Concurrent Deletion in a Distributed Content-Addressable Storage System with Global Deduplication

Przemysław Strzelczak, Elżbieta Adamczyk,
Urszula Herman-Iżycka, Jakub Sakowicz,
Łukasz Ślusarczyk, Jarosław Wrona,
Cezary Dubnicki

9LivesData, LLC

Who we are

- **9LivesData**
  - R&D company based in Warsaw, Poland
  - 50+ scientists and software engineers
  - designers/coders of HYDRAstor backend for NEC
- **HYDRAstor**
  - scalable, content-addressable backup storage
  - global dedup, self-healing
  - owned by NEC, on sale in the USA and Japan
  - started by 9LivesData founder in Princeton, NJ
  - **fastest and largest** dedup backup system*

* Curtis W.Preston performance comparison – 10/28/2010

Core features of secondary storage systems

- Large capacity (deduplication)
- High performance
- High reliability
- **Data deletion**

HYDRAstor

Agenda

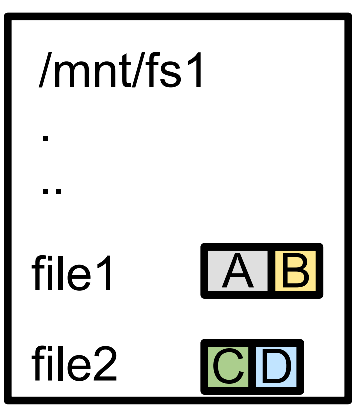
- Requirements and challenges
- Deletion algorithm for CAS systems
  - read-only period
  - centralized system
- Extension to support writes with deduplication
- Extension to distributed systems
- Implementation in HYDRAstor

HYDRAstor

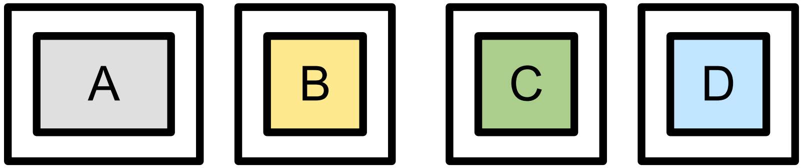
Key requirements

- Continuous system availability
  - no read-only period
  - failure tolerance
- Deduplication enabled during deletion
- Low impact on user operations
  - backup
  - restore
  - replication
- Deletion impact independent from system size

Summary of data model

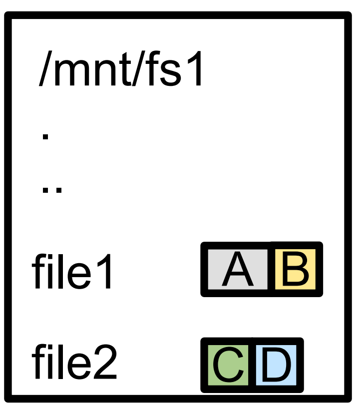


Files chunked into variable sized **blocks**

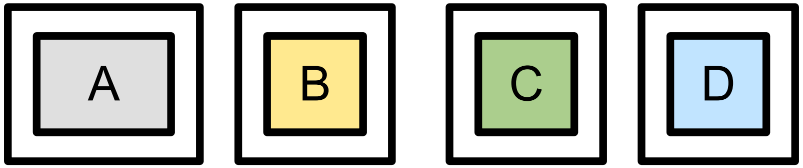


HYDRAstor

Summary of data model



Regular blocks
immutable
content-addressable



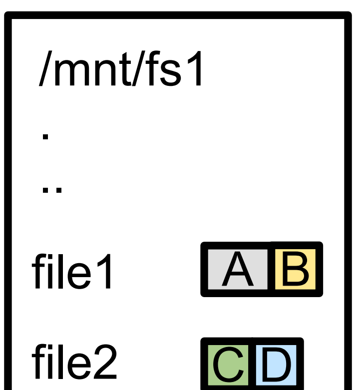
New blocks written **sequentially**

Files chunked into variable sized **blocks**

HYDRAstor

Summary of data model

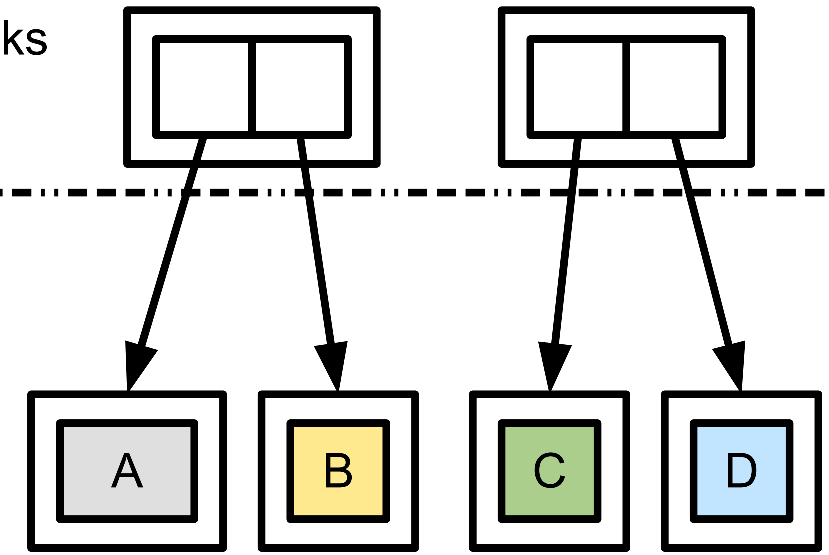
Regular blocks can contain **pointers** to other blocks



Regular blocks
immutable
content-addressable

New blocks written **sequentially**

Files chunked into variable sized **blocks**

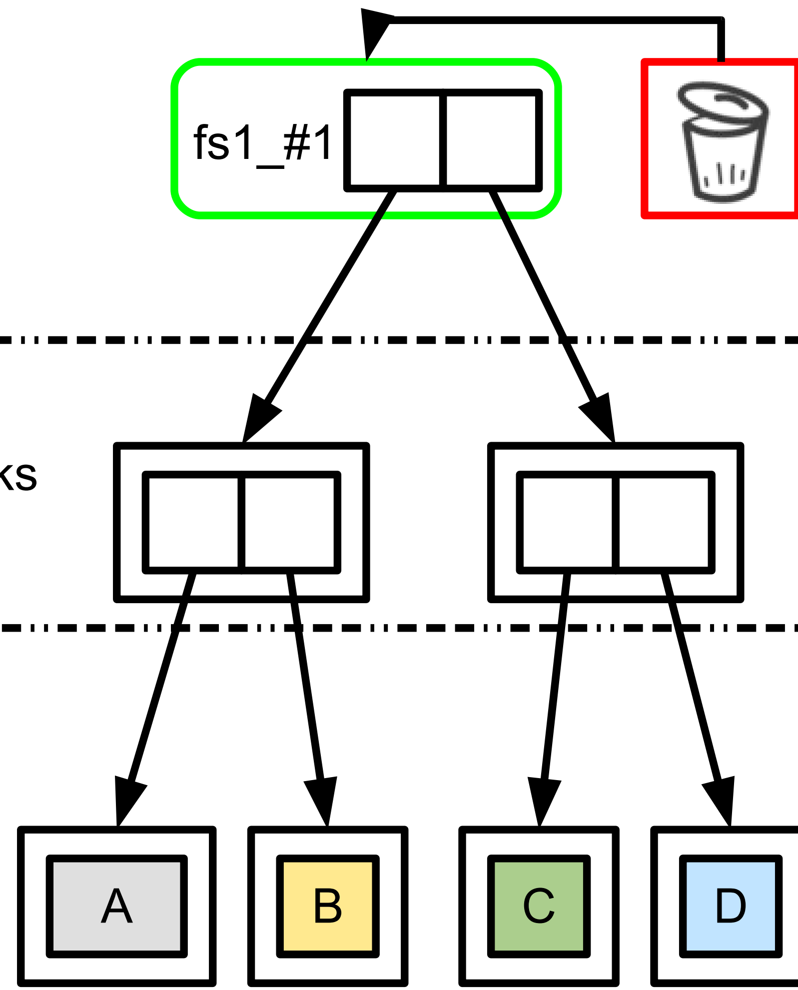


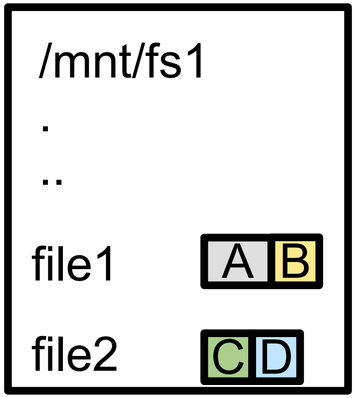
Summary of data model

Filesystem backups as **trees** of blocks built **bottom-up**

Named blocks:
- read using user provided **key**
- **retention roots** – tree preservation
- **deletion roots** – tree deletion

Regular blocks can contain **pointers** to other blocks



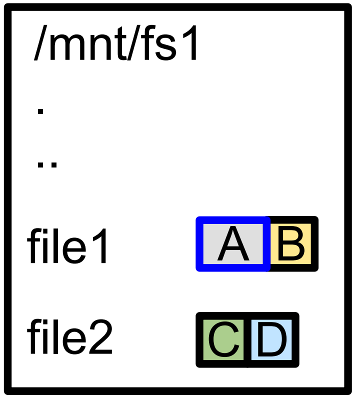


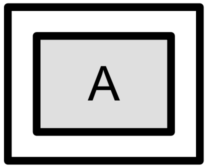
Regular blocks
- immutable
- content-addressable

