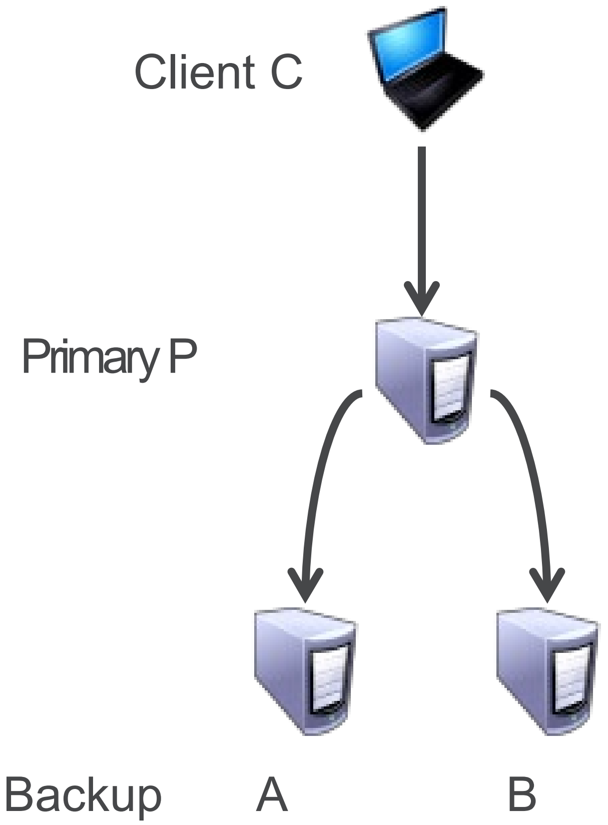
Two phase commit protocol



1. **C → P:** *“request <op>”*
2. P → **A, B:** *“prepare <op>”*
3. **A, B → P:** *“prepared”* or *“error”*
4. P → **C:** *“result exec<op>”* or *“failed”*
5. P → A, B: *“commit <op>”*

What if primary fails?
Backup fails?

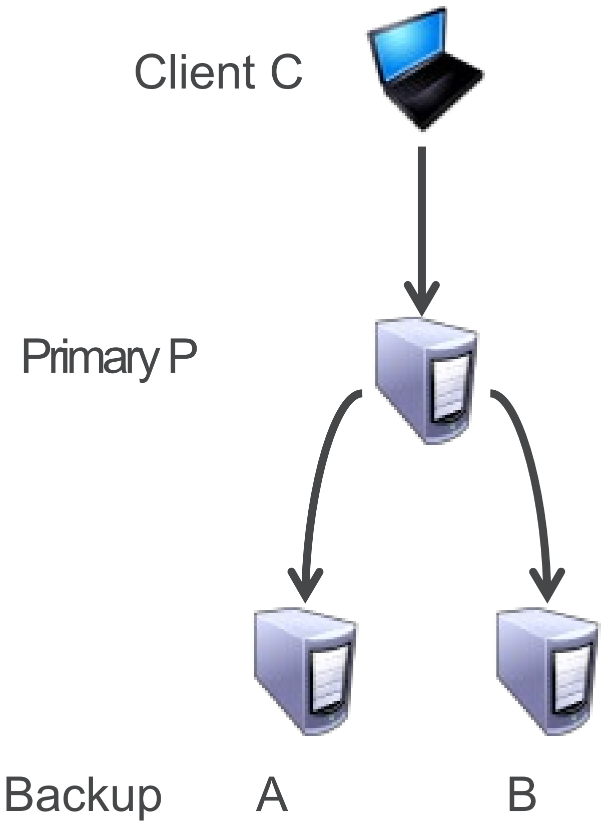
Two phase commit protocol

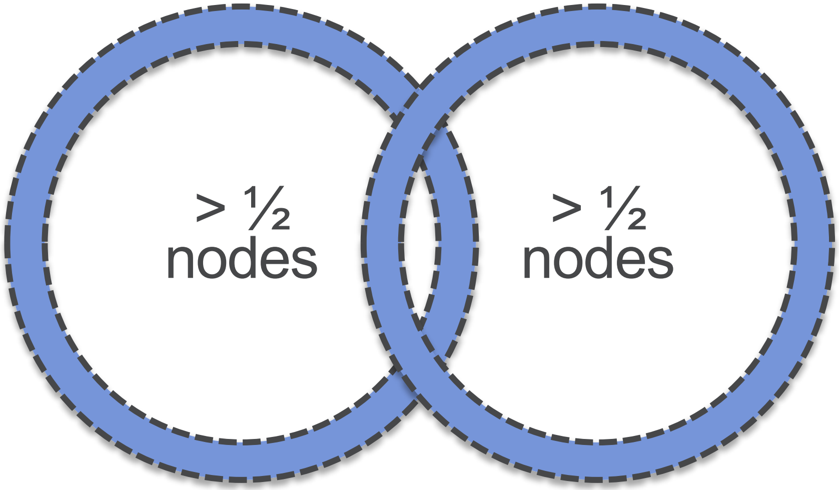


1. **C → P:** *"request <op>"*
2. P → **A, B:** *"prepare <op>"*
3. **A, B → P:** *"prepared"* or *"error"*
4. P → **C:** *"result exec<op>"* or *"failed"*
5. P → A, B: *"commit <op>"*

