Effective Replica Maintenance for Distributed Storage Systems

USENIX NSDI' 06

Byung-Gon Chun, Frank Dabek, Andreas Haeberlen, Emil Sit, Hakim Weatherspoon,
M. Frans Kaashoek, John Kubiatowicz, and Robert Morris

MIT Computer Science and Artificial Intelligence Laboratory,
Rice University/MPI-SWS, University of California, Berkeley

Present by: Hongyu Gao

Adapted from Szu-Jui, Wu's slides

Definitions

- **Distributed Storage System**: a network file system whose storage nodes are **dispersed over the Internet**
- **Durability**: objects that an application has put into the system are **not lost** due to disk failure
- **Availability**: get will be able to return the object **promptly**

Outline

- Motivation
- Understanding durability
- Improving repair time
- Reducing transient failure cost
- Conclusion

Motivation

- To store immutable objects **durably** at a **low bandwidth cost** in a distributed storage system

Understanding Durability

Providing Durability

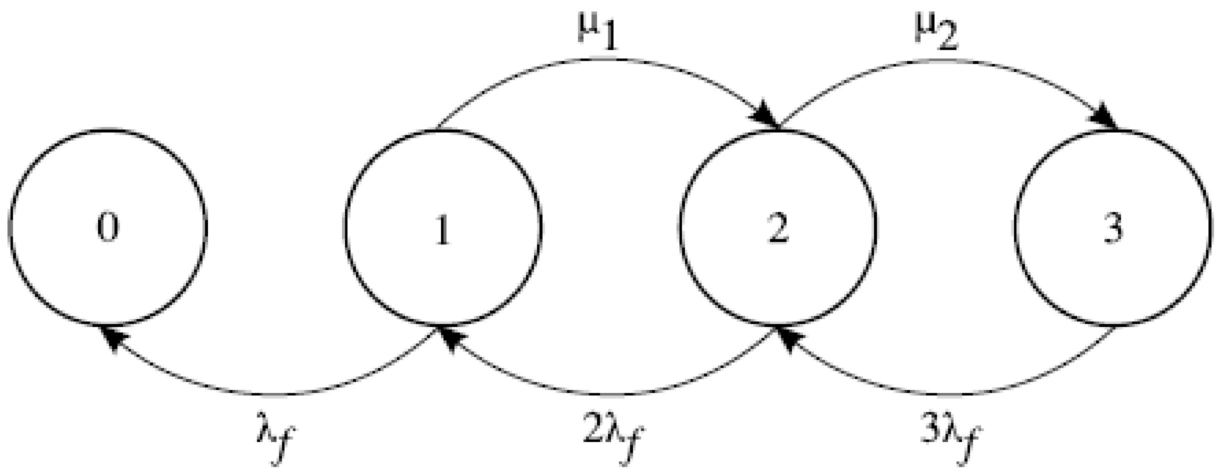
- Durability is less expensive and more useful than availability
- Challenges
  - Replication algorithm: Create new replica faster than losing them
  - Reducing network bandwidth
  - Distinguish transient failures from permanent disk failures

Challenges to Durability

- Create new replicas **faster** than replicas are destroyed
  - Creation rate < failure rate ➔ system is **infeasible**
    - Insight: Higher number of replicas do not allow system to survive a higher average failure rate
  - Creation rate = failure rate + ε (ε is small) ➔ burst of failure may destroy all of the replicas