New blocks written **sequentially**

Files chunked into variable sized **blocks**

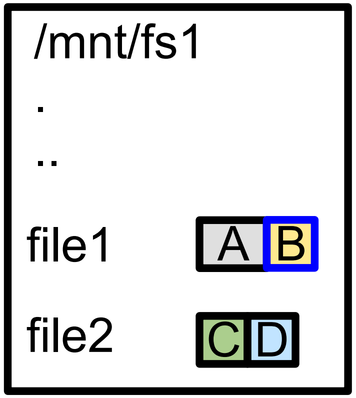
Data model in CAS storage system

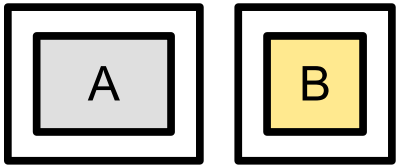




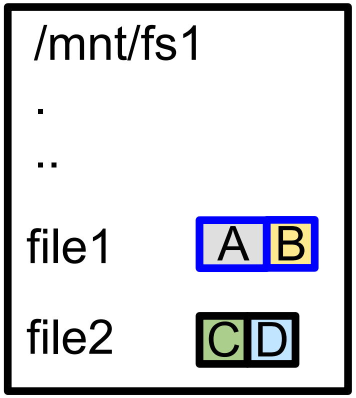
regular data block

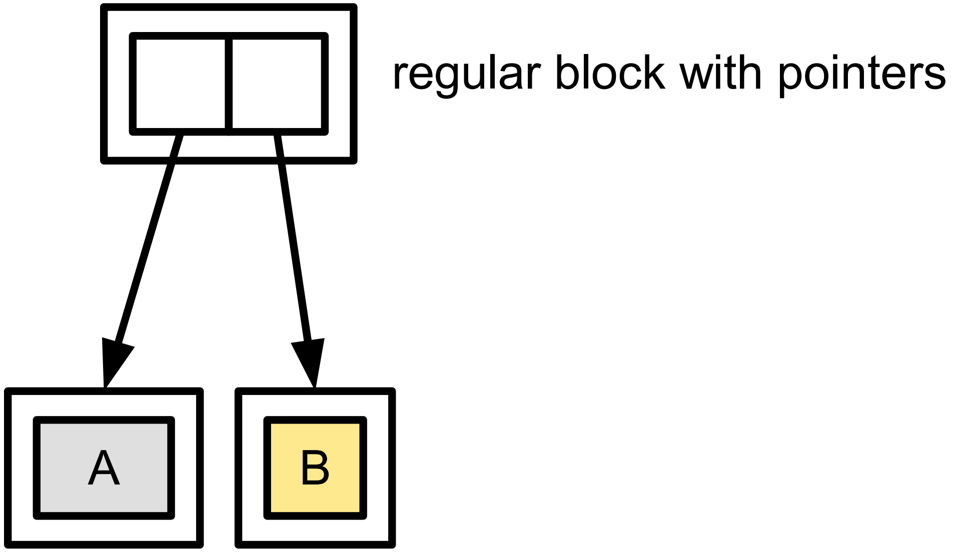
Data model in CAS storage system



