CSci 5105

Introduction to Distributed Systems

Replication

Today

- Replication
- Implementation of consistency protocols
- Chapter 7 TVS

Replication <=> Consistency

- Data are replicated for availability

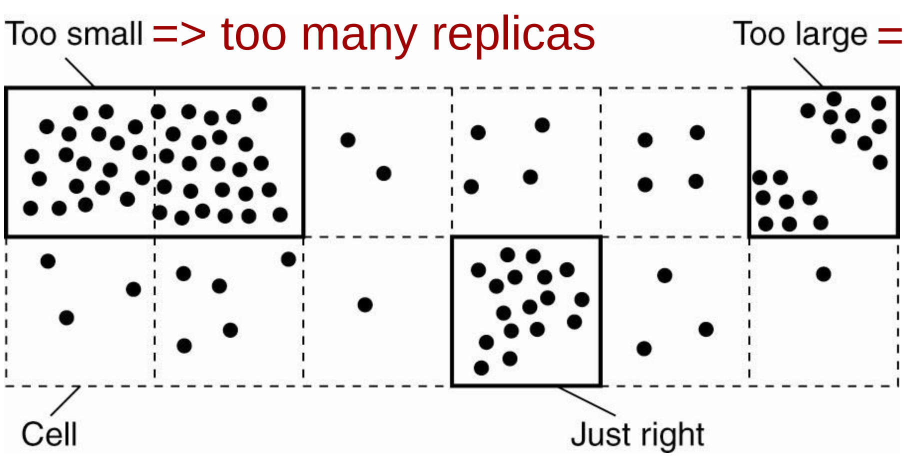
- Data are replicated for performance
  - Scaling in numbers: throughput
  - Scaling in geographical area: latency

When to replicate?

- Proactive
  - a-priori
- Reactive
  - as needed
- Tradeoffs?

Replica-Server Placement

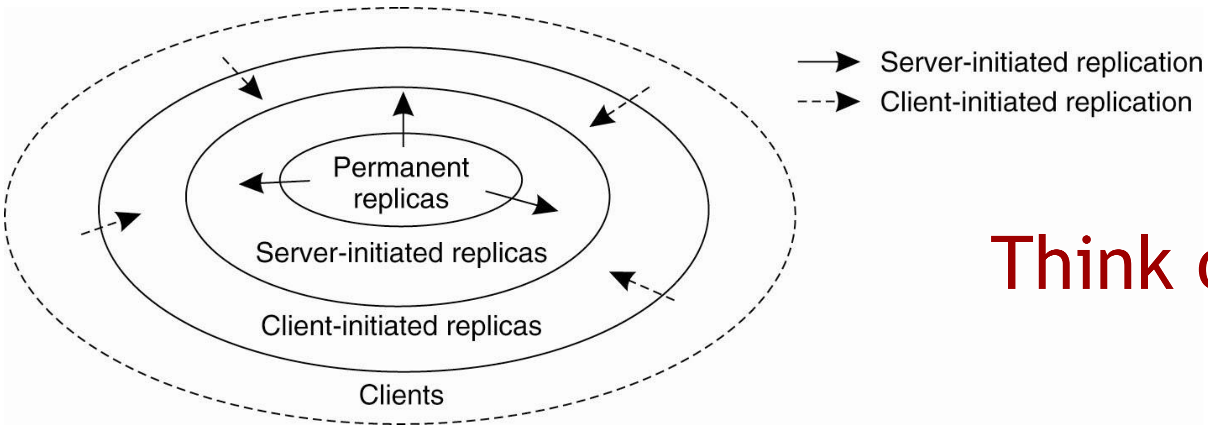
- How many server replicas?
- Where should they be placed in the network?
  - Not often studied
- Node (dot) is an interested data client
  - Each cell is handled by a replica
  - Place replica in K most dense cells; K is given



If same size cell requirement

Content Replication and Placement

- Types of content placement



Think of web content

- Permanent
  - distributed data-store (across servers~local domain)
  - geo-distributed mirrors
- Client-initiated: caching

Server-Initiated Replicas

- Simple scheme (how many + ~ placement)
  - Server counts client access to file F
  - Assume clients are sent to nearest server P which in turn routes to server S holding F
  - `#Req (F, S) > rep-threshold`, replicate F
  - `#Req (F, S) < del-threshold`, delete F (unless last one)
  - rep-threshold is usually > del-threshold
  - If > ½ of requests arrive to server Q from a particular server P, migrate F from Q to P
    - Why?

Keeping replicas in synch: update propagation

How to propagate data updates between replicas?