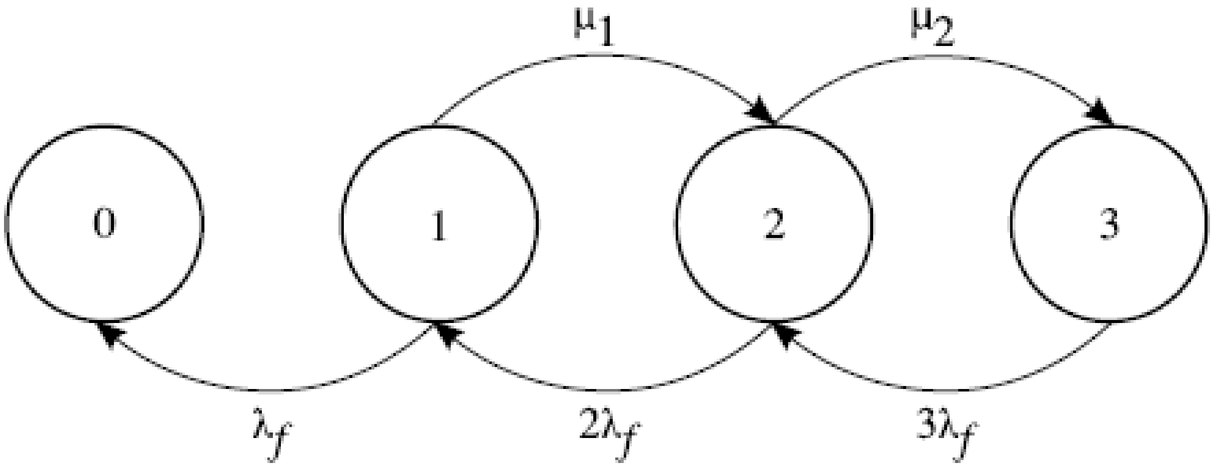
Number of Replicas as a Birth-Death Process

- Assumption: independent exponential inter-failure and inter-repair times
- λ_f : average failure rate
- μ_i: average repair rate at state i
- r_L : lower bound of number of replicas (r_L = 3 in this case)



Model Simplification

- Fixed μ & allowing transition from state 0 to 1 ➔ the equilibrium number of replicas is **$\Theta = \mu / \lambda$**
- If Θ < 1, the system can no longer maintain full replication regardless of r_L



Real-world Settings

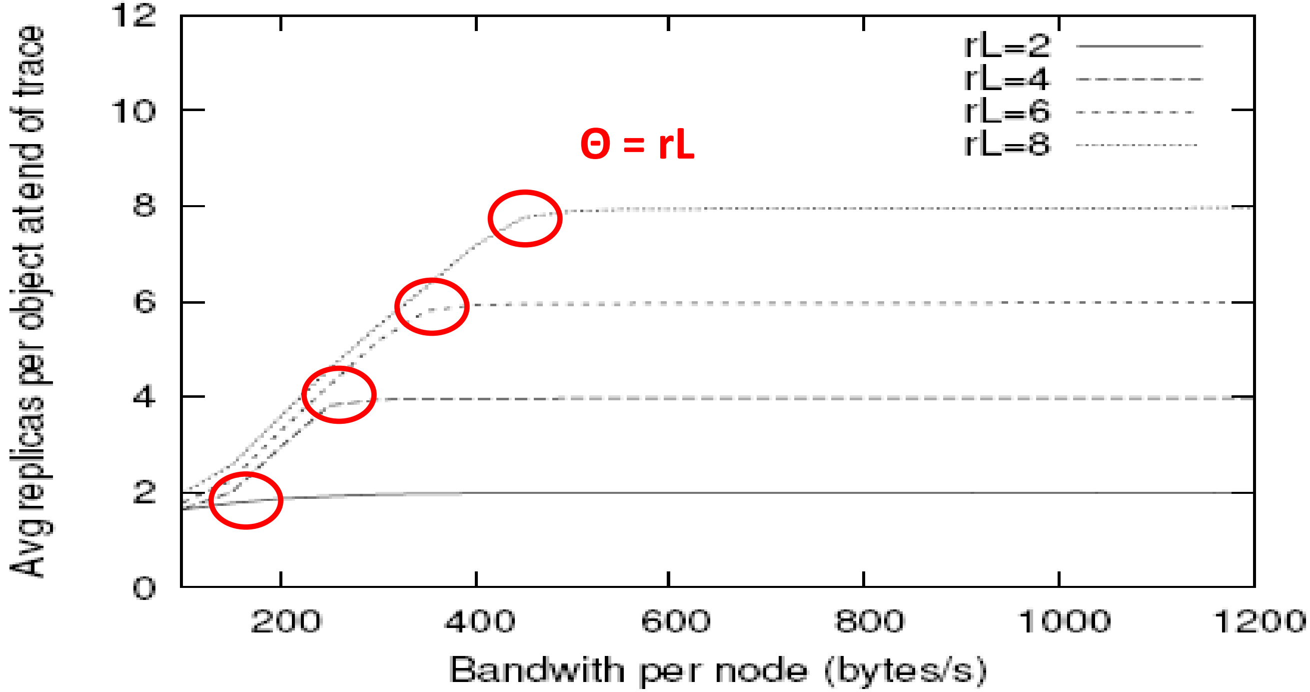
- Planetlab
  - 490 nodes
  - Average inter-failure time 39.85 hours
  - 150 KB/s bandwidth
- Assumption
  - 500 GB unique data per node
  - $r_L = 3$