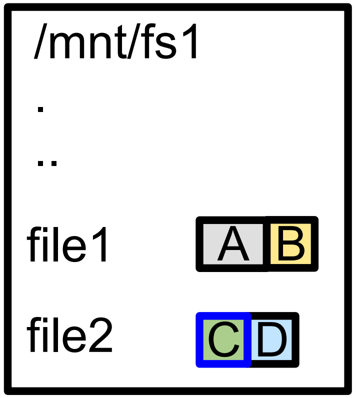


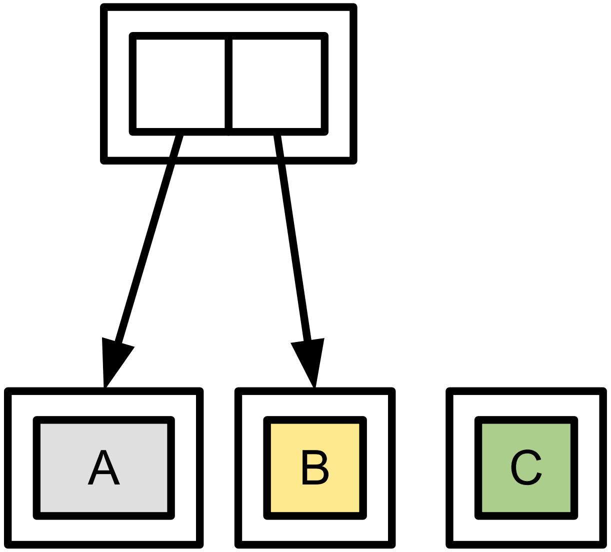
Data model in CAS storage system



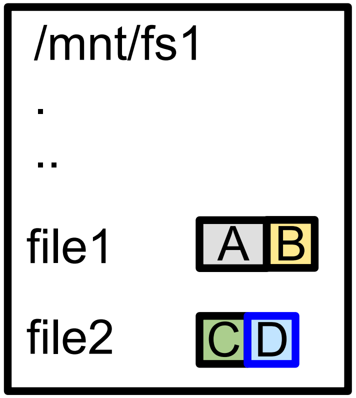


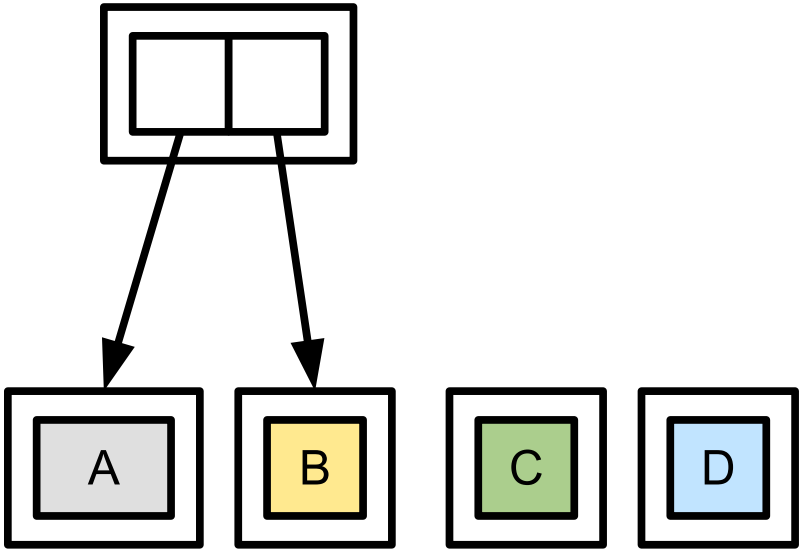
Data model in CAS storage system





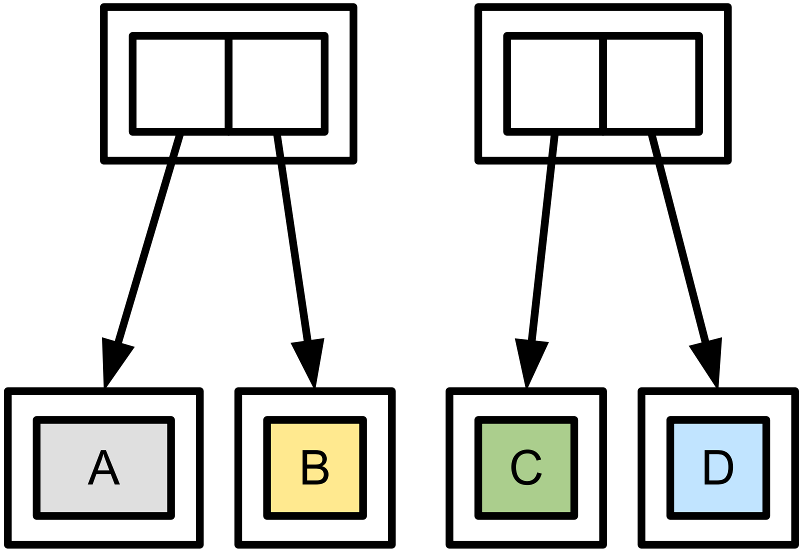
Data model in CAS storage system

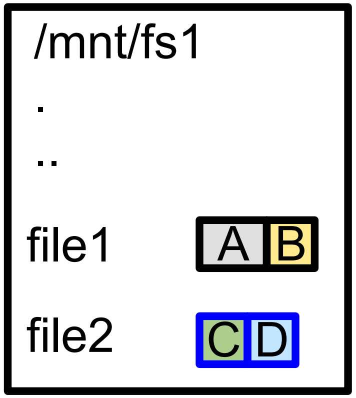