"Okay" (i.e., op is stable) if written to > ½ backups

Two phase commit protocol





- Commit sets always overlap ≥ 1 node
- Any >½ nodes guaranteed to see committed op

CONSTANTS AND CONSTRAINTS

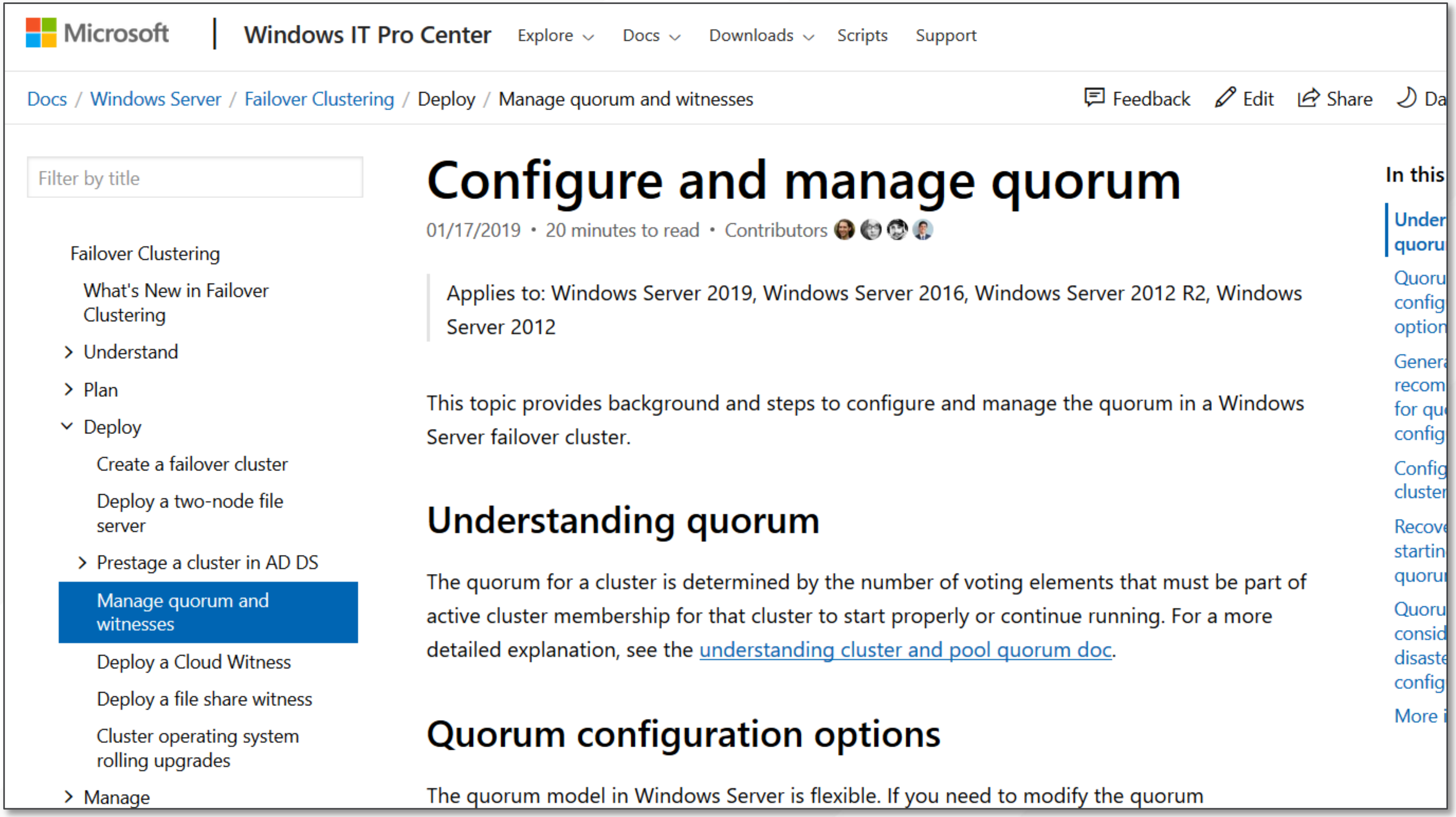
- N: Total #Replicas
- N_R: #Replicas in Read Quorum
- N_W: #Replicas in Write Quorum
- Constraints for *strong* consistency:
    1. $N_R + N_W > N$
    2. $N_W > N/2$

QUORUM CONSENSUS

- Write operations can be propagated in background to replicas not in quorum
  - Assumes eventual repair of any network partition