- λ = 0.439 disk failures / year
- μ = 3 disk copies / year

- $\Theta = \mu / \lambda = 6.85$

Impact of Θ

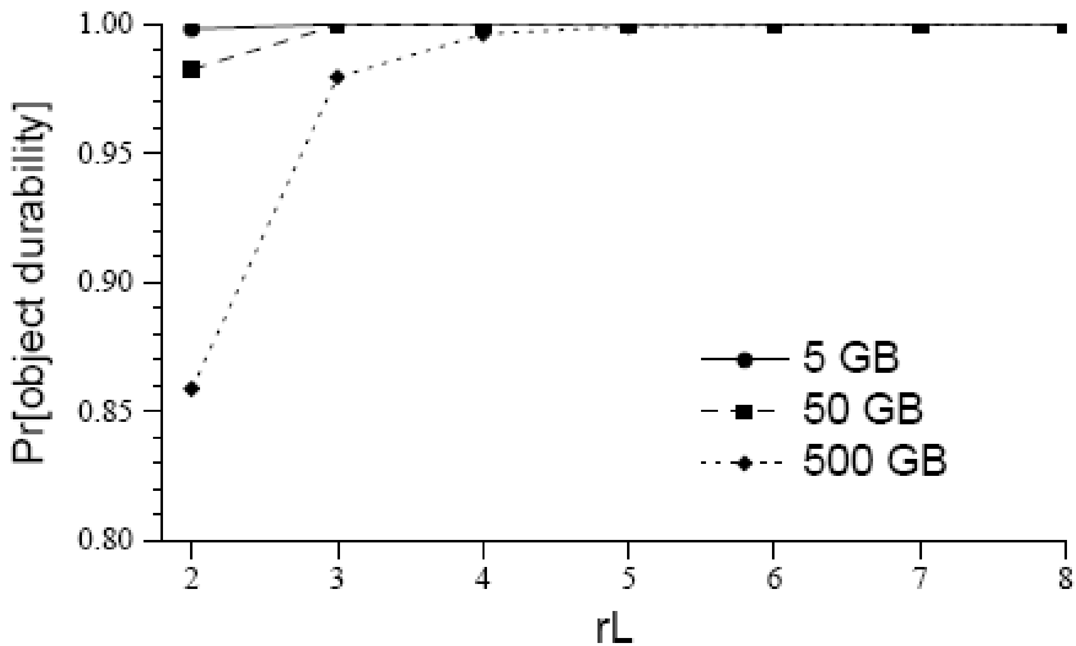
- Θ is the **theoretical upper limit** of replica number
- bandwidth ↑ ➔ μ ↑ ➔ Θ ↑
- r_L ↑ ➔ μ ↓ ➔ Θ ↓



r_L vs Durablility

- Higher r_L would cost high but tolerate more burst failures
- Larger data size ➔ μ ↓ ➔ need higher r_L

Analytical results from Planetlab traces (4 years)



Choosing r_L

- Guidelines
  - Large enough to ensure durability
  - One more than the maximum burst of simultaneous failures
  - Small enough to ensure $r_L <= \Theta$

