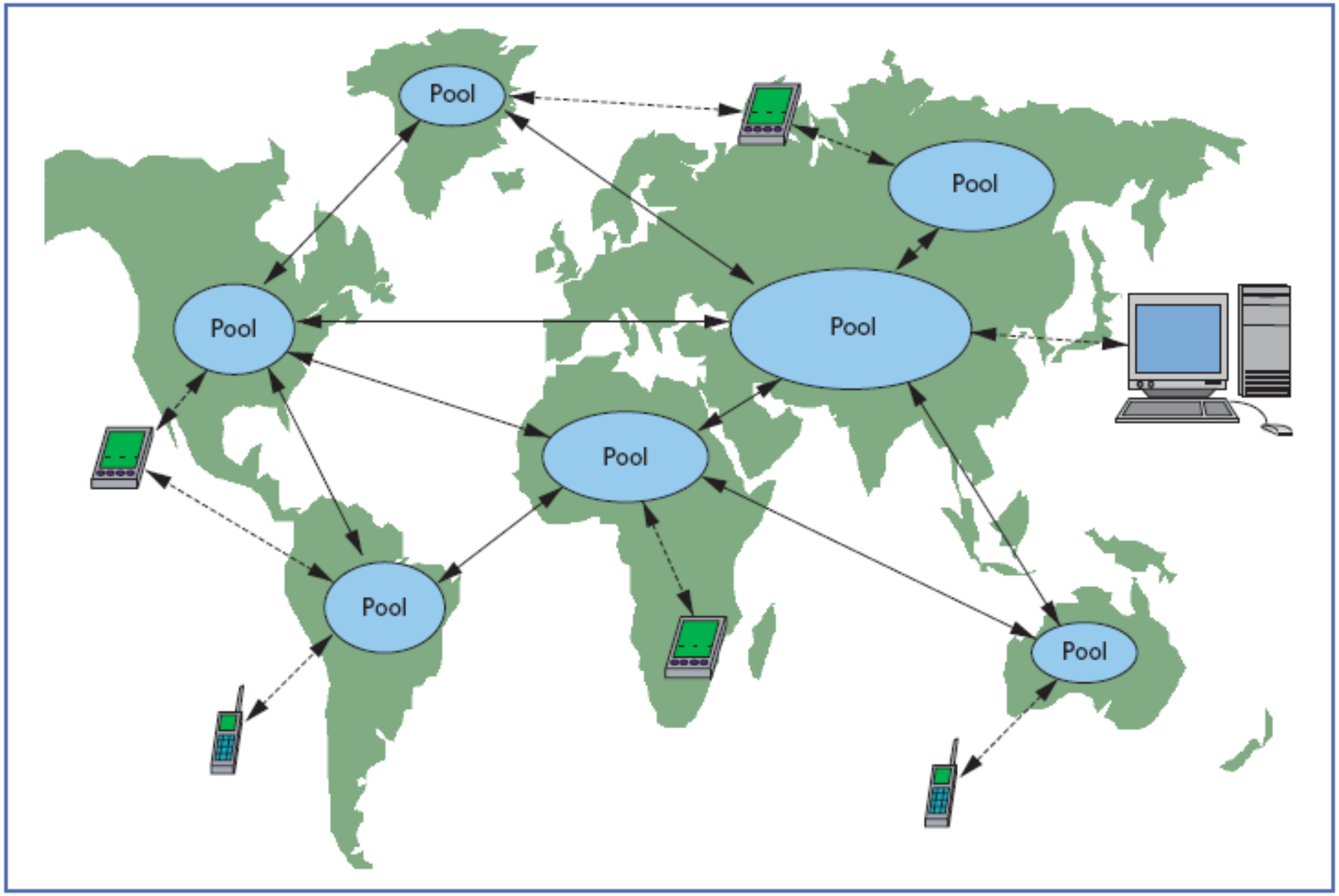
Peer to Peer III

Building a Global Storage Service

Today

- How to build a global data storage utility
  - Pooling resources from millions of devices that
    - Are not trusted
    - Come and go
    - Are independent (no centralized management)
- Target
  - 10^{10} users x 10^{4} files of 10^{4} bytes each = 10^{18} bytes

OceanStore



Goals

- Data lasts forever
- Data is tamper-proof and private
  - subject to user-specified authorization
- Access is generally quick

Issues

- What is the unit of replication
  - File or block?
- Finding a nearby copy
- Updating all copies
  - consistently
- Access control
- Integrity
- Fault tolerance

OceanStore

- **Infrastructure**
  - lots of administrative domains
  - servers trusted in aggregate, but not individually
    - arbitrary passive failures (crashes)
    - arbitrary active failures
      - really smart and malicious people and computers out there …
      - all communication is subject to eavesdropping and disruption