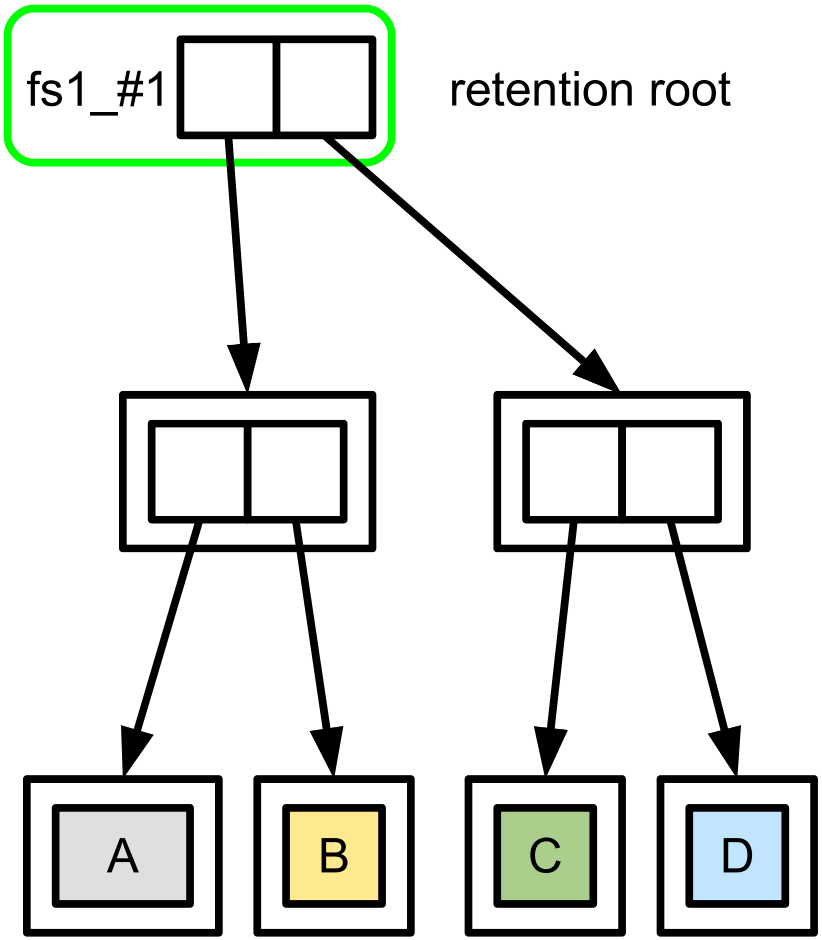
HYDRAstor

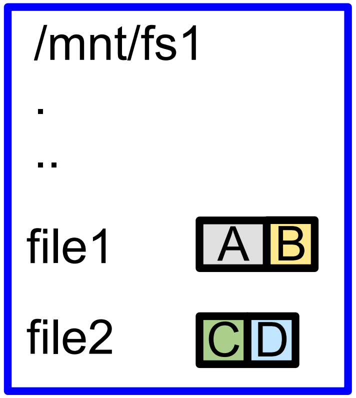
Data model in CAS storage system



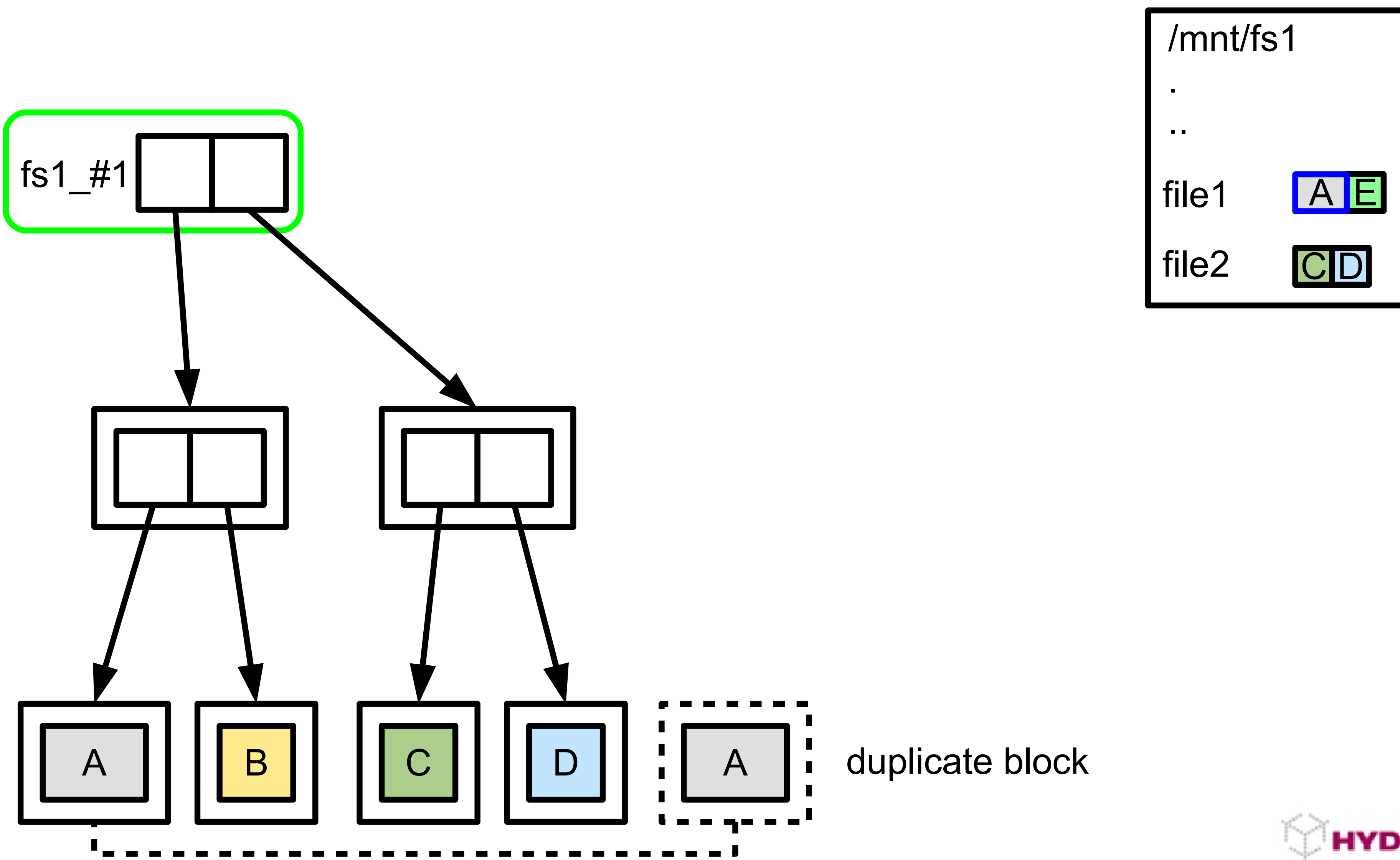


Data model in CAS storage system



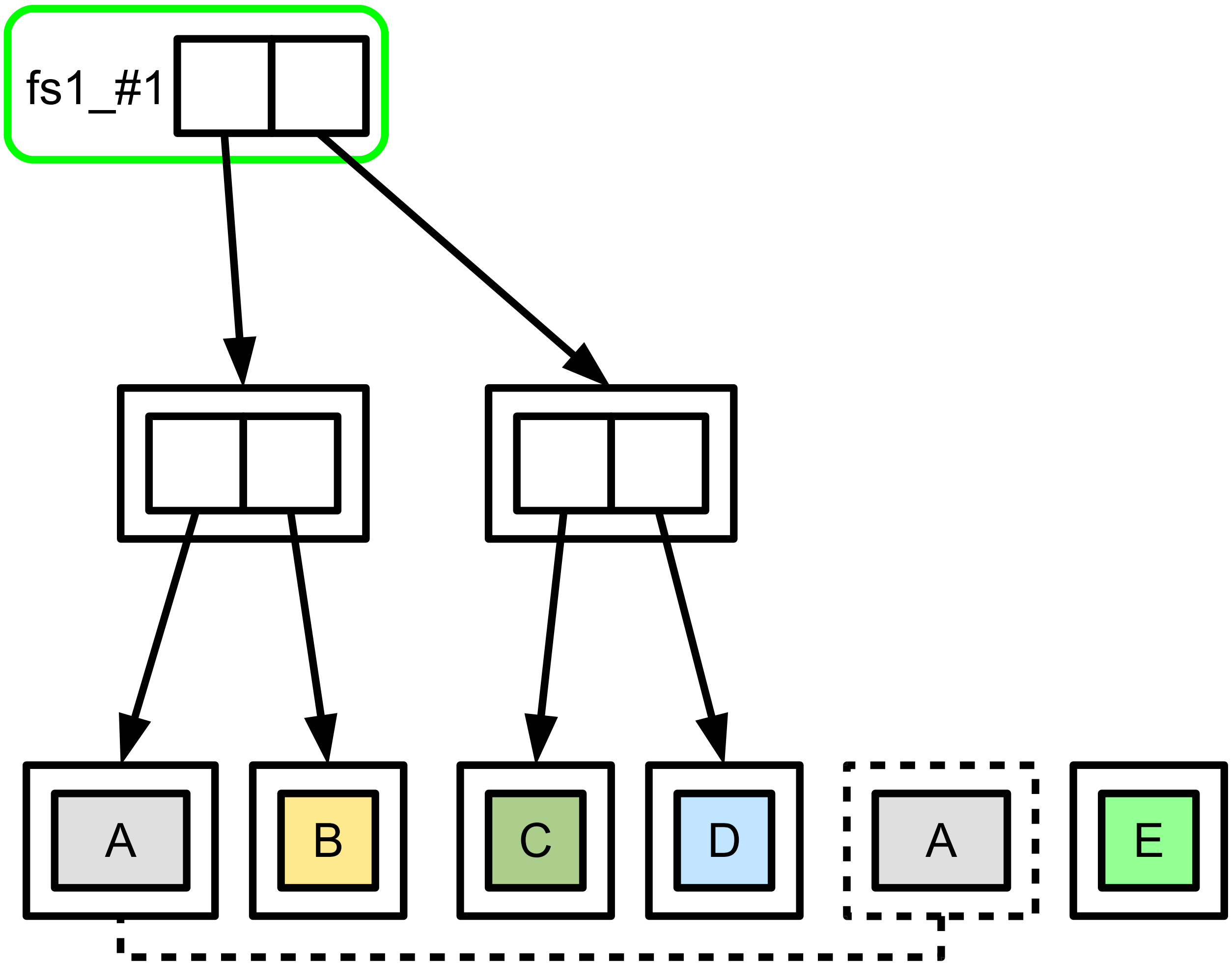


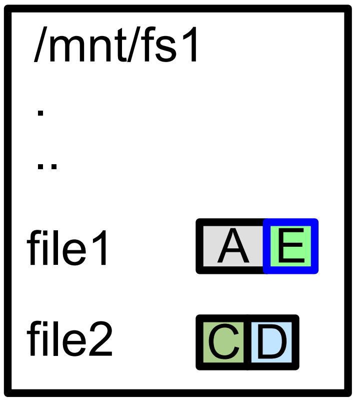
Data model in CAS storage system