Improving Repair Time

Definition: Scope

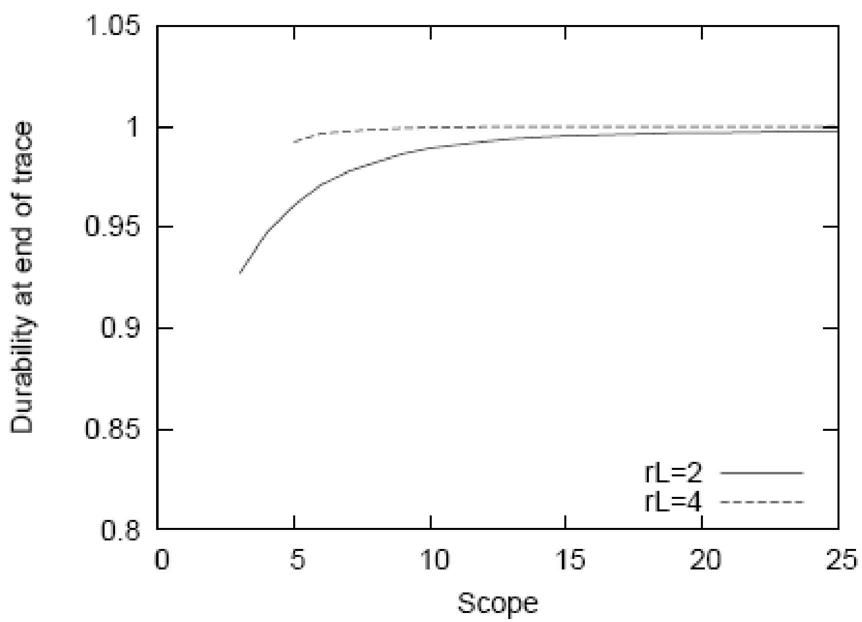
- Each node, n, designates a **set** of other nodes that can potentially hold copies of the objects that n is responsible for. We call the **size** of that set the node's **scope**.
- scope $\in [r_L, N]$
  - N: number of nodes in the system

Effect of Scope

- Small scope
  - Easy to keep track of objects
  - More effort of creating new objects
- Big scope
  - **Reduces repair time**, thus increases durability
  - Need to monitor many nodes

Scope vs. Repair Time

- Scope ↑ ➔ repair work is spread over more access links and completes faster
- r_L ↓ ➔ scope must be higher to achieve the same durability



Reducing Transient Costs

The Reasons

- Not creating new replicas for transient failures
  - Unnecessary costs (replicas)
  - Waste resources (bandwidth, disk)

- Solutions
  - **Reintegration**
  - **Timeouts**
  - **Batch**

Reintegration

- Reintegrate replicas stored on nodes after transient failures
- System must be able to track more than r_L number of replicas
- Depends on **a**: the average fraction of time that a node is available

Reintegration, cont'd

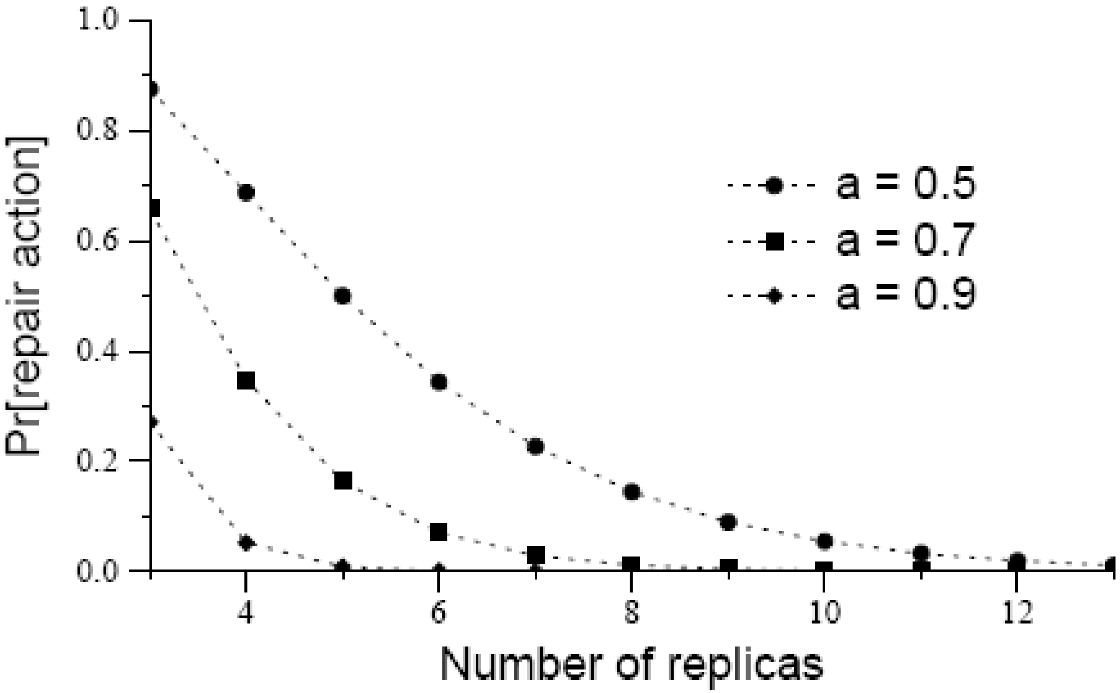
- Pr[new replica needs to be created] == Pr[less than r_L replicas are available] :

