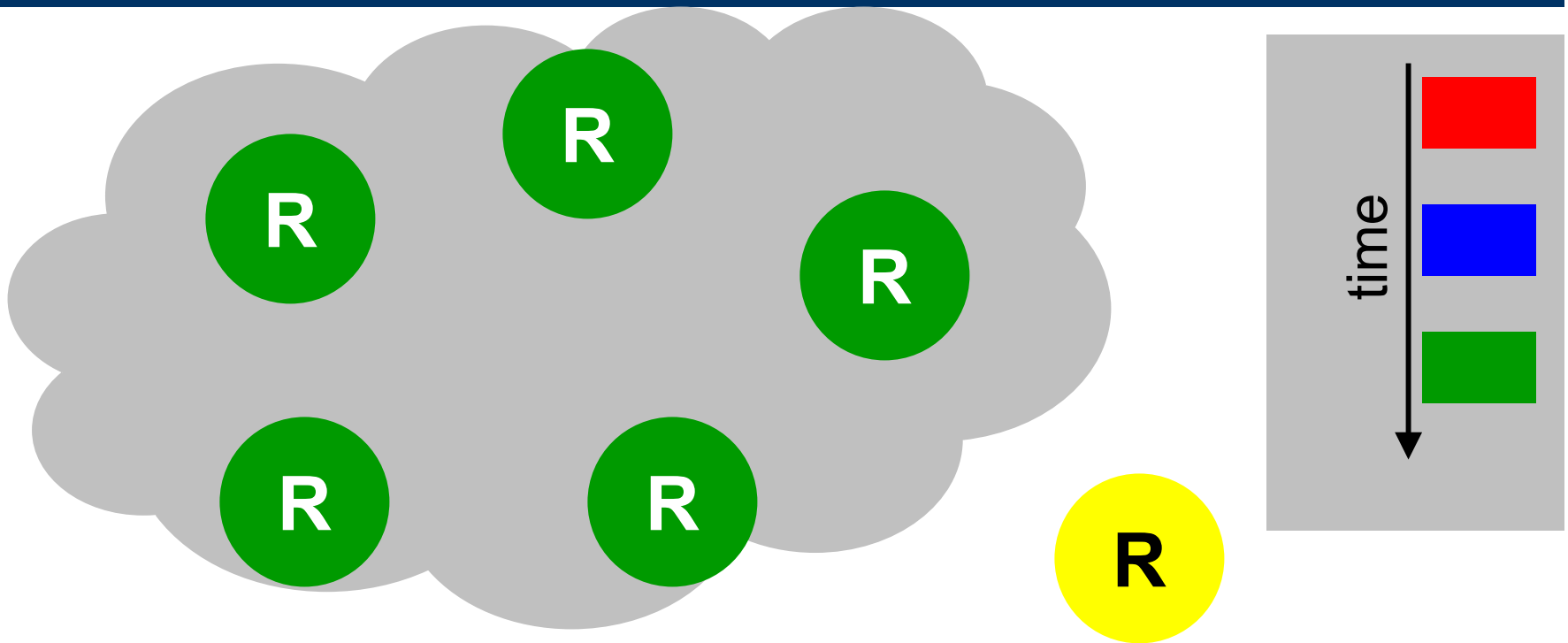
Eager Replication and mobile nodes



- Read on disconnected clients may give stale data
- Eager replication prohibits updates if any node is disconnected

Lazy replication and mobile nodes

- With Lazy group replication, we have to wait for all nodes to come online to commit

- Lazy master replication cannot work for mobile nodes and network connection is needed for transaction to complete

Example replication scenario: #1

- Replicated DNS servers
  - One primary DNS server
  - Multiple replicas
    - DNS1.UGA.EDU 128.192.1.9
    - DNS2.UGA.EDU 128.192.1.193
    - DNS3.UGA.EDU 168.24.242.249
  - Replicas use zone transfers to get an uptodate database from the the primary server
  - Transfers database every so often
  - Inconsistent state between transfers