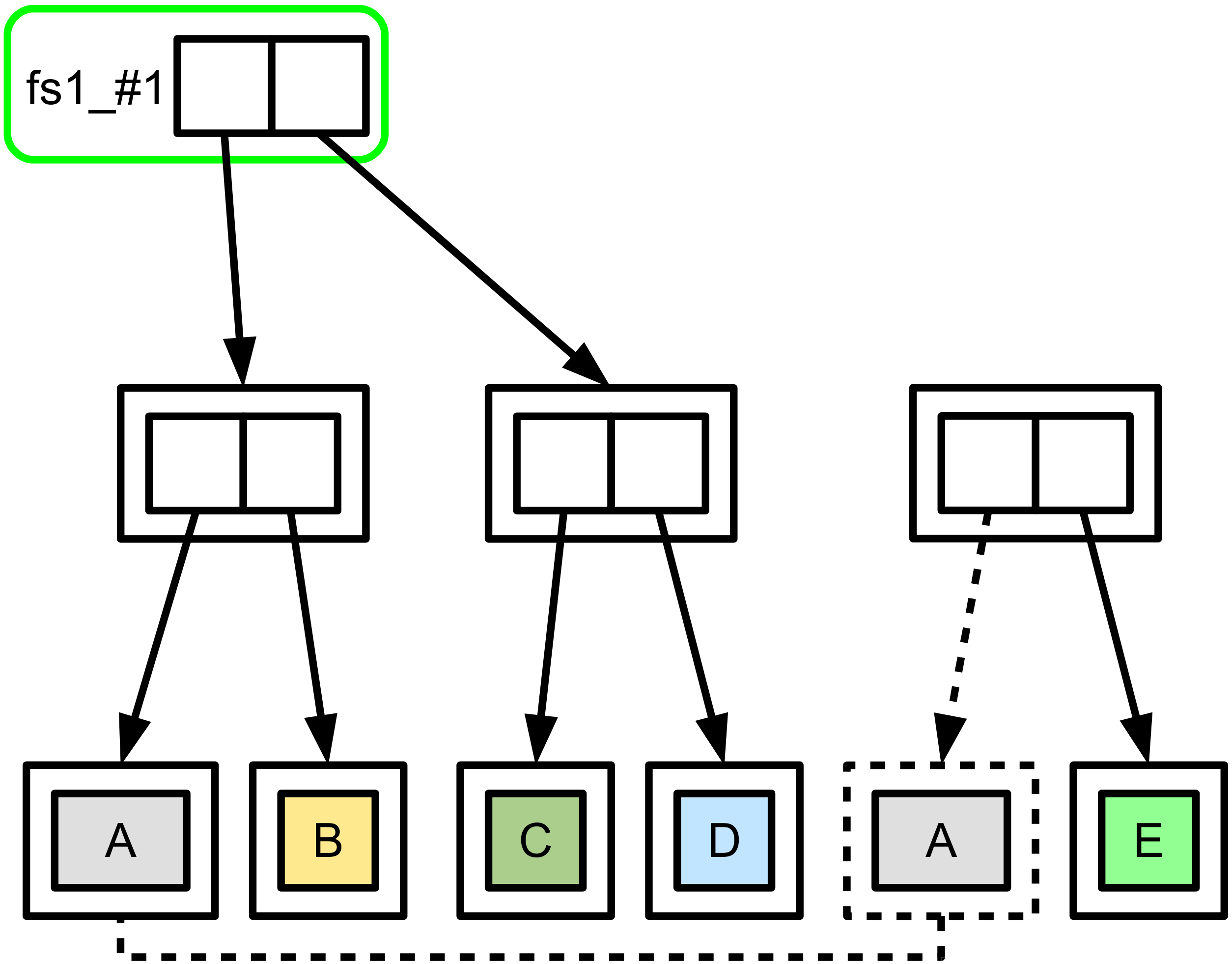
HYDRAstor

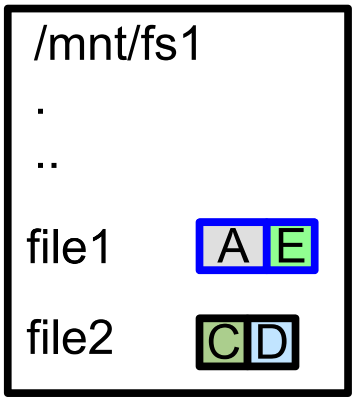
Data model in CAS storage system



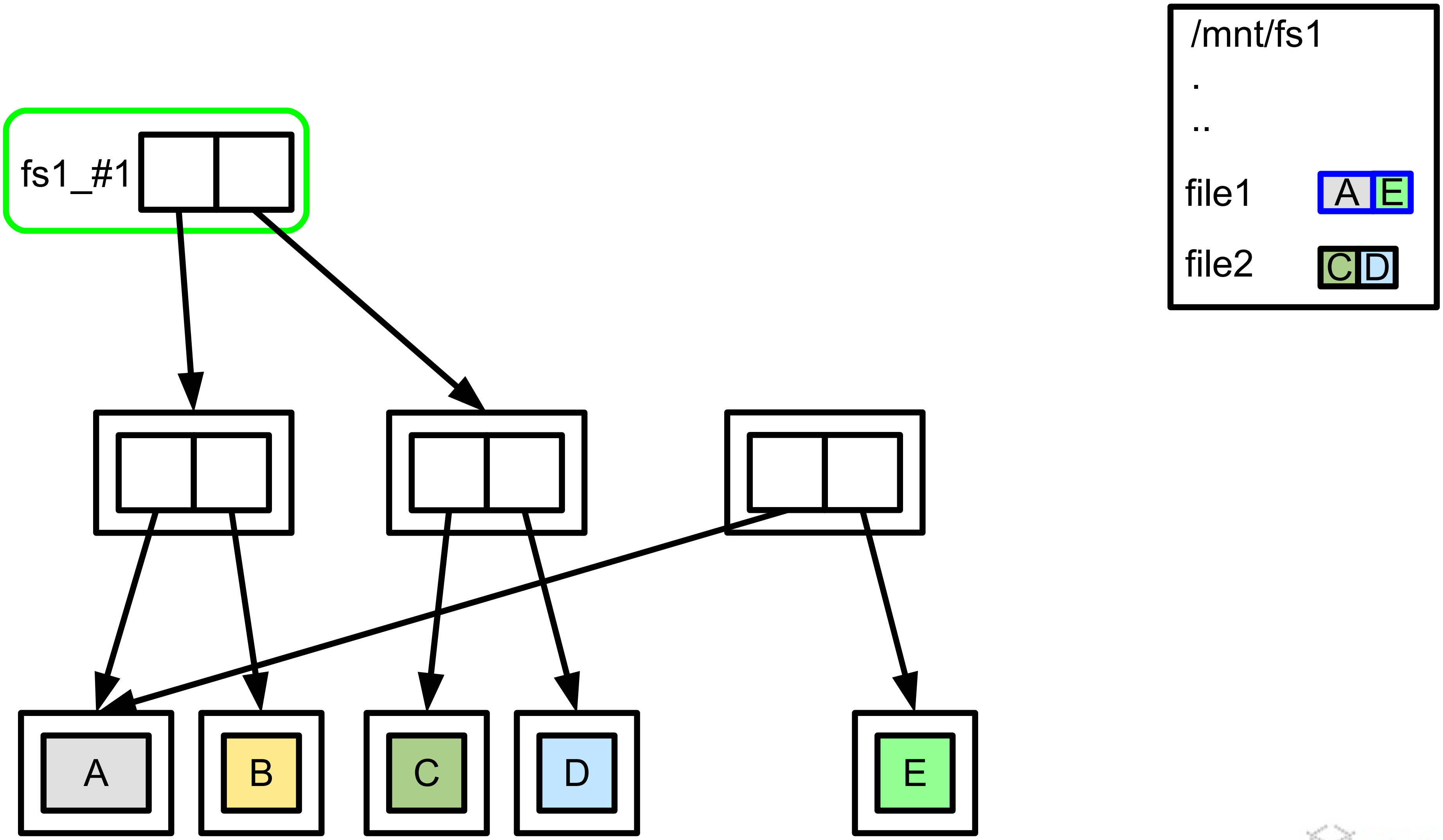


Data model in CAS storage system

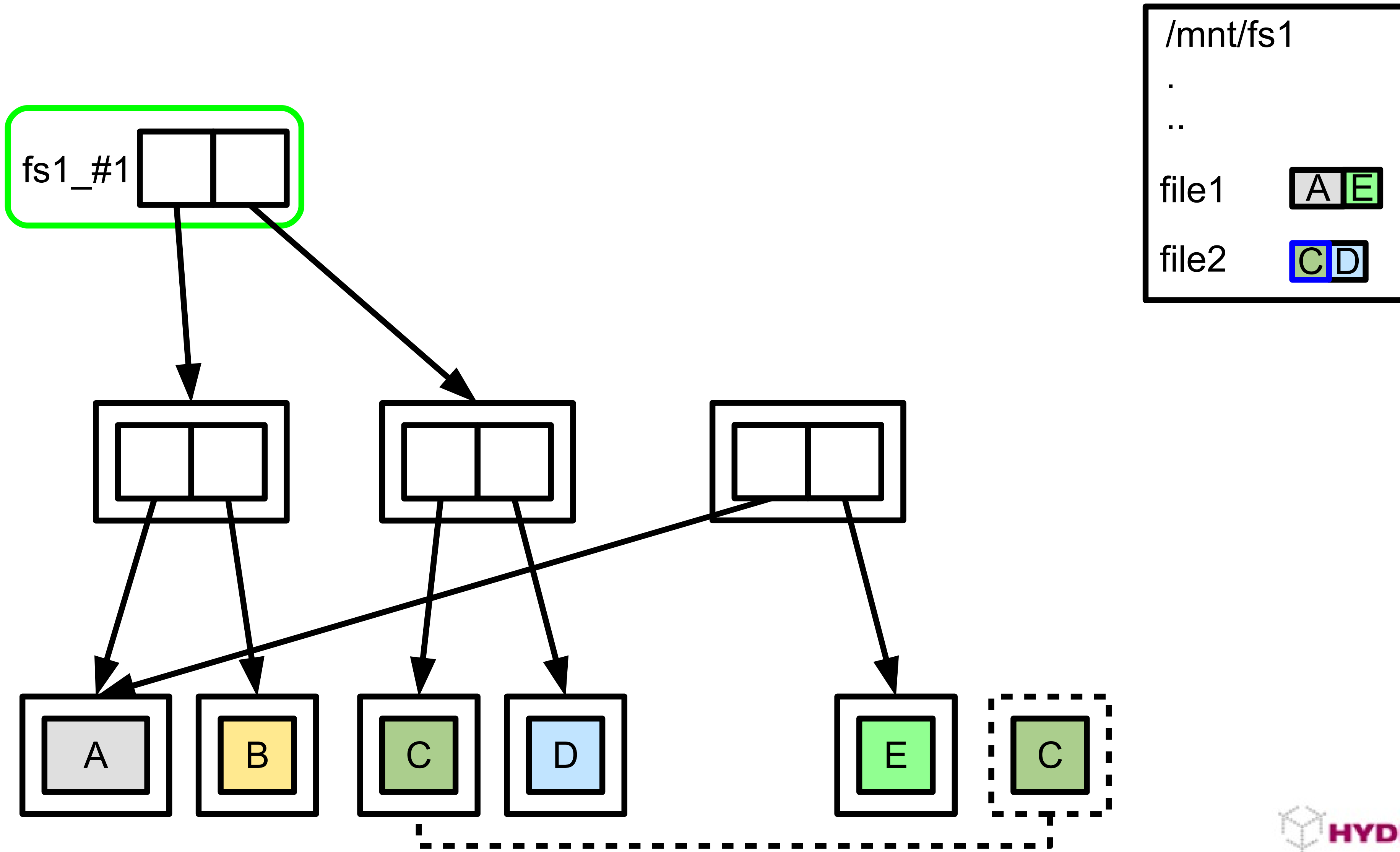




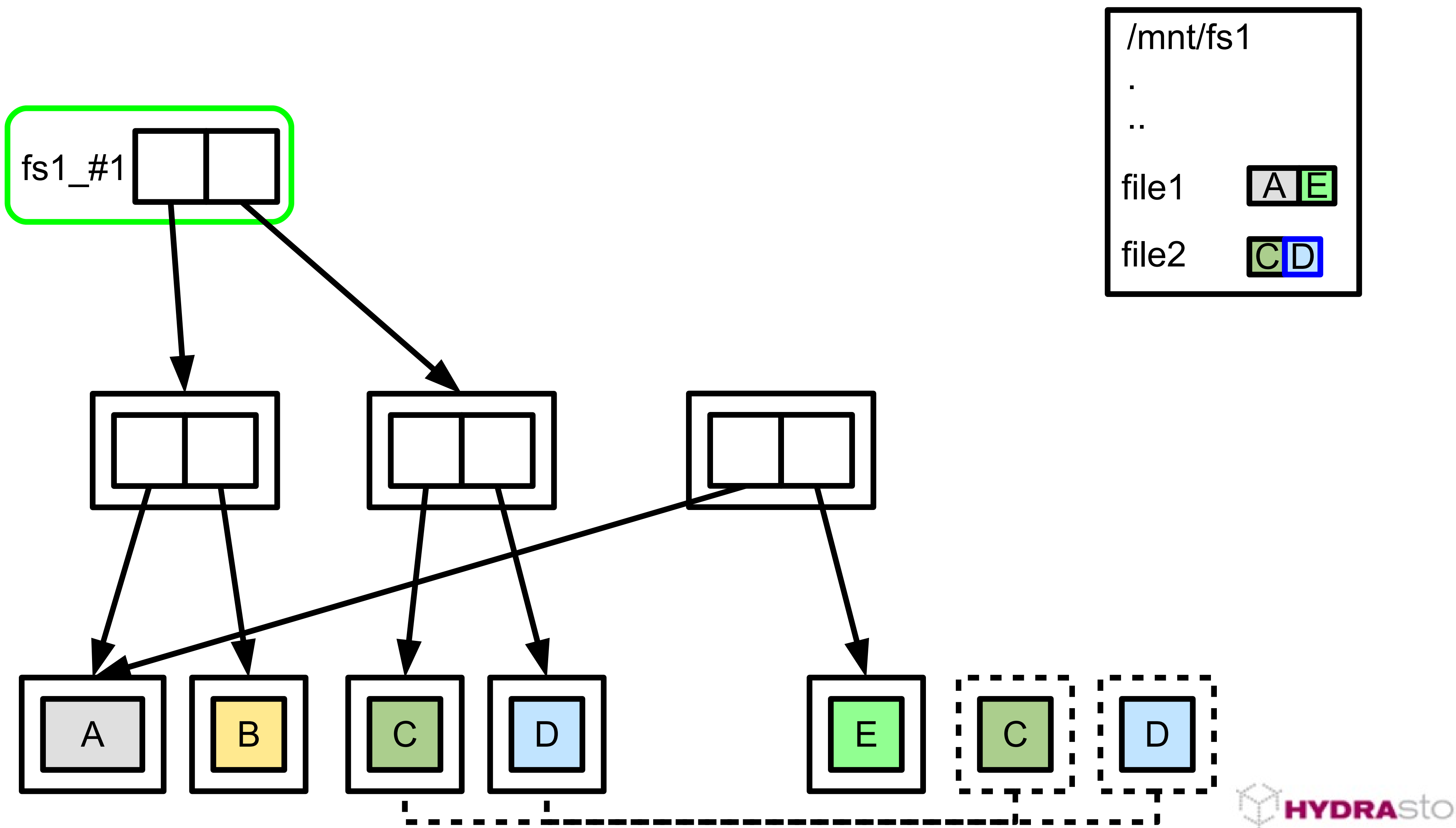
Data model in CAS storage system



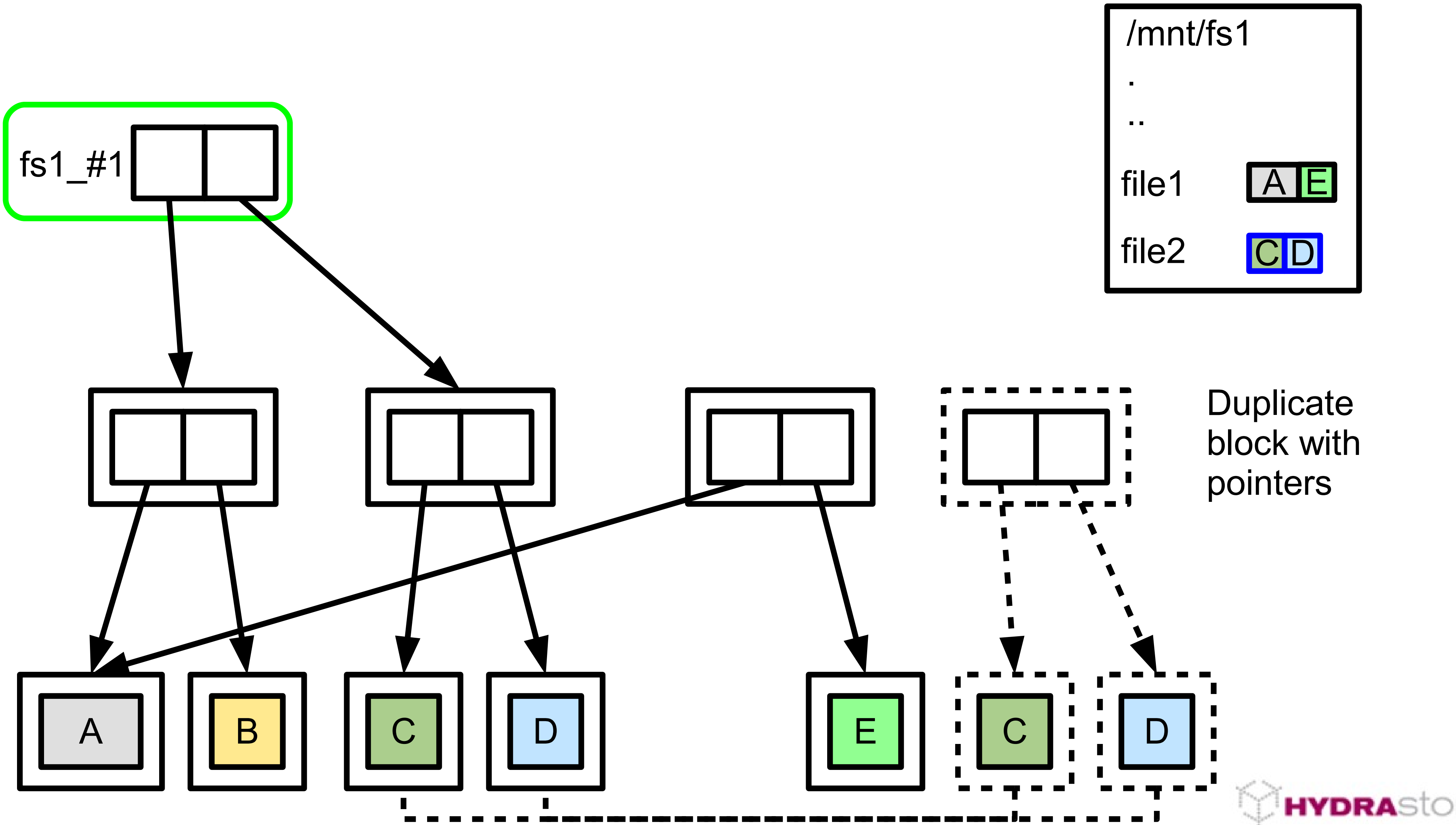