$$\Pr[R < r_L \mid r \text{ extant copies}] = \sum_{i=0}^{r_L - 1} \binom{r}{i} a^i (1-a)^{r-i}.$$

- Chernoff bound: **$2r_L/a$** replicas are needed to keep at least r_L copies available (with high enough probability)

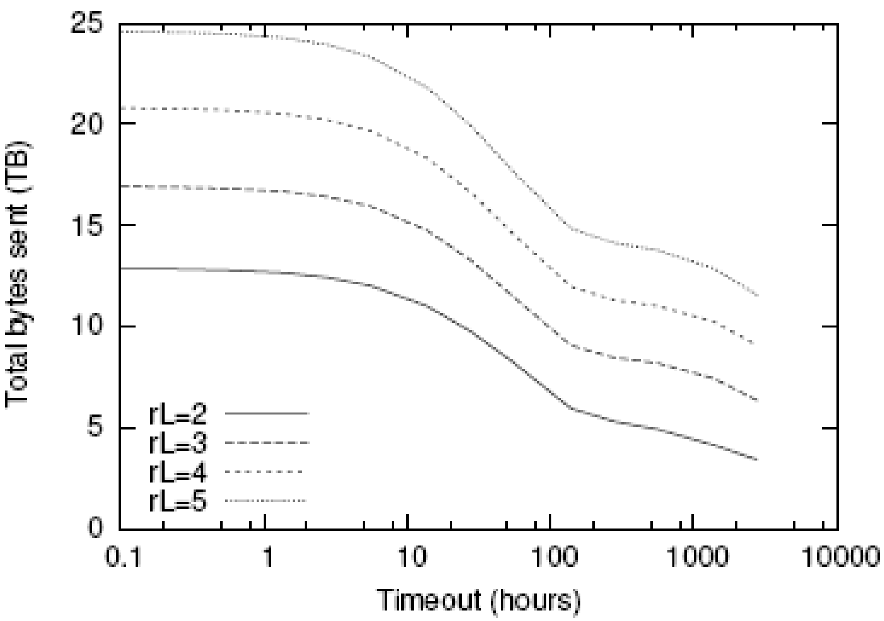
Node Availability vs. Reintegration

- Reintegrate can work safely with **$2r_L/a$** replicas
- $2/a$ is the **penalty** for not distinguishing transient and permanent failures
- $r_L = 3$