Lazy, master replication

Example replication #2

- Palm Pilot Synchronization
- Database (your address book) is in PIM (Outlook say), Palm Desktop, your Palm device. Updates are allowed anywhere. You could authorize your secretary to add items to your Outlook
- Lazy group update

Example replication #3

- Gnutella – when you add a new song into your computer, when do the other nodes see it? Eventually

- Lazy group update

Example replication #4

- Newsgroups
- Everyone can post to newsgroup. You post in comp.risks from UGA, and your friend also posts at the same time from GATECH. My friend at Duke will see it in some order (UGA first and then GATECH or the other way around)

- Lazy group replication

Example replication #5

- Distributed databases with ACID syntax
- Eager group or master

- HARP
- Eager master

Convergence property

- If no new transactions arrive, if all the nodes are connected together, they will all converge to the same replicated state after exchanging replica updates

- Updates may be lost because of newer updates

- Commutative updates – incremental transformations that can be applied in any order

Two-tier replication

- Mobile nodes
  - Disconnected most of the time.
  - Mobile nodes store Master version and Tentative version
    - Master version on disconnected or lazy replica maybe outdated
    - Most recent value due to local updates is maintained as a tentative value
- Base Nodes
  - Always connected. Store a replica of the database. Items are mastered in base nodes

Two-tier transaction

- Base transaction
  - Work only on master data
  - Produce new master data
- Tentative transaction
  - Work on local tentative data
  - Produce new tentative versions
  - Also produce base transaction to be run at a later time on the base nodes
- Acceptance criteria for each transaction update

Key properties of two-tier replication schemes

- Mobile nodes may make tentative database updates
- Base transactions execute with single-copy serializability so the master base system state is the result of a serializable execution
- A transaction becomes durable when the base transaction completes
- Replicas at all connected nodes converge to the base system state
- If all transactions commute, there are no reconciliations

Discussion

Course project

- Project goal:
  - Solve a real world problem using technologies that we discuss in class. Think of a problem that you face in your life. Try to solve it for the course project
    - E.g. I am developing a “study search” system. I want the students to access it, not only from their desktops, but from their PalmPilots
- Since time is limited, however, I will reward those that aim high even if they do not completely succeed. The key to a successful class project is ensuring that some aspects of your work are completely done; it is hard to grade a project where nothing quite works.
- The projects will be graded as follows -- by what you discover in doing the project, how coherently you present your results, and how well you put your work in perspective with other research

Course project

- Keys to a successful project:
  - What are you trying to achieve? (e.g. I am developing a global peer-to-peer file system)
  - List the specific project goal (e.g. I am looking at the scalable communication protocol for this file system)
  - How do you measure success? (e.g. I will be successful if I can scale better than n^2 [n is number of nodes])
  - Expected obstacles (e.g. I need to run experiments on n hosts, where n is as large as possible. I need to install…)
  - Methods and tools that may be used for the implementation (e.g. I need a palm pilot and DEV kit)
  - Deliverables
  - Considerations (e.g. scalability, robustness, security)