Consistency tells us what to propagate, but different options for *how*

1. Propagate only a notification of an update

2. Transfer data from one copy to another

3. Propagate the update *operation* to other copies: Active replication

Who initiates update propagation?

- Replicas include client caches
- Push: server-based
- Pull: client-based
- Tradeoffs

Issue	Push-based	Pull-based
State at server	List of client replicas and caches	None
Messages sent	Update (and possibly fetch update later)	Poll and update
Response time at client	Immediate (or fetch-update time)	Fetch-update time

if update
notification

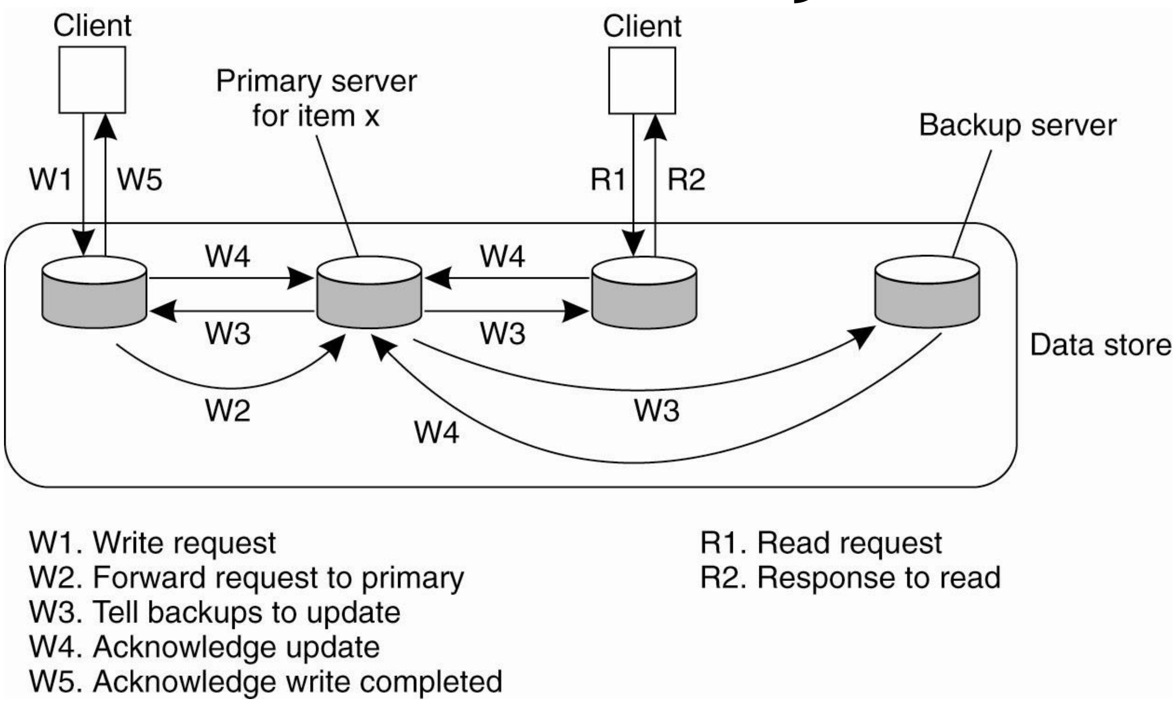
Leases

- Timed control: optimizations
- Promise server will push updates for a specified time
  - `hockey.com` only wants `nbcsports.com` updates until the hockey gold medal game finishes
  - after this client must poll
  - allows switching between push and pull
- Age-based lease
  - data is guaranteed to be valid for time K
  - after K, can poll