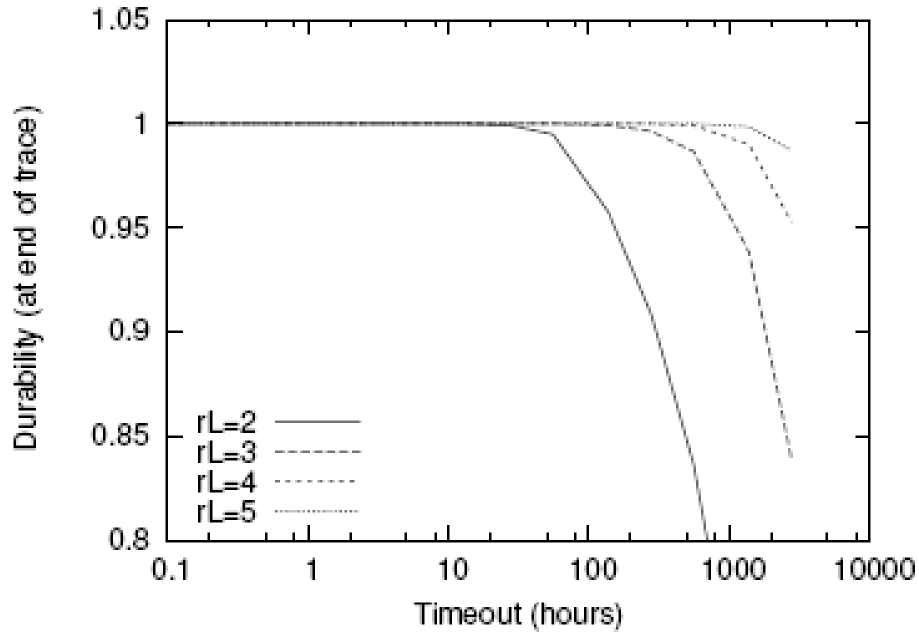


Timeouts

- Timeout > average down time
  - Average down time: 29 hours
  - Reduce maintenance cost
  - Durability still maintained

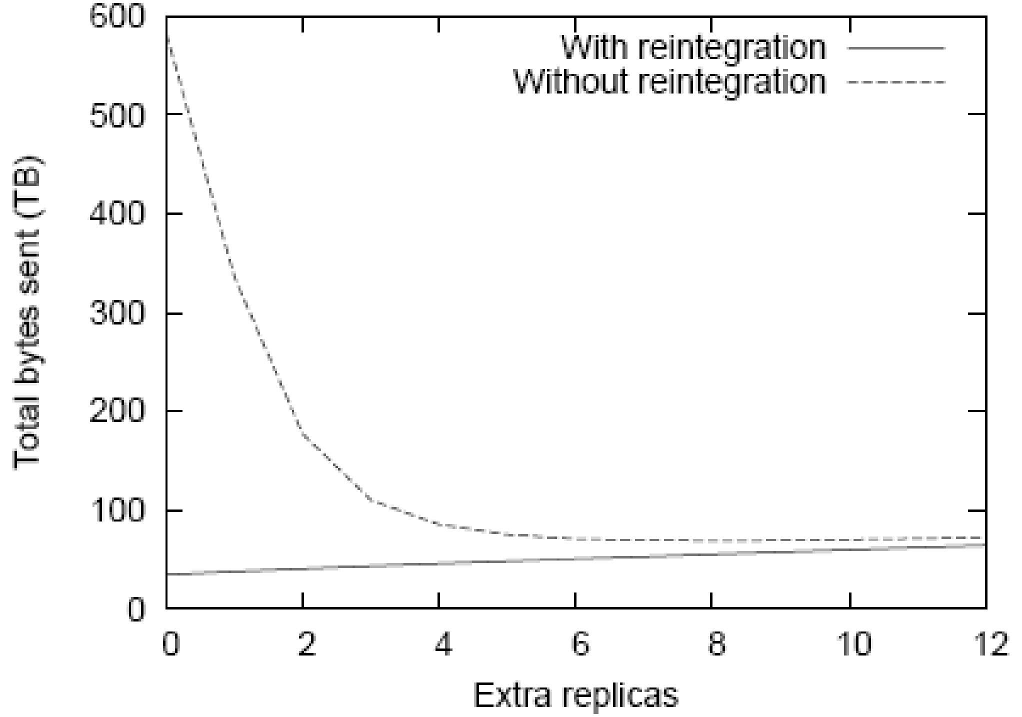


- Timeout >> average down time
  - Durability begins to fall
  - Delays the point at which the system can begin repair