Course project Outline

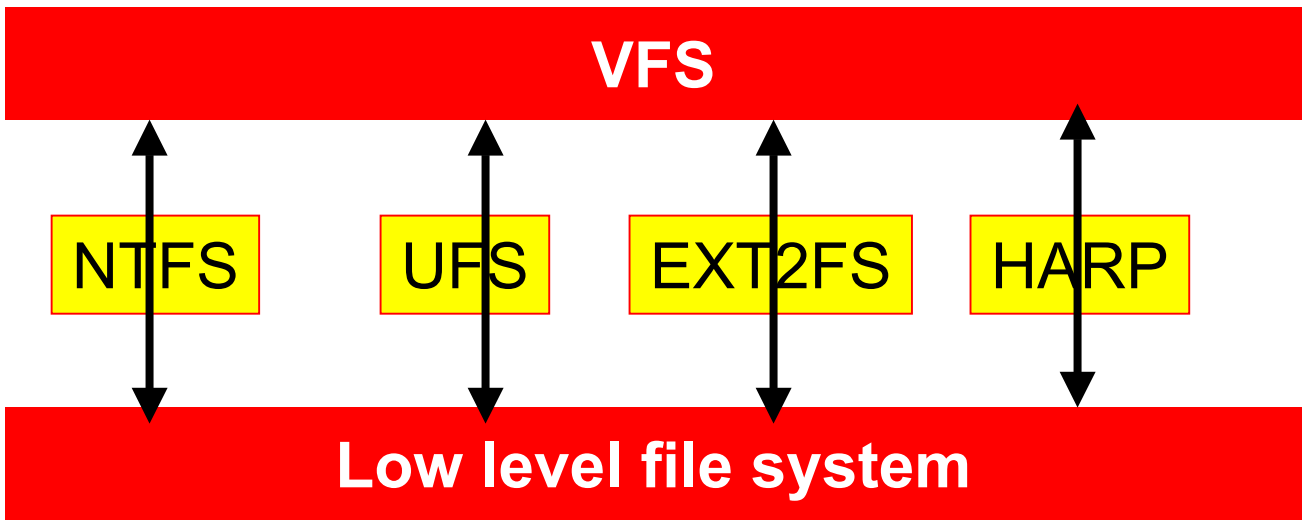
- Proposals should include:
  - a description of your topic,
  - a crisp statement of the hypothesis that you will test,
  - a statement of why you think the topic is important,
  - a description of the methods you will use to evaluate your ideas, and
  - references to at least three papers you have obtained with a summary of how they relate to your work. Proposals should not exceed 2 pages in length.

Outline

- Replication in the Harp File System, Barbara Liskov, Sanjay Ghemawat, Robert Gruber, Paul Johnson, Liuba Shrira, Michael Williams, MIT

Highly Available, Reliable, Persistent (HARP) fs

- Replicated Unix file system accessible via the Virtual File System (VFS) interface
- VFS is a software abstraction in UNIX like OS. It provides a unified approach a number of different file systems

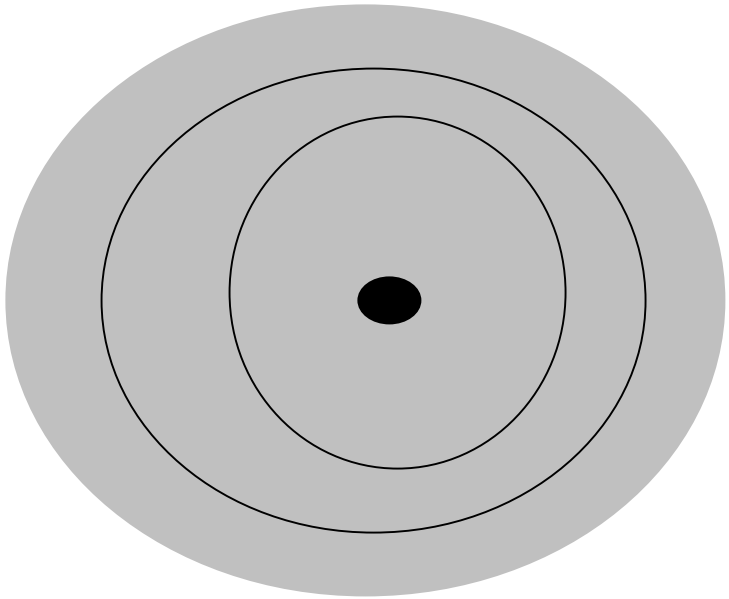


HARP

- Provides highly available, reliable storage for files
- Guarantees atomic file operations in spite of concurrency and failure

- Primary copy replication
  - Master server authoritative
  - Replicas – backup servers
  - Updates are sent to “enough” replicas to guarantee fail-safe behavior
- Log structured updates

Using logs for throughput



- Disks are partitioned into tracks, sectors and cylinders
- Writing a file might involve writing blocks in different tracks (slow because of seeks)
- Log structure file systems allow user to write sequentially onto disk. Logs contain the transformations. Lazy update.