Implementation

- How are consistency models actually implemented?
- Protocol and architecture

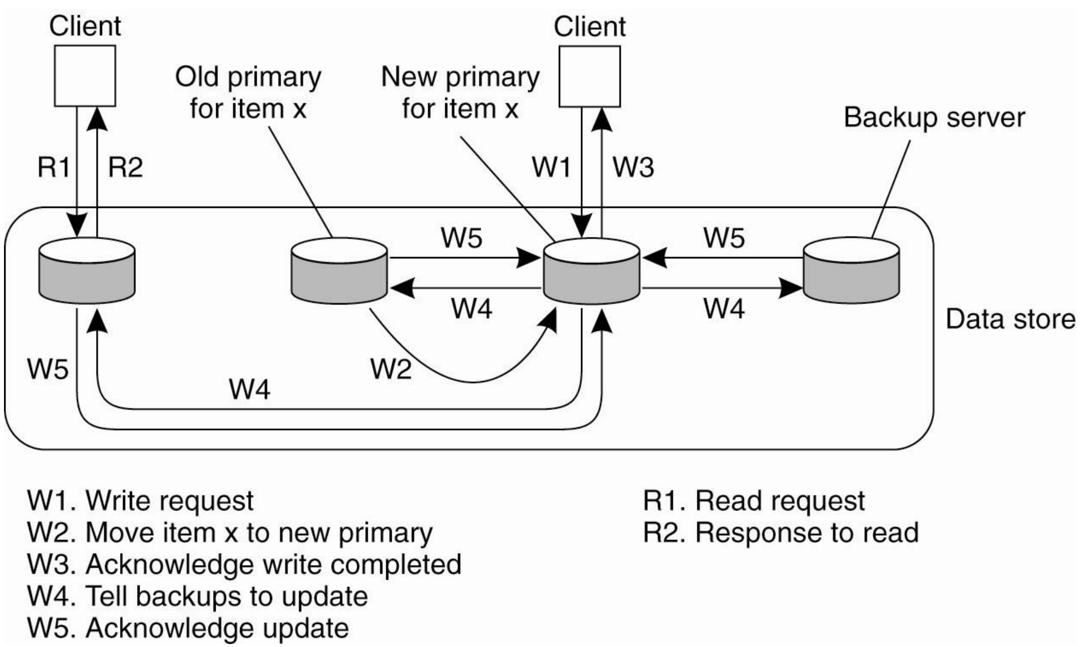
Remote-Write Protocols: Primary-Backup



- Good for sequential consistency
- Weakness?

Local-Write Protocols

- Primary migrates to current writer
- Advantages? Disadvantages?



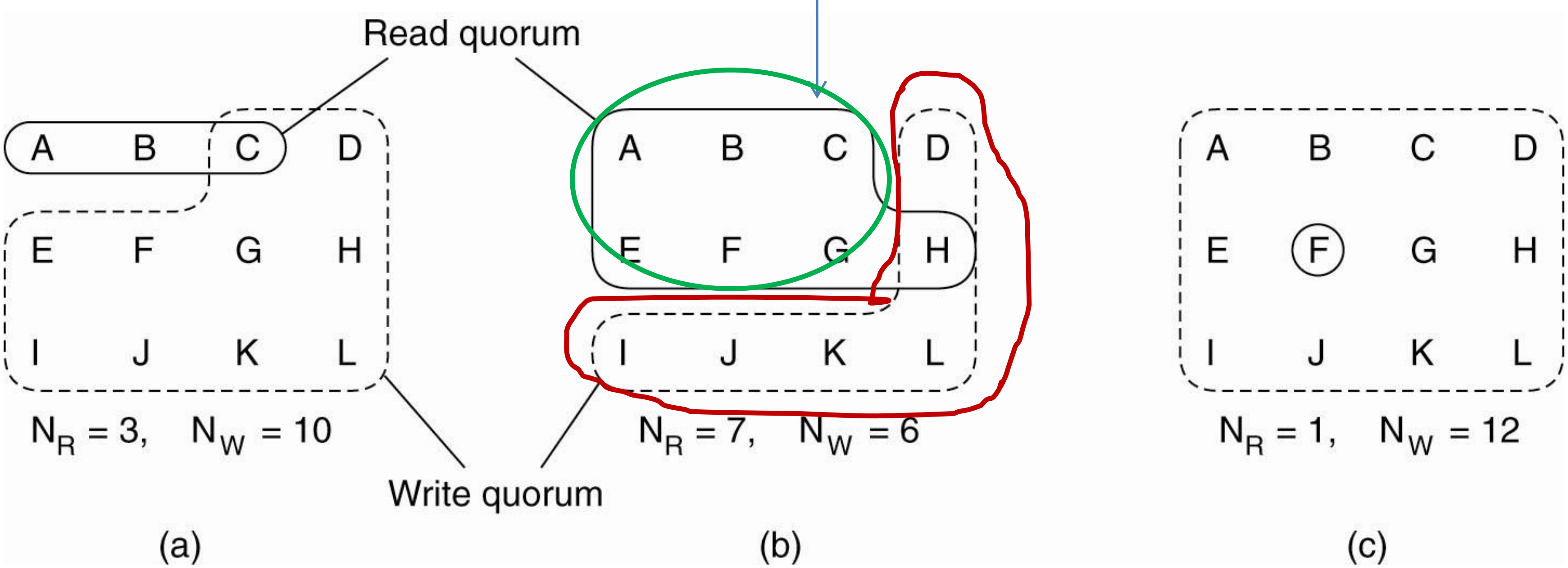
Replicated-Write Protocols

- These schemes write to a single primary
  - Global primary
  - Changing primary
  - Problems?