Batch

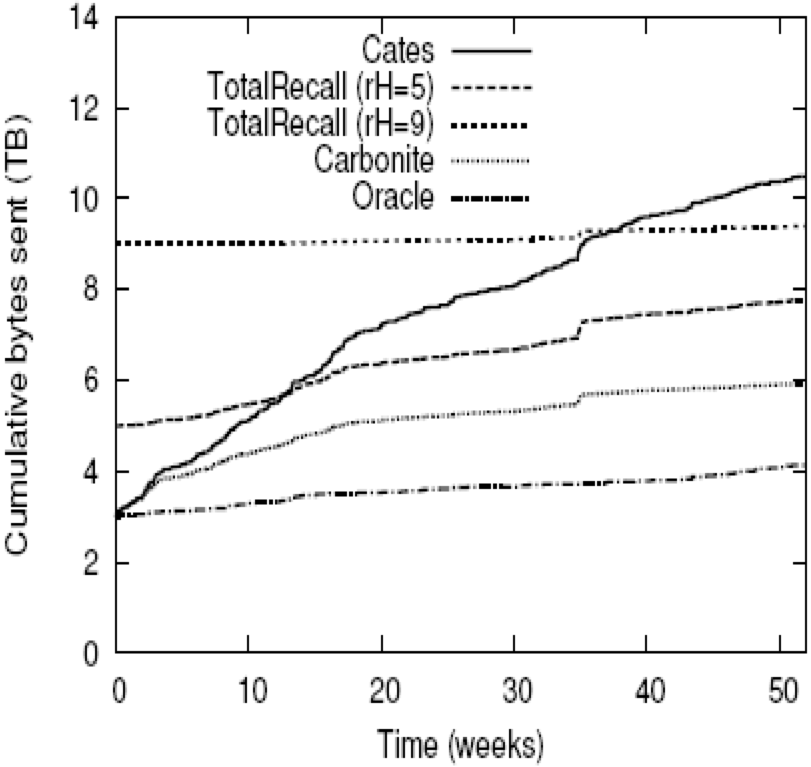
- In addition to r_L replicas, make **e** additional copies
  - Makes repair less frequent
  - Use up more resources
- $r_L = 3$

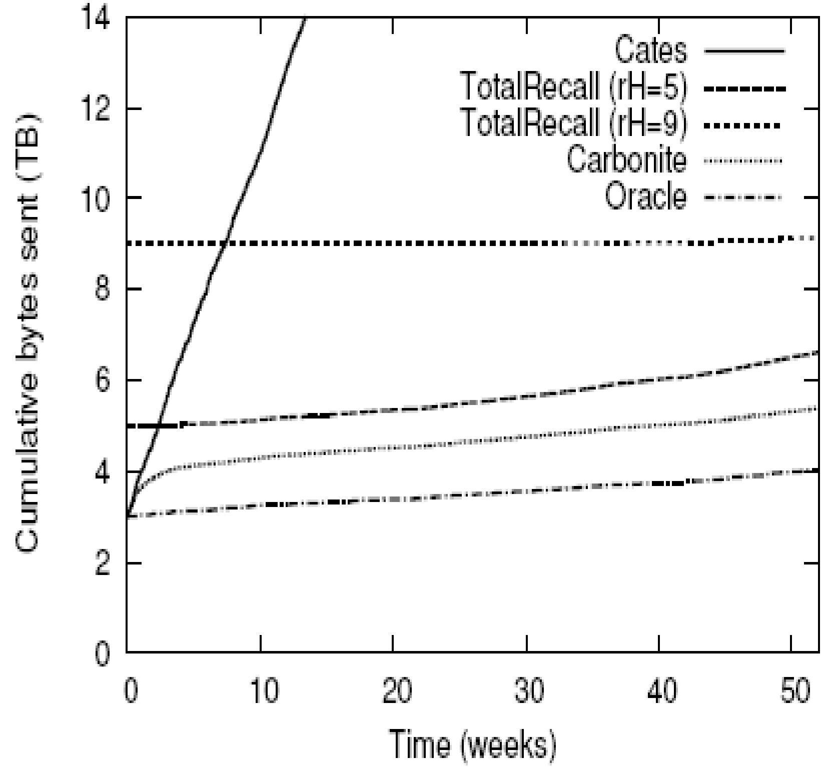


Four Replication Algorithms

- Cates
  - Fixed number of replicas r_L
  - Timeout
- Total Recall
  - Batch
- Carbonite
  - Timeout + reintegration
- Oracle
  - Hypothetical system that can differentiate transient failures from permanent failures

Effect of Reintegration





Conclusion

- Many design choices remain to be made
  - Number of replicas (depend on failure distribution and bandwidth, etc)
  - Scope size
  - Response to transient failures
    - Reintegration (extra copies #)
    - Timeouts (timeout period)
    - Batch (extra copies #)

Discussion

- Raise insightful questions:
  - Replica # ? (Not answered)
  - Scope size ? (Not answered)
  - Repair algorithm ?
- Unrealistic model for replica failure and repair