Data model in CAS storage system



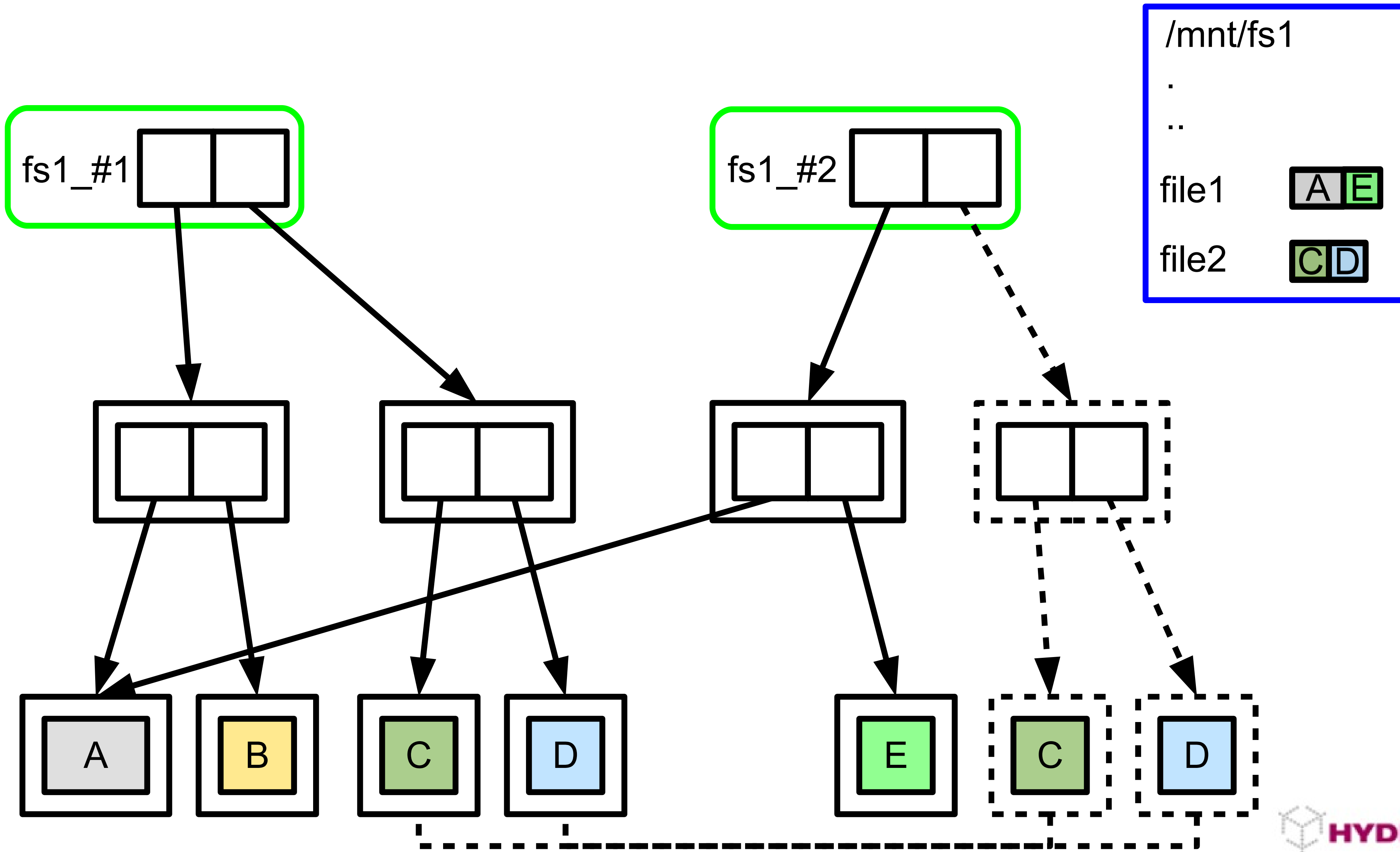
Data model in CAS storage system



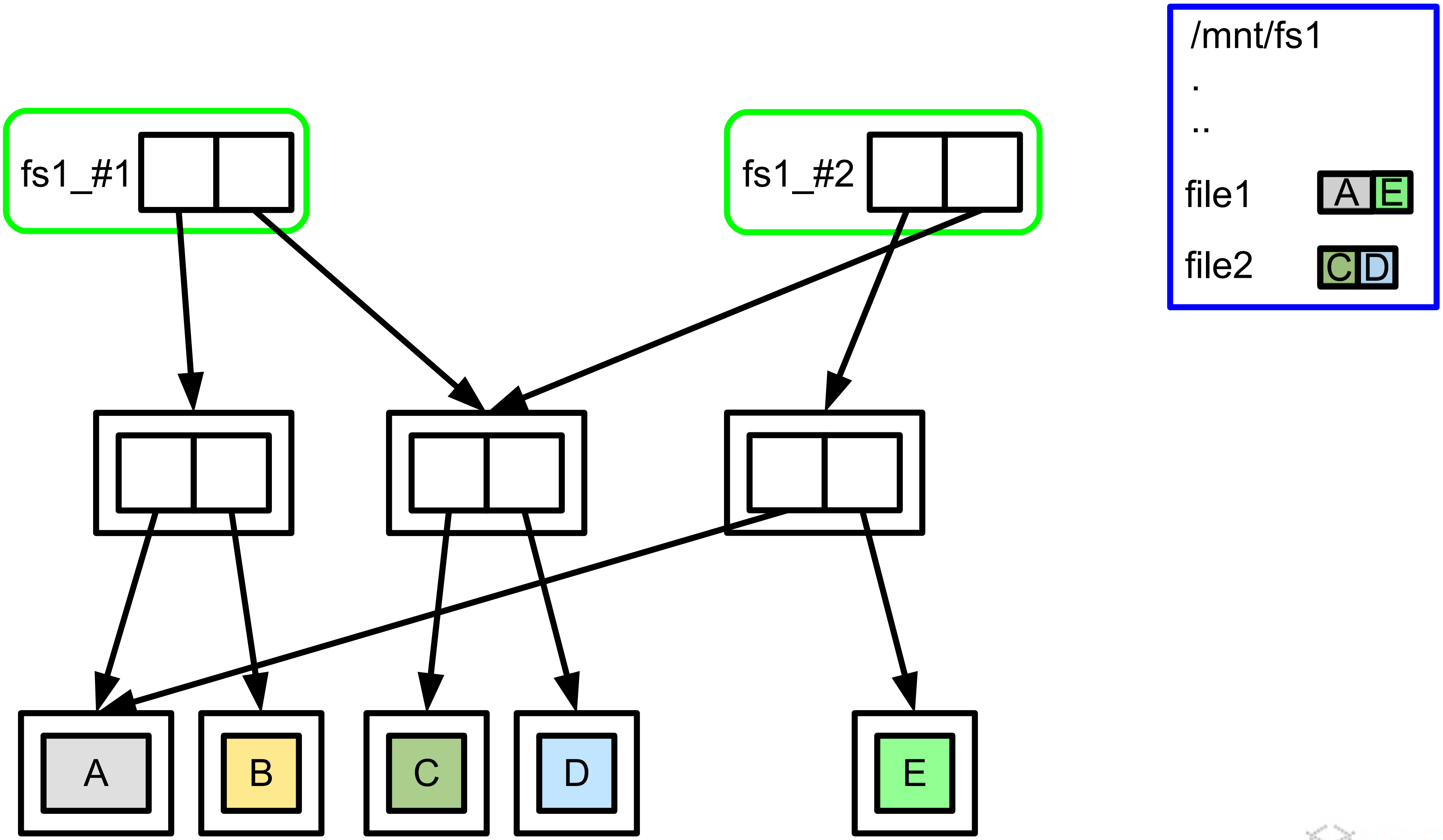
Data model in CAS storage system



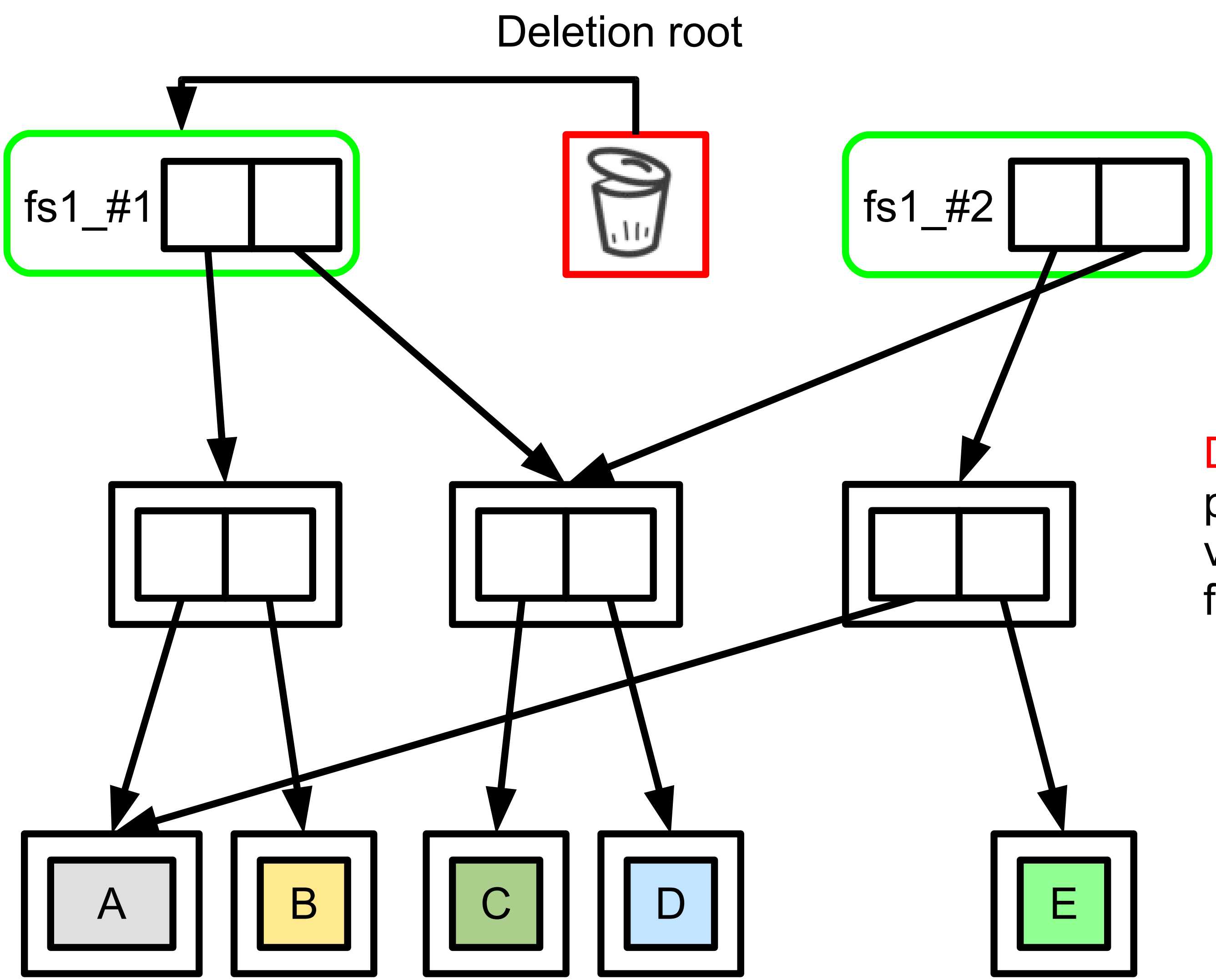
Data model in CAS storage system



Data model in CAS storage system

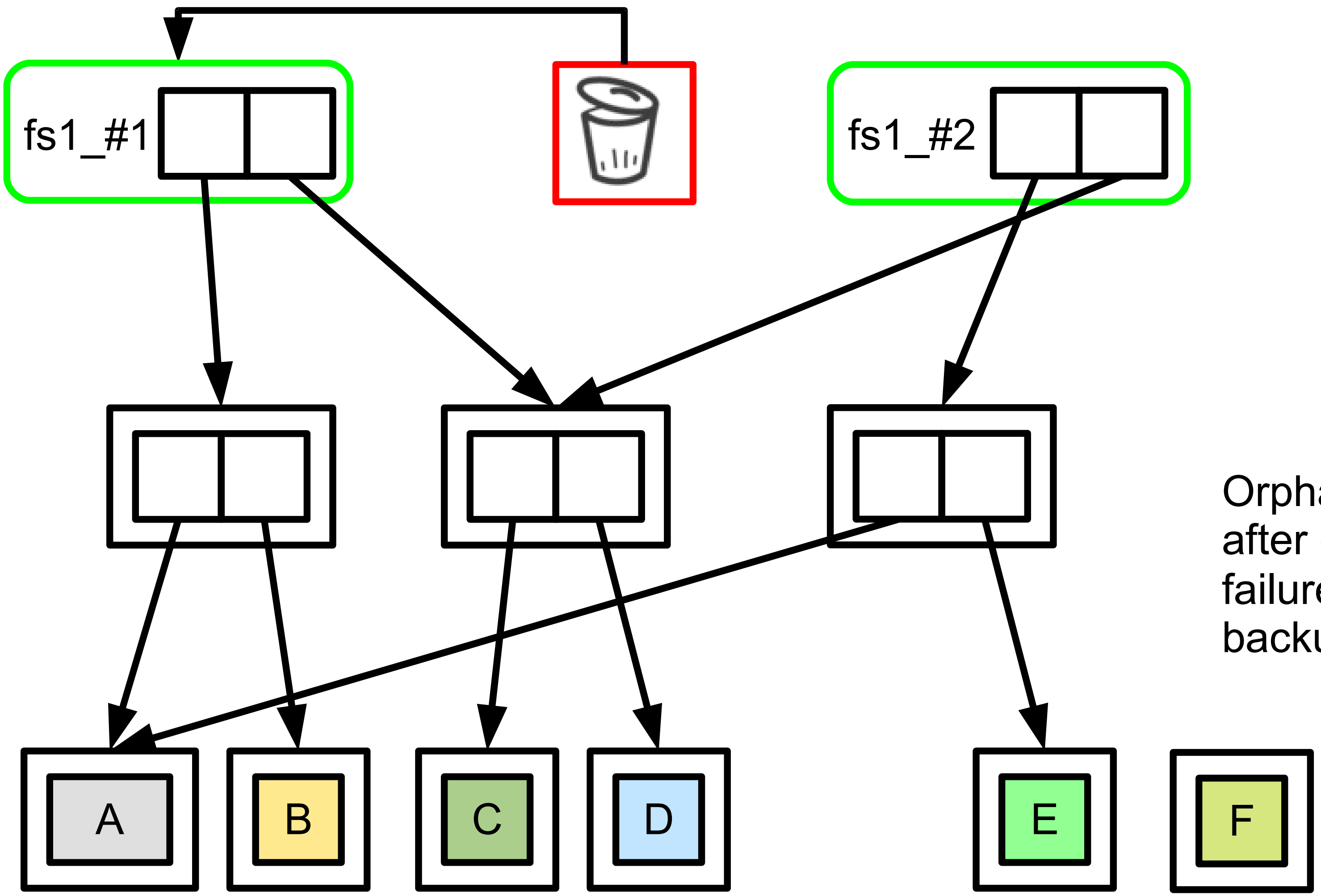