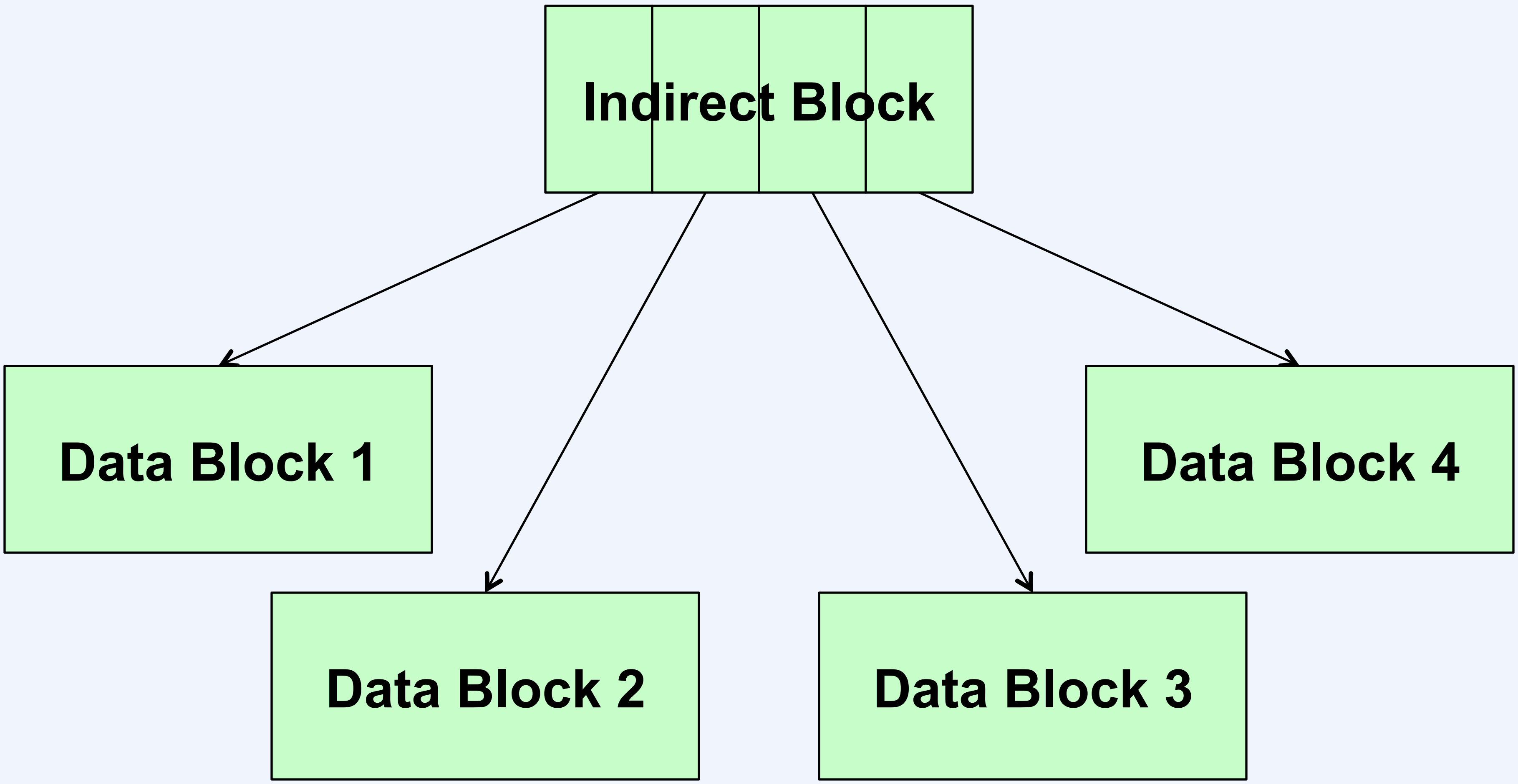
OceanStore

- Designed and developed at Berkeley, early 2000's
- 5 Key Techniques
  - End-to-end encryption, *self-certifying* data
  - Tapestry self-organizing routing infrastructure
  - Erasure coding for durability (m-of-n)
  - Byzantine update commitment
  - Dynamic replica management
- Great use of a DOLR (Tapestry)
- *Teaser* for many other techniques we will see throughout the semester

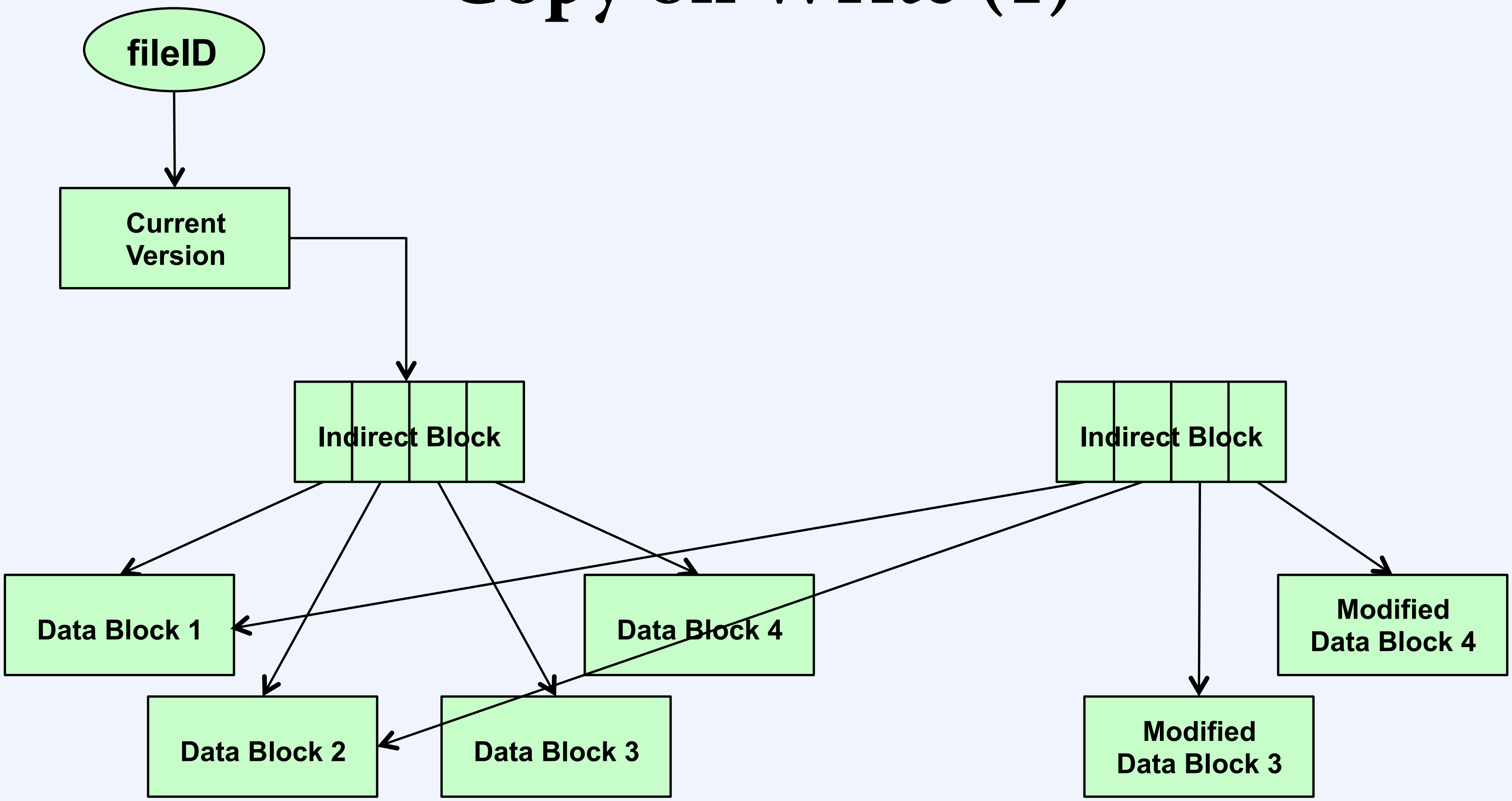
Caveats

- It's far more complex than PuddleStore!
- This presentation is based on a number of OceanStore papers
  - not everything is totally clear