- Voting scheme
  - More general
  - Writes to any replica

Quorum-Based Protocols

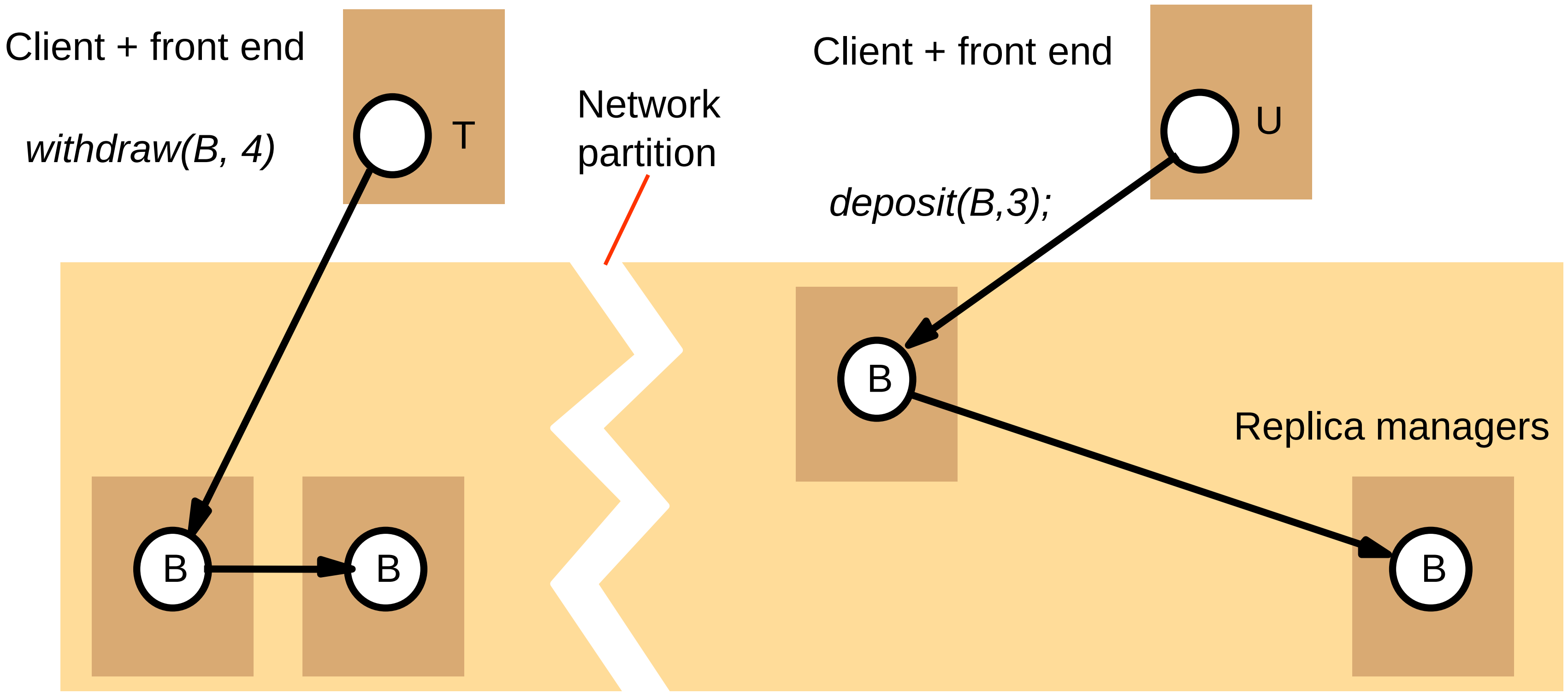
- Voting: N servers
  1. $N_r + N_w > N$ (read-write conflicts)
  2. $N_w > N/2$ (write-write conflicts)



Client-centric consistency

- Many client replicas
  - assume one server or servers are consistent
  - cache coherence ~ DFS
- Read-only caches: updates only at server
  - server-directed vs. client-directed coherence
  - server invalidates vs. client polling
- Read/write cache
  - write-through: write locally and to server
  - write-back: write locally, delay to server
  - same coherence choices

Network Partitions



- Uh Oh: how can we achieve any kind of consistency?

Next Time

Next topic: Fault Tolerance

Read Chapter 8 TVS

Have a great weekend!