Data model in CAS storage system



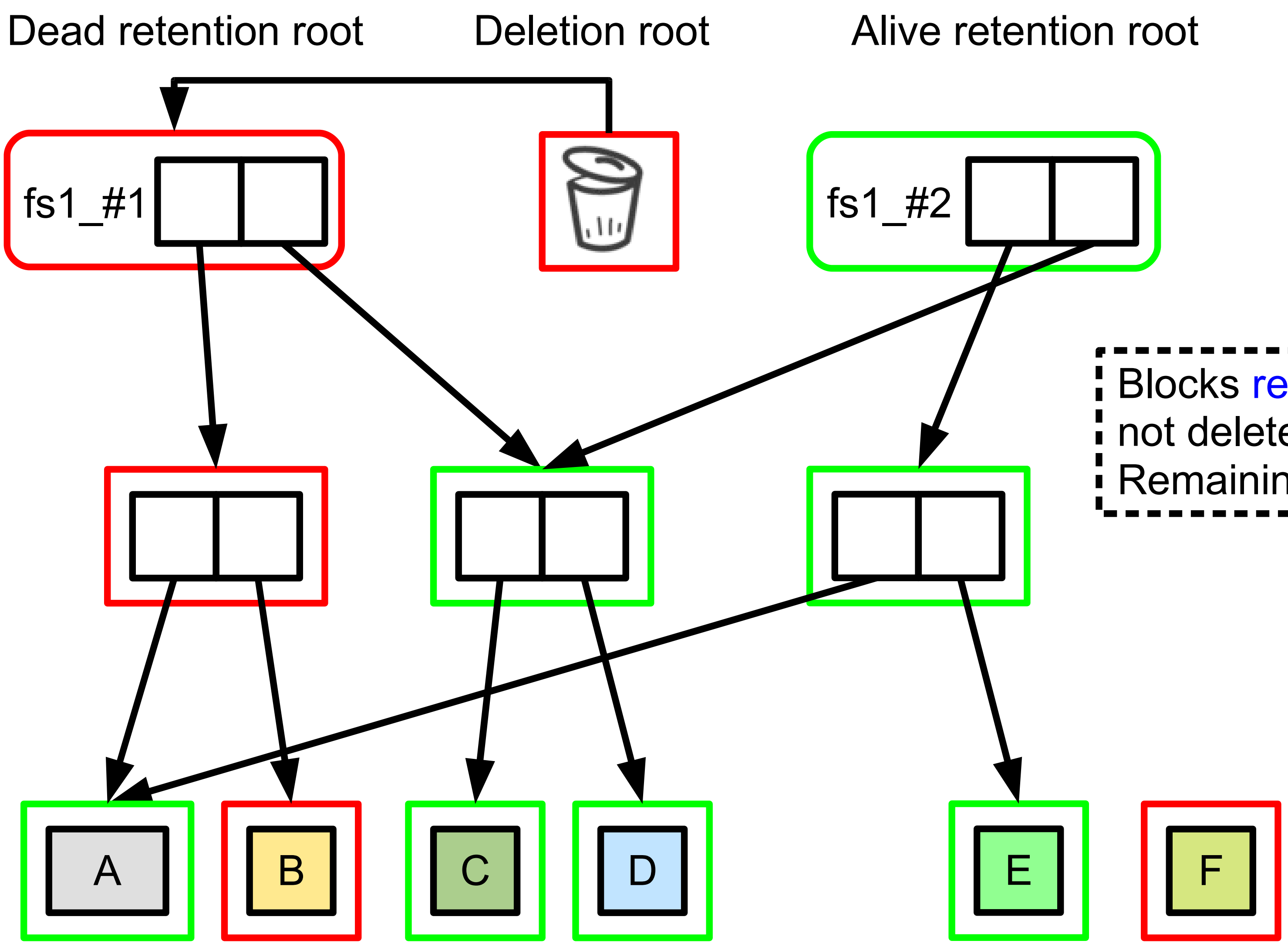
Delete previous version of a filesystem

Data model in CAS storage system



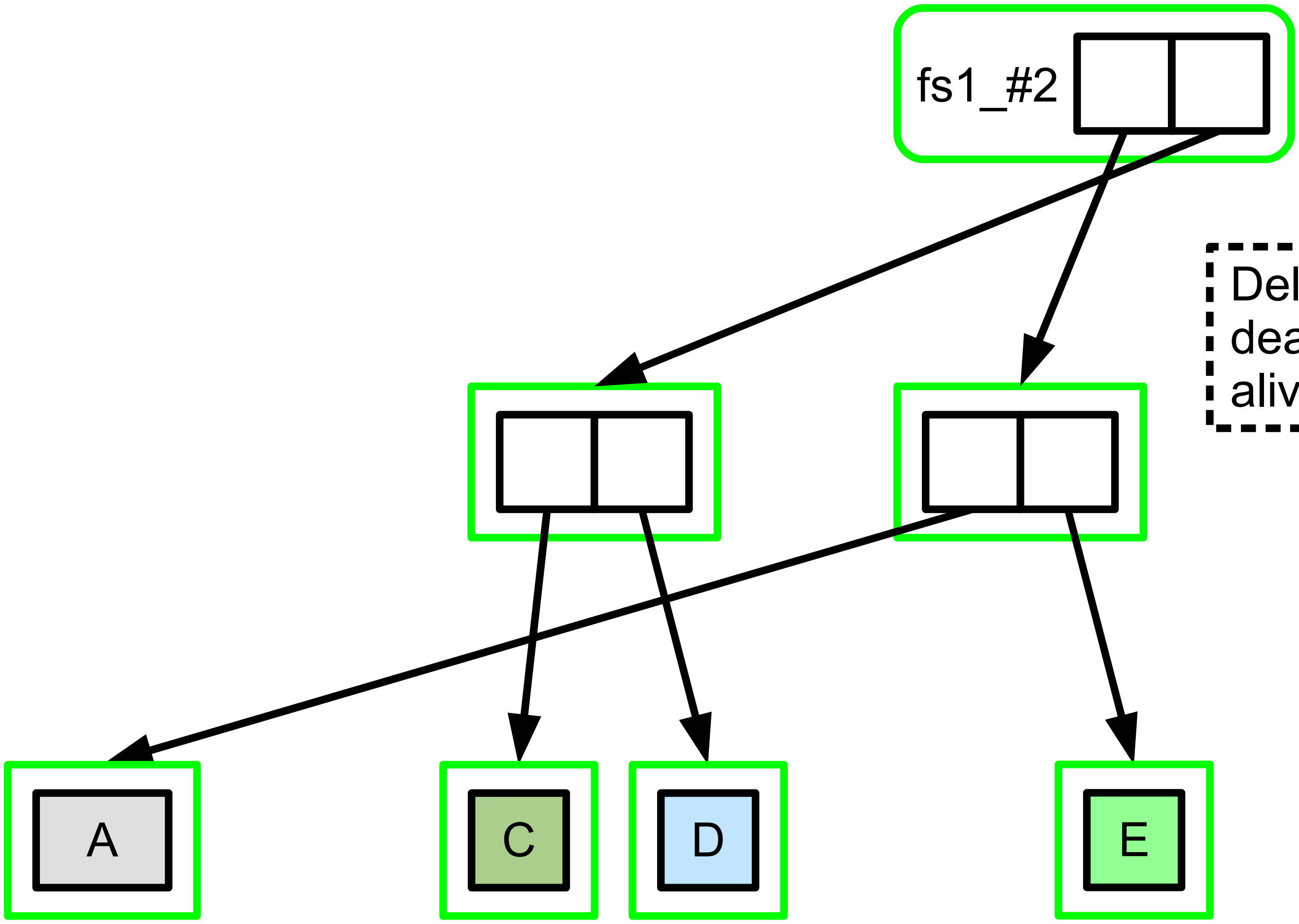
Orphaned block F after e.g. client failure during 3rd backup

Deletion semantics



Blocks reachable from not deleted roots are alive. Remaining blocks are dead.