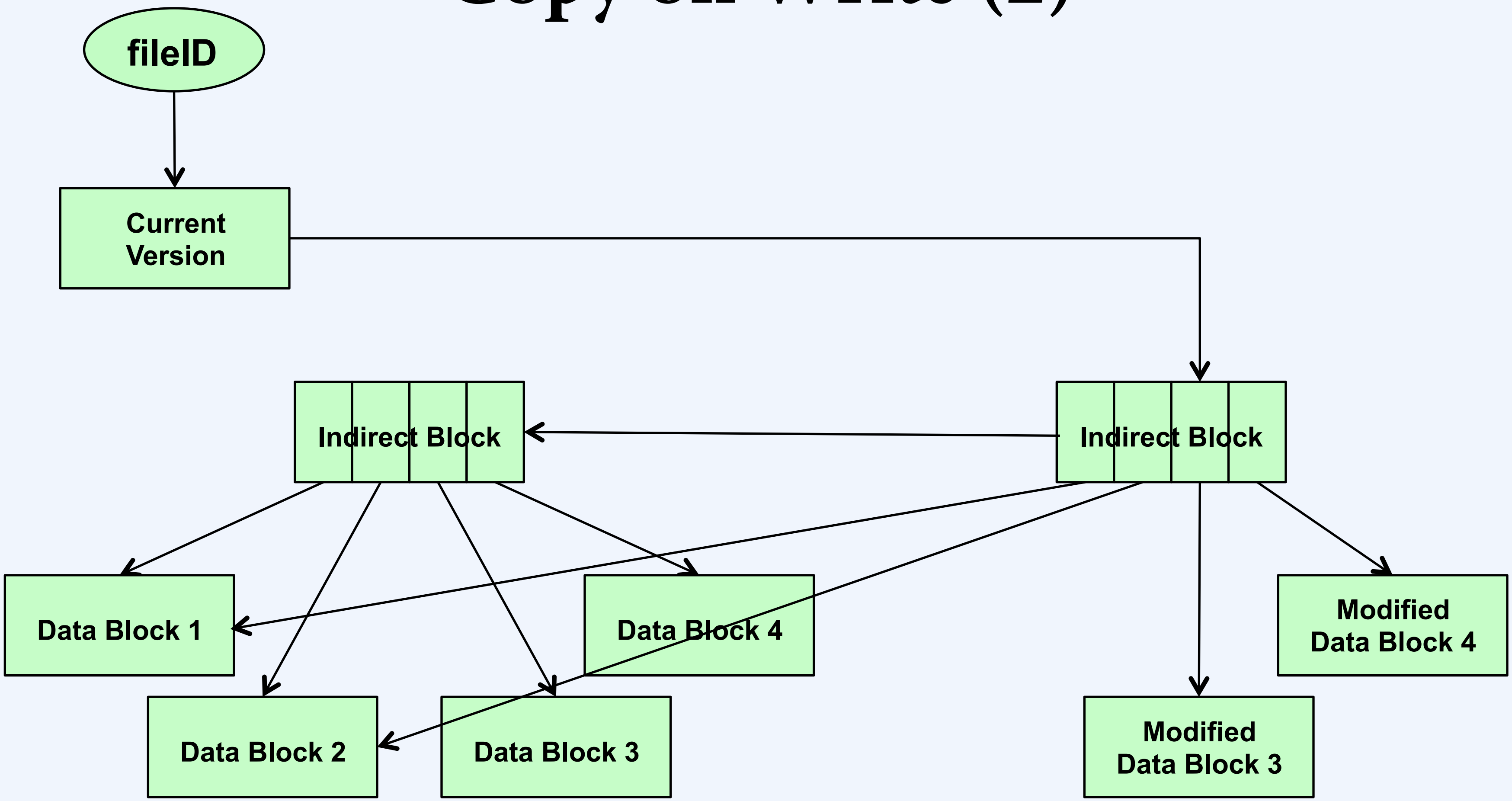
File Format


Indirect Block
Data Block 1
Data Block 2
Data Block 3
Data Block 4

Copy on Write (1)



Copy on Write (2)



Questions

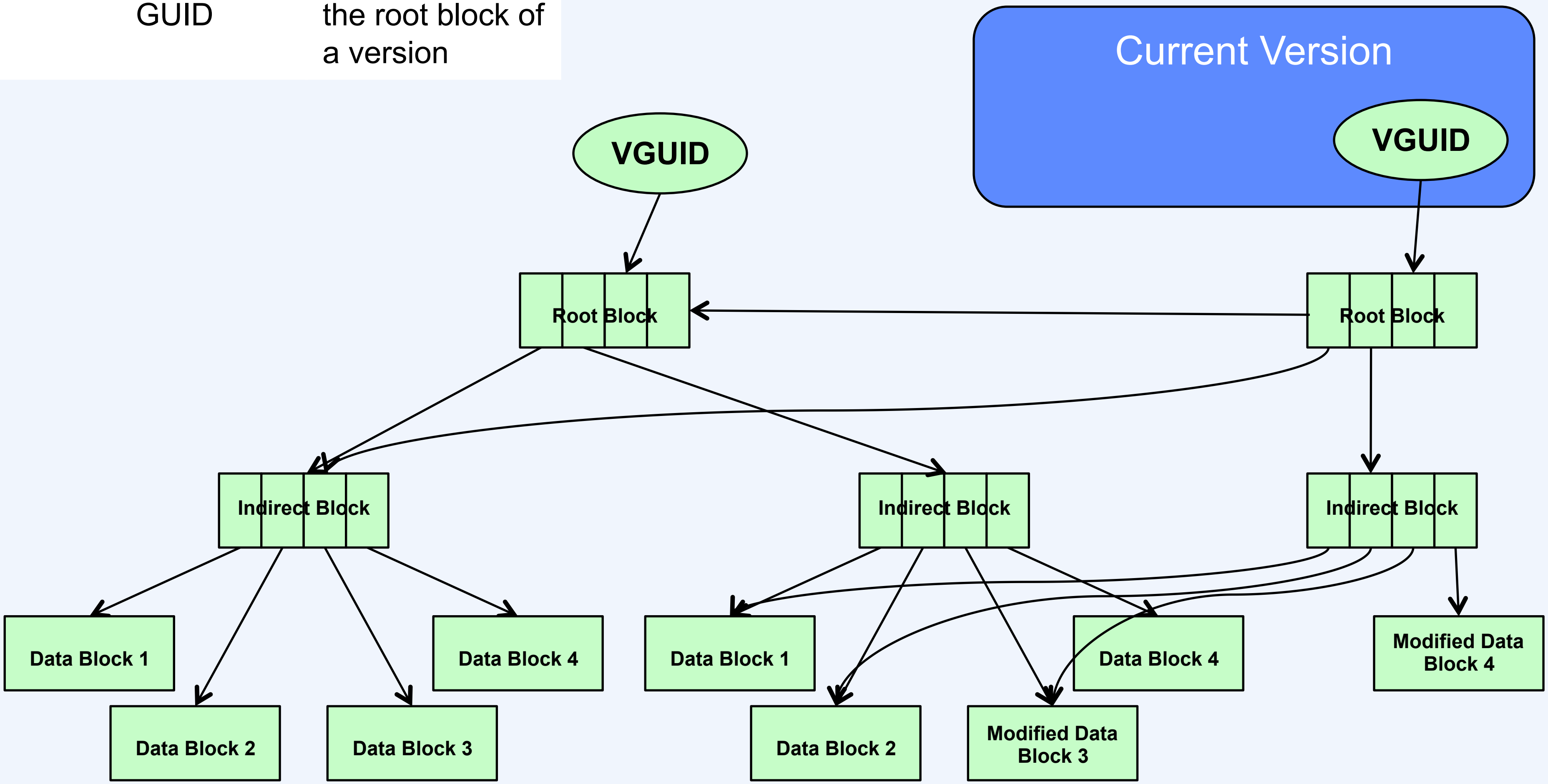
- Where do the blocks go?
- How are they referenced?
- What if we can't trust the computers that hold them?
- What if there are communication failures?

The Answer

- Tapestry
  - blocks are identified by *secure hashes of their contents*
  - contents may be encrypted if necessary
  - multiple copies may be published for redundancy
- Self-certifying blocks
  - What happens if the the block changes?