- Operations are slowed by the necessity of first gathering a quorum
  - Though previously, all writes had to go to all replicas
    - With quorum system, must only contact subset of replicas

QUORUMS IN MICROSOFT ACTIVE DIRECTORY


Microsoft
Windows IT Pro Center
Explore
Docs
Downloads
Scripts
Support
Docs / Windows Server / Failover Clustering / Deploy / Manage quorum and witnesses
Feedback
Edit
Share
Da
Filter by title
Failover Clustering
What's New in Failover Clustering
Understand
Plan
Deploy
Create a failover cluster
Deploy a two-node file server
Prestage a cluster in AD DS
Manage quorum and witnesses
Deploy a Cloud Witness
Deploy a file share witness
Cluster operating system rolling upgrades
Manage
In this
Under quoru
Quoru config option
Genera recom for qu config
Config cluster
Recove startin quoru
Quoru consid disaste config
More i

Configure and manage quorum

01/17/2019 • 20 minutes to read • Contributors

> Applies to: Windows Server 2019, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012

This topic provides background and steps to configure and manage the quorum in a Windows Server failover cluster.

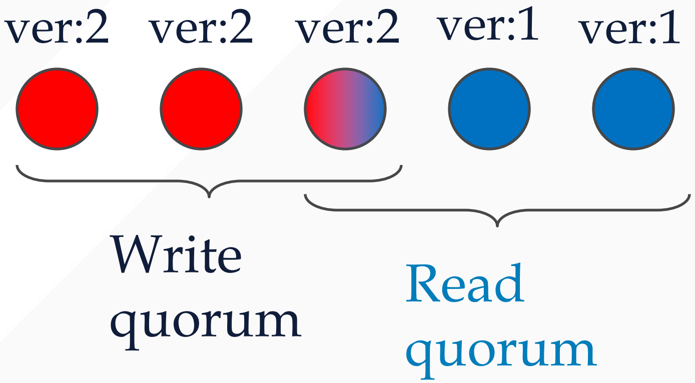
Understanding quorum

The quorum for a cluster is determined by the number of voting elements that must be part of active cluster membership for that cluster to start properly or continue running. For a more detailed explanation, see the understanding cluster and pool quorum doc.

Quorum configuration options

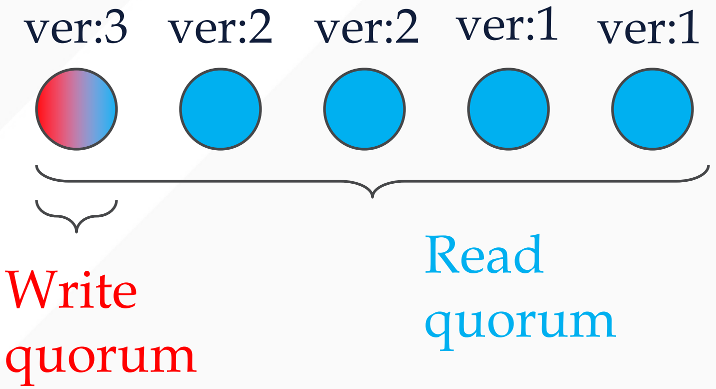
The quorum model in Windows Server is flexible. If you need to modify the quorum

QUORUM EXAMPLE



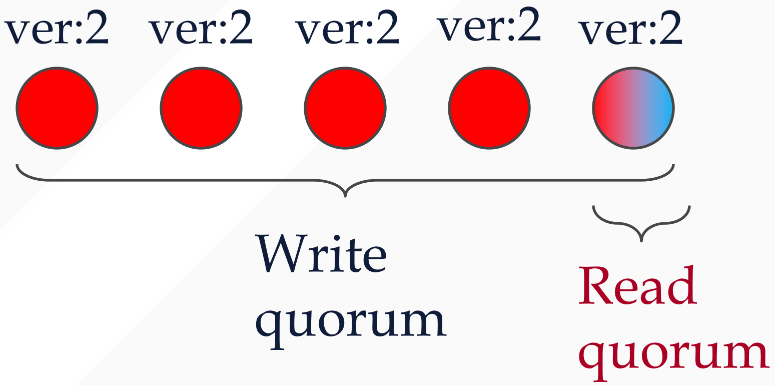
- 5 replicas, read quorum: 3, write quorum: 3
  - R+W>5 votes ensures overlap between any read/write quorum
- How does this perform for reads?
- How does this perform for writes?

QUORUM EXAMPLE



- 5 replicas, read quorum: 5, write quorum: 1
  - R+W>5 votes ensures overlap between any read/write quorum
- How does this perform for reads?
- How does this perform for writes?

QUORUM EXAMPLE



- 5 replicas, read quorum: 1, write quorum: 5
  - R+W>5 votes ensures overlap between any read/write quorum
  - Also called ROWA (read one, write all)
- How does this perform for reads?
- How does this perform for writes?

STRONGLY CONSISTENT AND EVENTUALLY CONSISTENT EXAMPLES

UC San Diego

UC San Diego