OceanStore File

BGUID	block GUID	secure hash of a block of data
VGUID	version GUID	secure hash of the root block of a version



Benefits of Replication

- Example with 2 replicas
  - N = total number of servers = 10^6
  - M = number of servers expected to be down = 10^5 (10%)
- What is the probability that at least one copy of a replicated block is available?

$$\sum_{i=0}^{1} \frac{\binom{M}{i}\binom{N-M}{2-i}}{\binom{N}{2}}$$

- = 0.99

Erasure Codes

- **Divide an object into *m* equal-size fragments and code them as *n* equal-size fragments, *n>m***
  - ***m/n = rate of encoding = r***
  - ***1/r* = multiplicative storage increase**
- **Original object can be reconstructed from any m out of n fragments**
  - **handles *n-m* "erasures"**

Benefits of Erasure Codes

- N = total number of servers = 10^6
- M = number of servers expected to be down = 10^5
- n = 32, m = 16
- What is the probability that at least m fragments of an erasure-coded block is available?

$$\sum_{i=0}^{n-m} \frac{\binom{M}{i}\binom{N-M}{n-i}}{\binom{N}{n}}$$

- = 0.999999998

In Practice

- Original block is stored in Tapestry by its BGUID
- Erasure-coded fragments stored by id *f*(BGUID, *k*), where k is the block number
- Can be replicated at will
- These are called *archival copies*

The Down Side …

- Fetching and assembling the fragments is expensive
  – how can it be made cheaper (on average)?

Caching

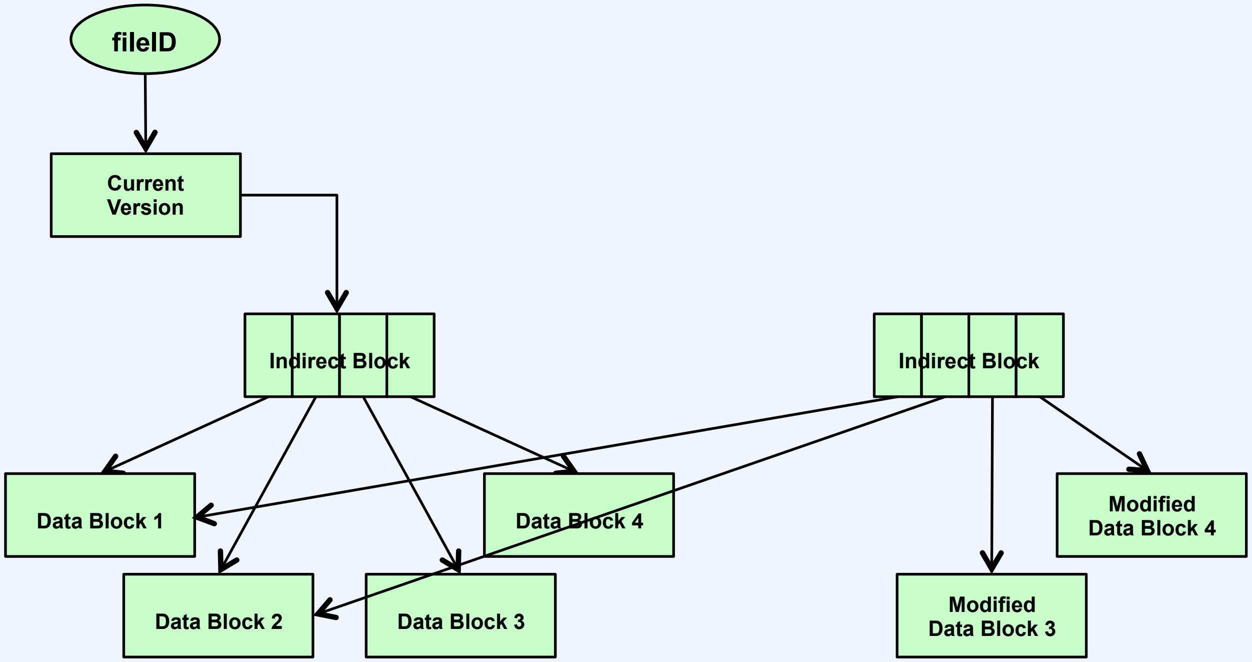
- If you've reconstructed a block
  - publish it in tapestry
  - available for others until you delete it

Where Are We?

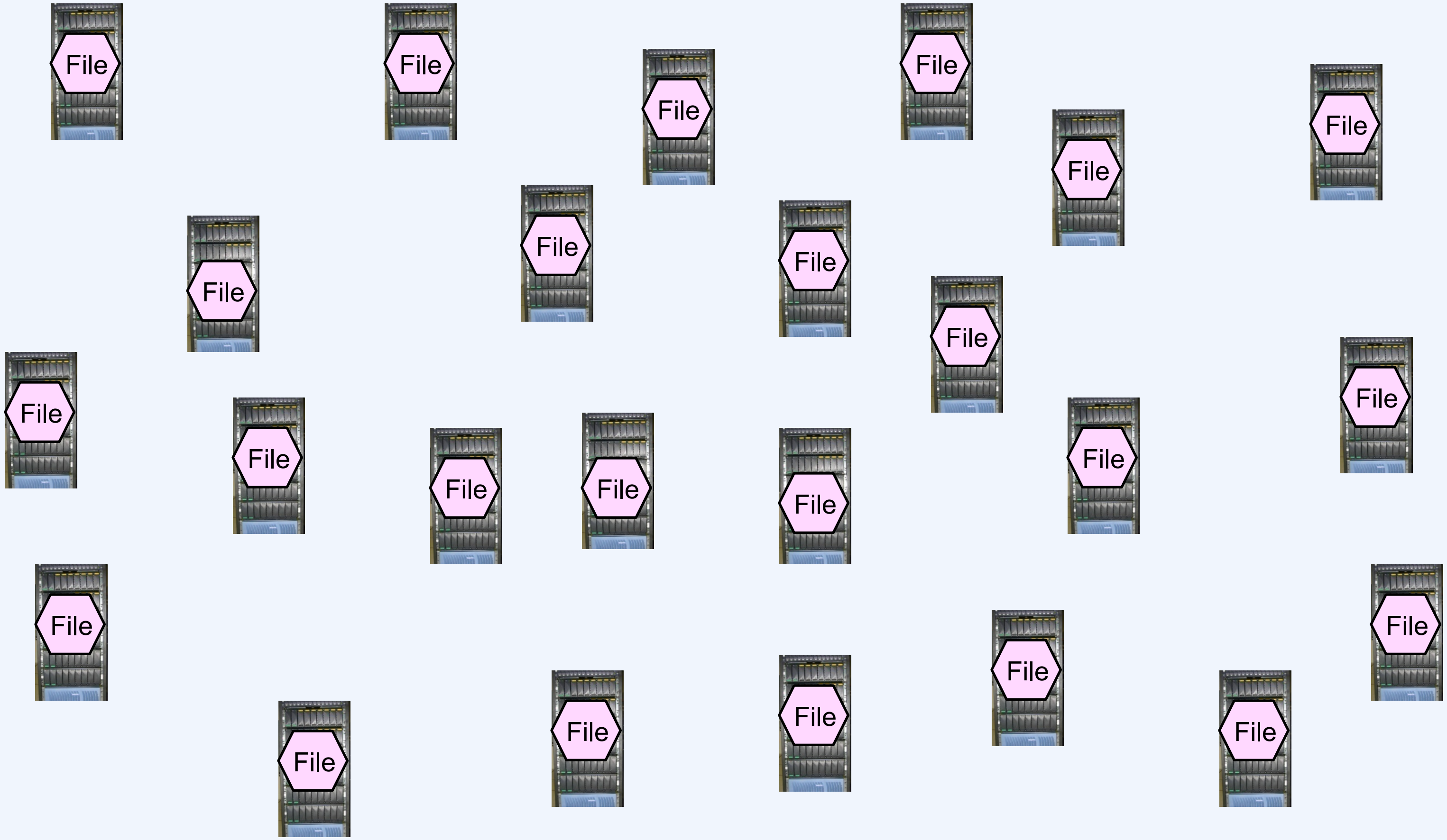
- Blocks are widely replicated using erasure coding
  - fragments form the archival copy of the file
- Caching takes place as file is accessed

How do you find a file

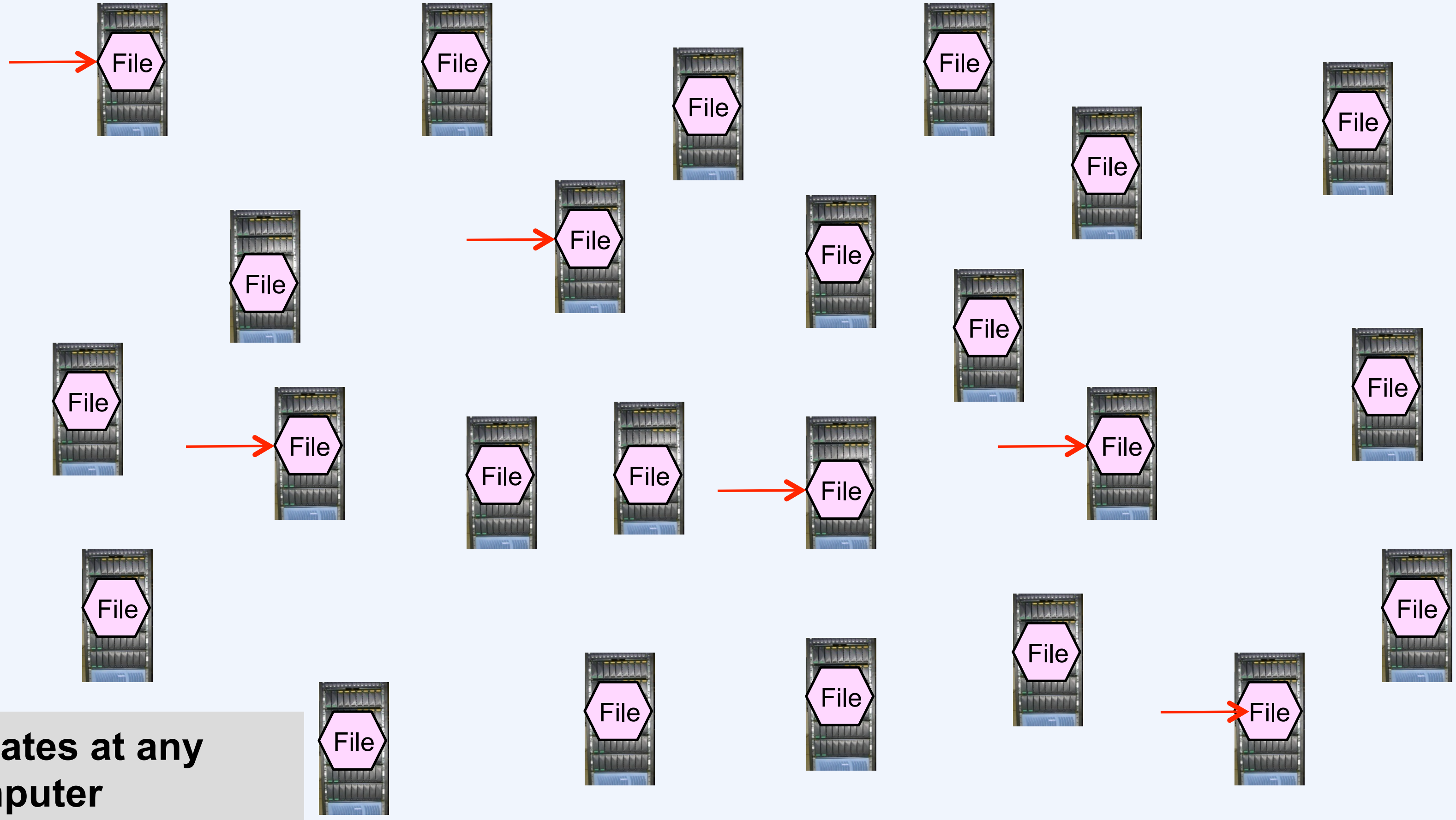
- Need to go from file name to a record
- Actually just the pointer to the latest version
  - Only mutable record!
  - Id is a hash of (file name, owner id)



Distributing a Mutable File



Group Replication



- Updates at any computer
- Collectively determine effective order

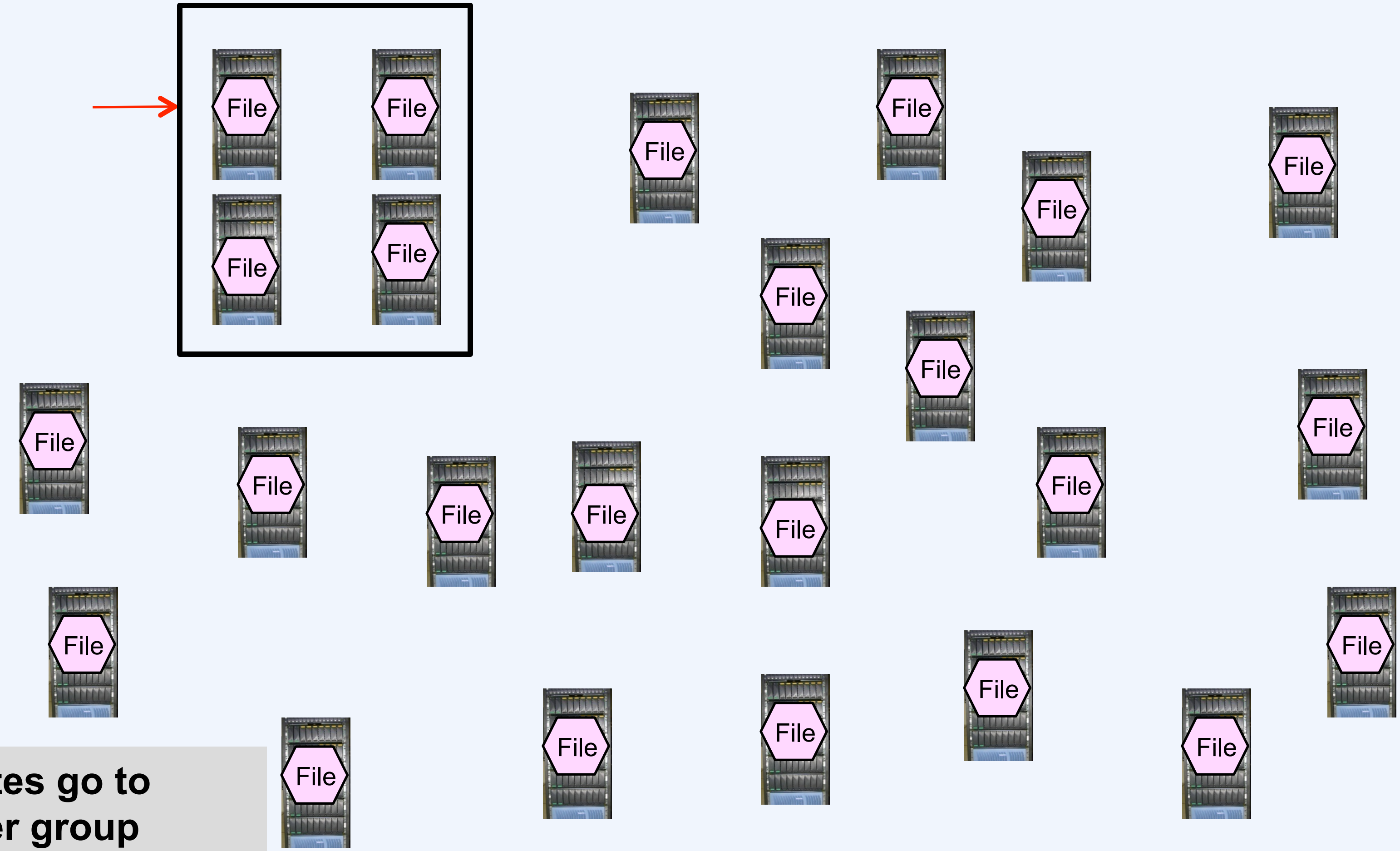
Master Replication

- Updates just at one computer
- It propagates them to the others

Issues

- Should we trust the server holding the primary copy?
  - should it be special?
    - No!

Compromise

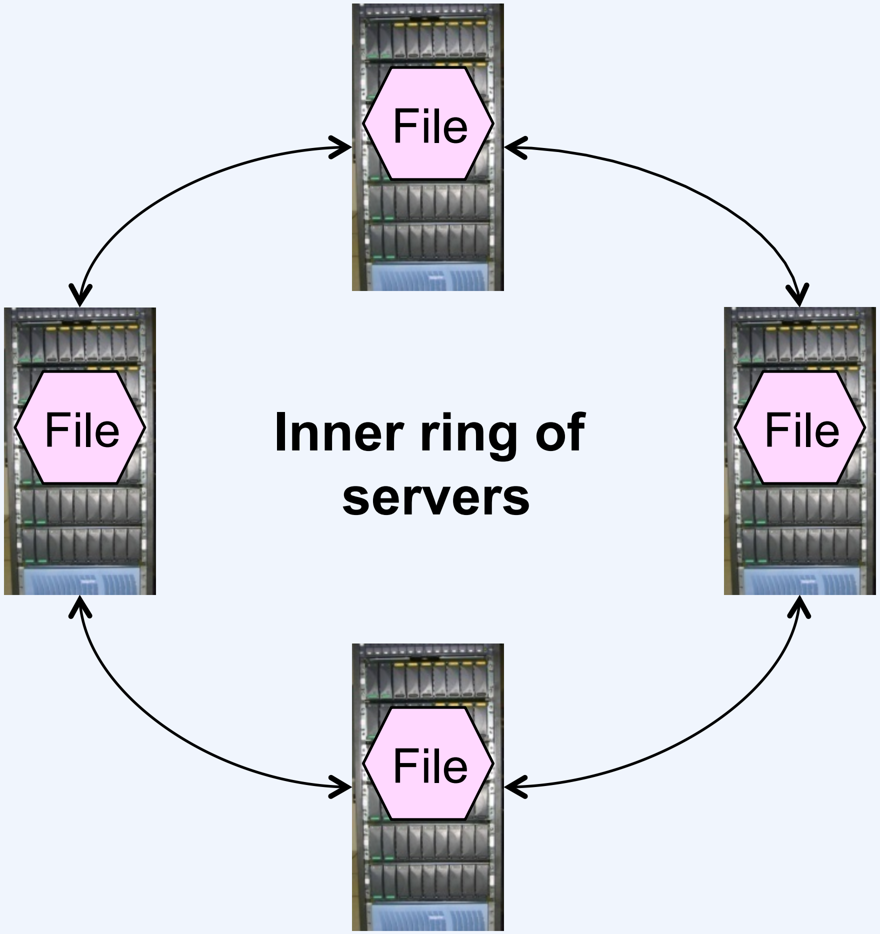


- Updates go to master group
- It propagates them to the others

Inner Ring

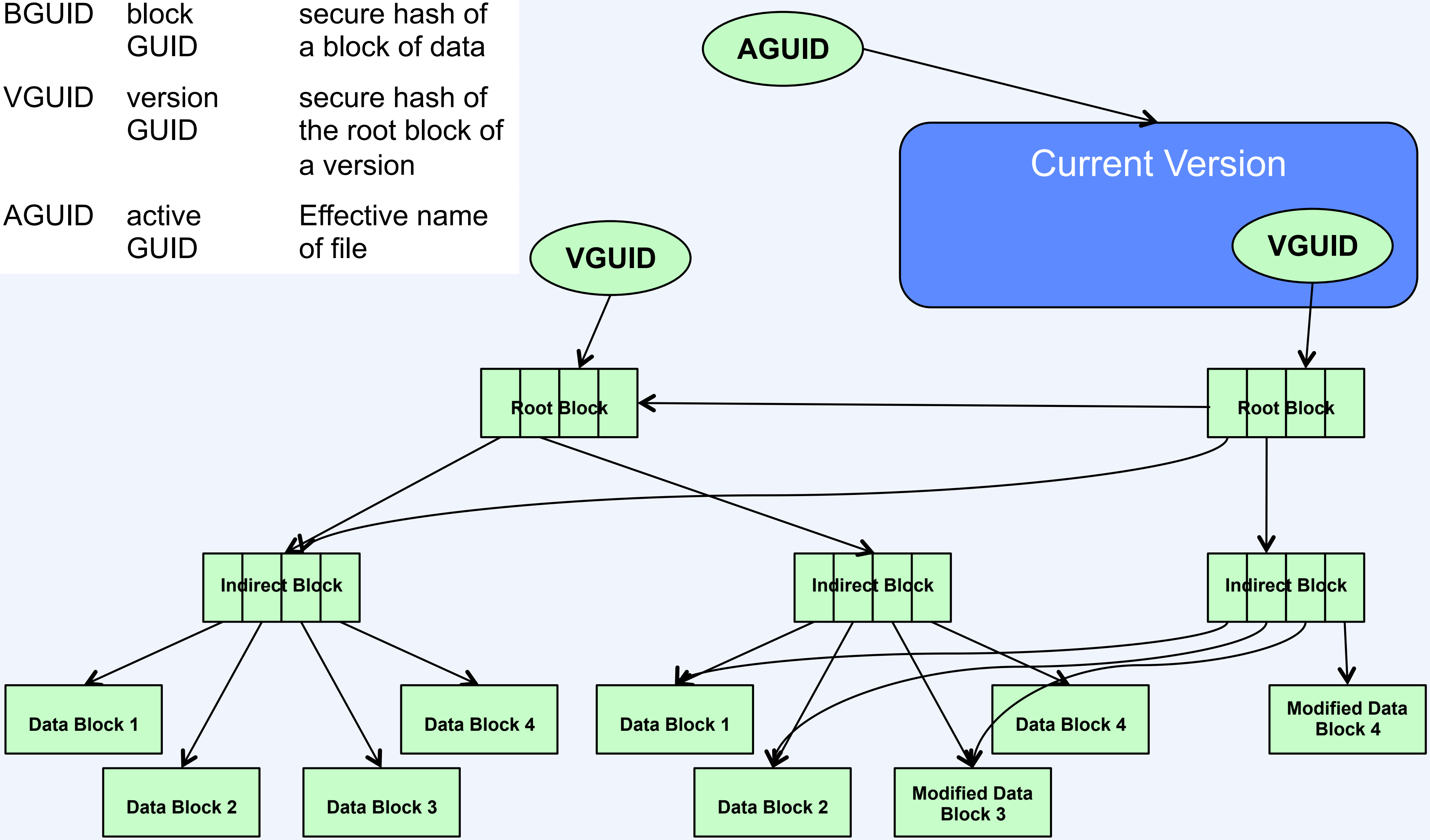
- A collection of servers are responsible for the primary copy
  - called the *inner ring*
- Collectively make updates
  - together they hold the *primary replica*
- File is identified by its *active GUID (AGUID)*
  - secure hash of application-specific name and owner's public key
- Each inner-ring server publishes the AGUID in Tapestry
  - each holds a copy of the current VGUID

Replicated Primary Replica

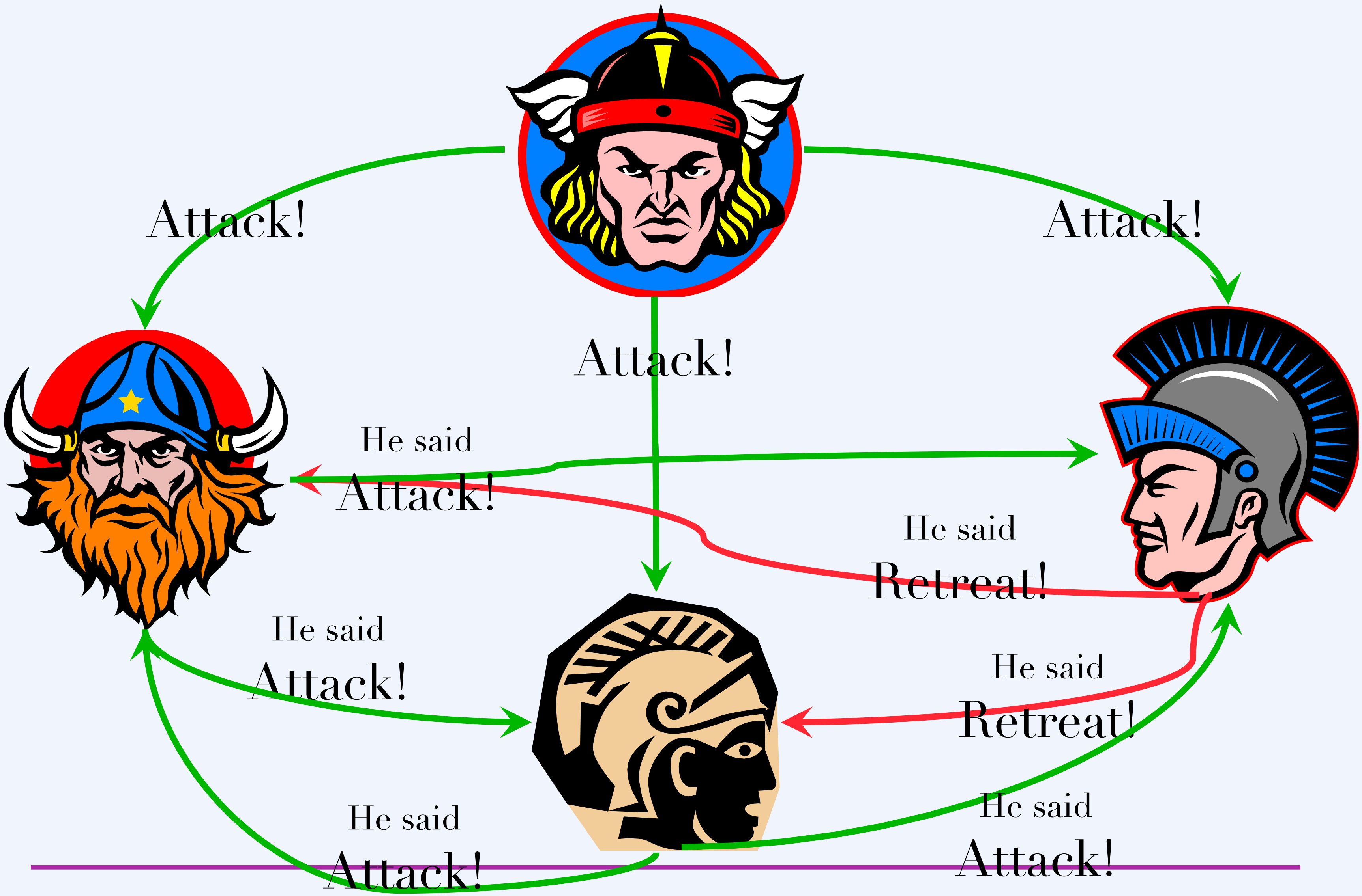


OceanStore File

BGUID	block GUID	secure hash of a block of data
VGUID	version GUID	secure hash of the root block of a version
AGUID	active GUID	Effective name of file

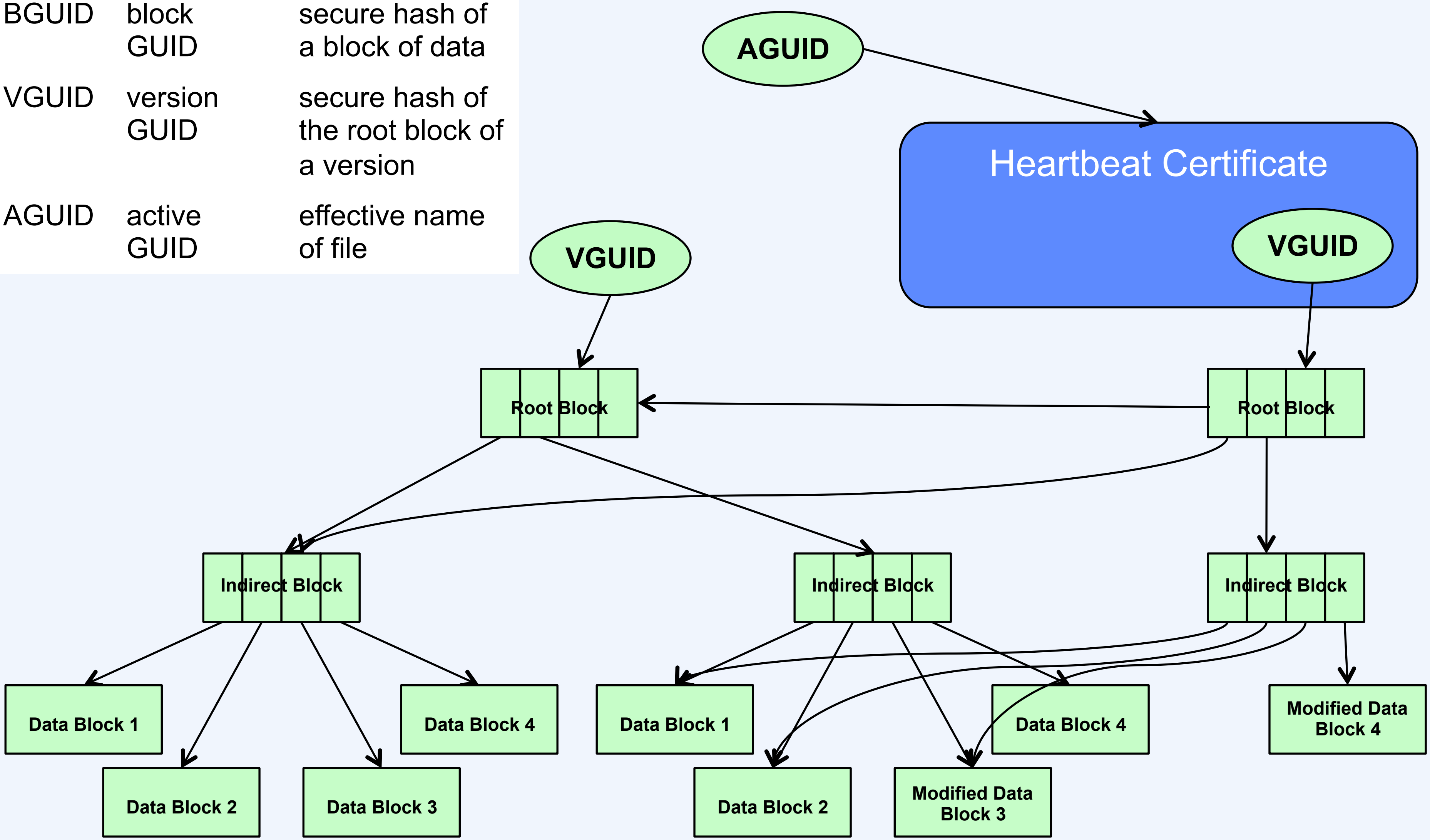


Byzantine Generals Problem



OceanStore File

BGUID	block GUID	secure hash of a block of data
VGUID	version GUID	secure hash of the root block of a version
AGUID	active GUID	effective name of file



Almost There …

- What about:
  - getting notified about file updates?
  - finding the AGUID in the first place?

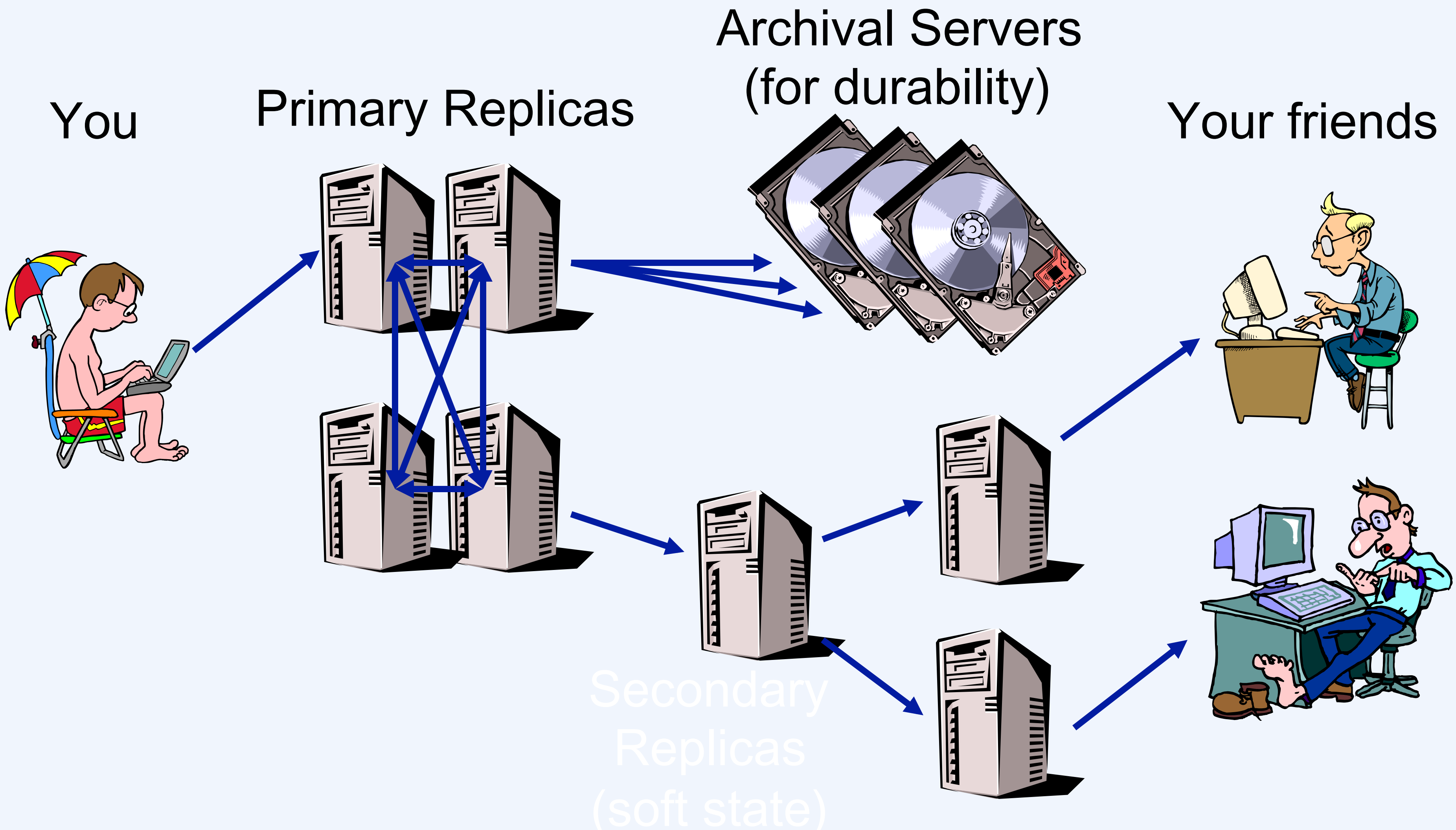
Secondary Replicas

- May be stored on any server
- Hold copies of heartbeat
  - new copies pushed to them by inner-ring servers
    - new secondary replicas find and link to existing ones via Tapestry
    - forms tree of replicas

Where Are We?

- File update
  - clients send update requests to primary replica
  - inner-ring machines come to agreement on update order
    - commit changes to local copies
    - propagate heartbeats to secondary replicas
    - fragment new blocks and add to archival servers using erasure code

Putting it all together: the Path of a Write



Where Are We?

- File read
  - given a BGUID, block is found via Tapestry
    - first look up BGUID
    - if not found compute fragment GUIDs and look them up
      - verify results
      - combine into block
  - to find latest version of file, contact primary replica (or, perhaps, secondary replica)

What Else?

- Not described here
  - all file blocks are encrypted by clients
  - directories
  - access control
- Not implemented in Pond
  - automatic repair
    - periodic verification of data
    - periodic copying of data to new disks
  - introspection
    - adaptive system management
- Not clear
  - management of inner rings

Performance

	LAN		WAN		Predominant operations in benchmark
Phase	*Linux NFS*	*Pond*	*Linux NFS*	*Pond*	
1	0.0	1.9	0.9	2.8	Read and write
2	0.3	11.0	9.4	16.8	Read and write
3	1.1	1.8	8.3	1.8	Read [status only]
4	0.5	1.5	6.9	1.5	Read
5	2.6	21.0	21.5	32.0	Read and write
Total	4.5	37.2	47.0	54.9	