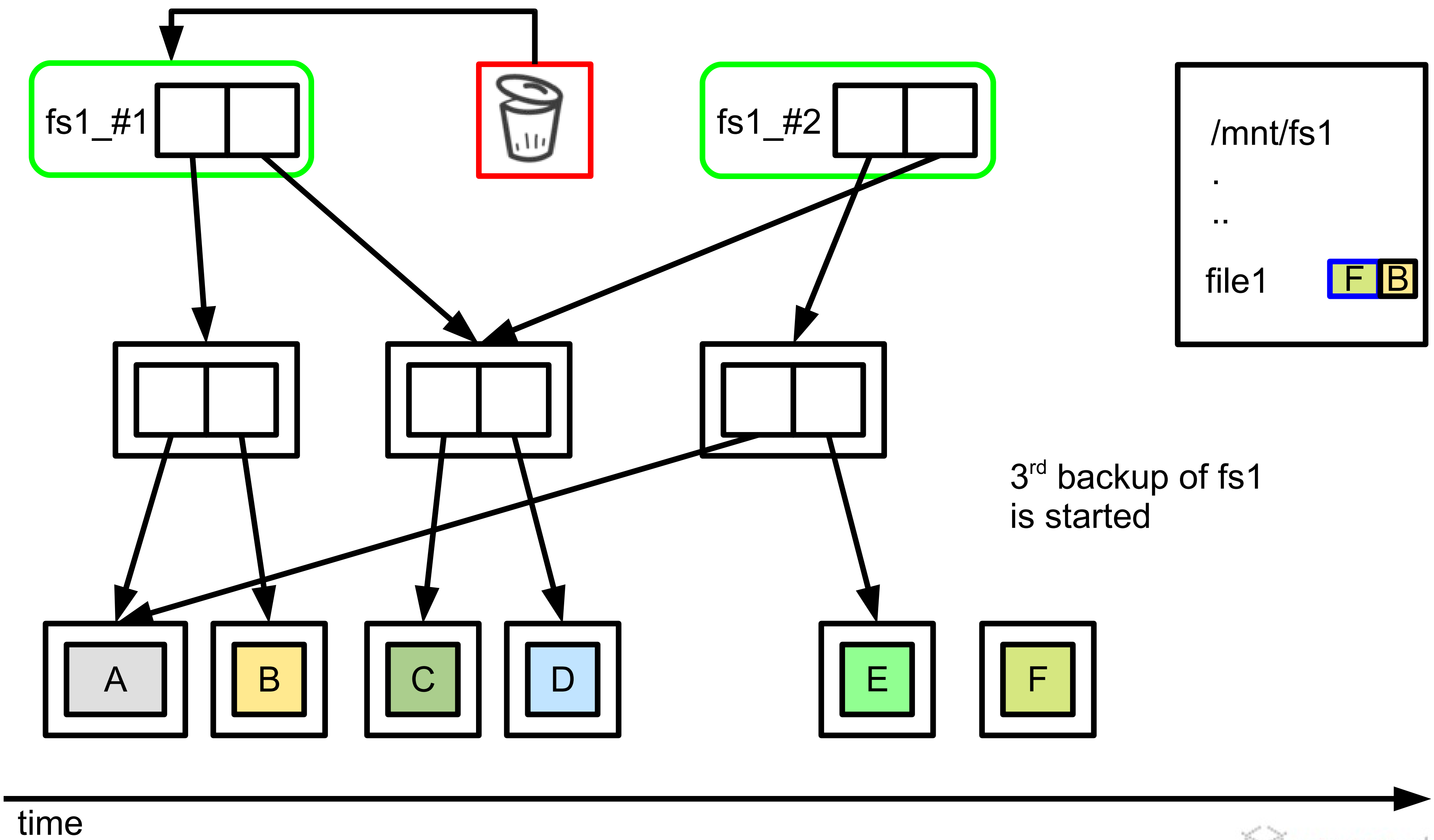
Deletion semantics



Deletion algorithm deletes dead blocks and preserves alive blocks

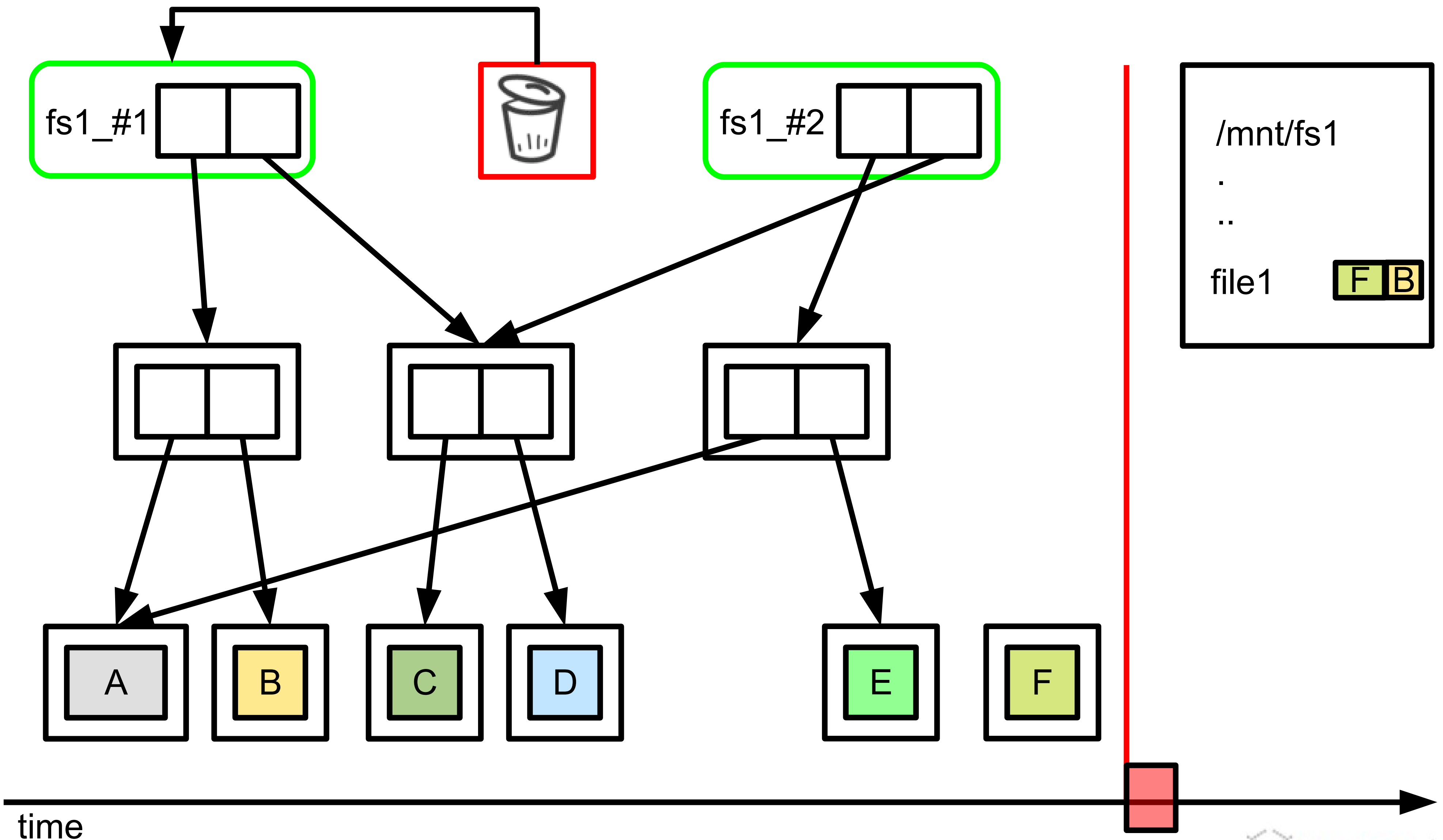
Challenges for deletion in CAS systems

Challenges for deletion in CAS systems

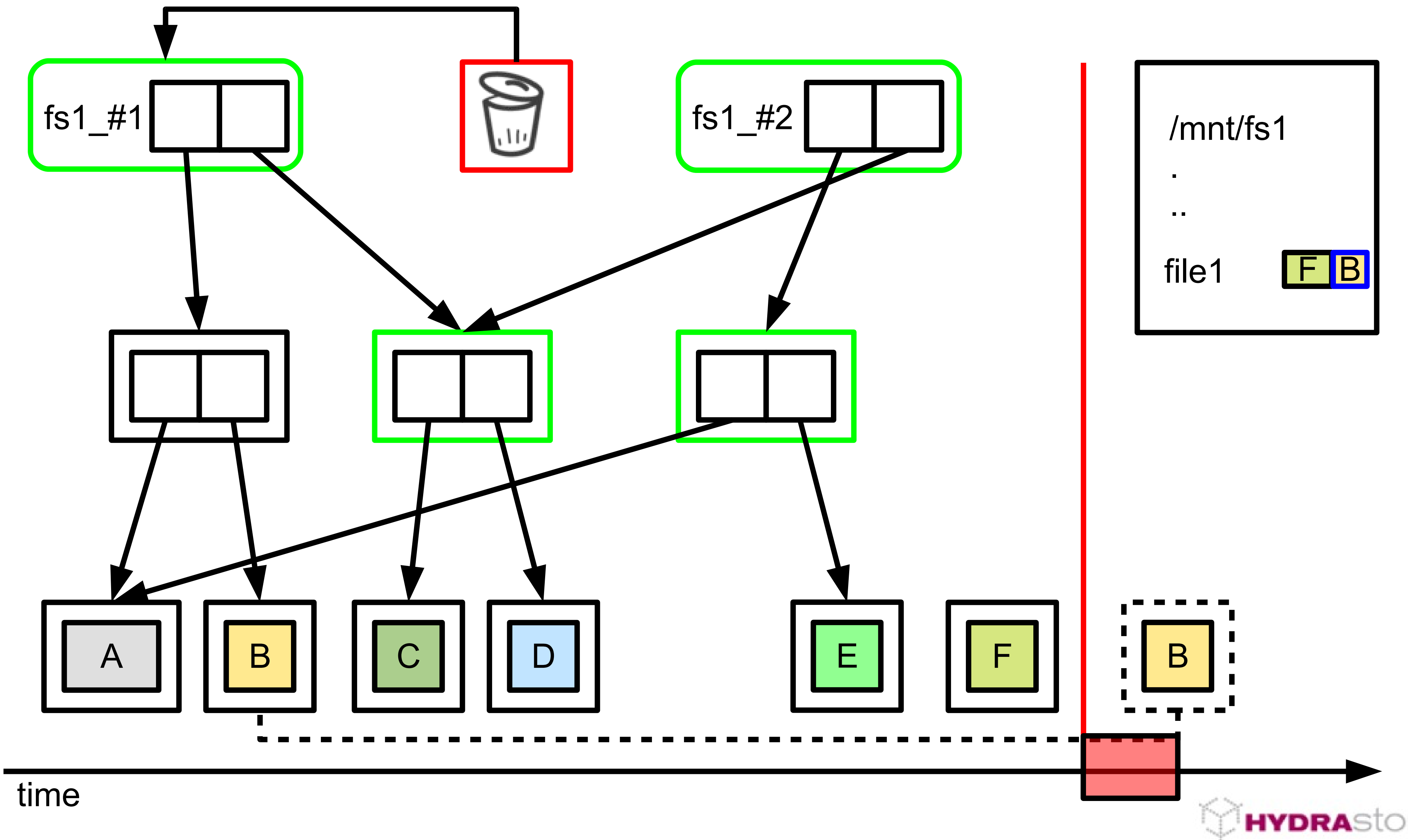


3rd backup of fs1
is started

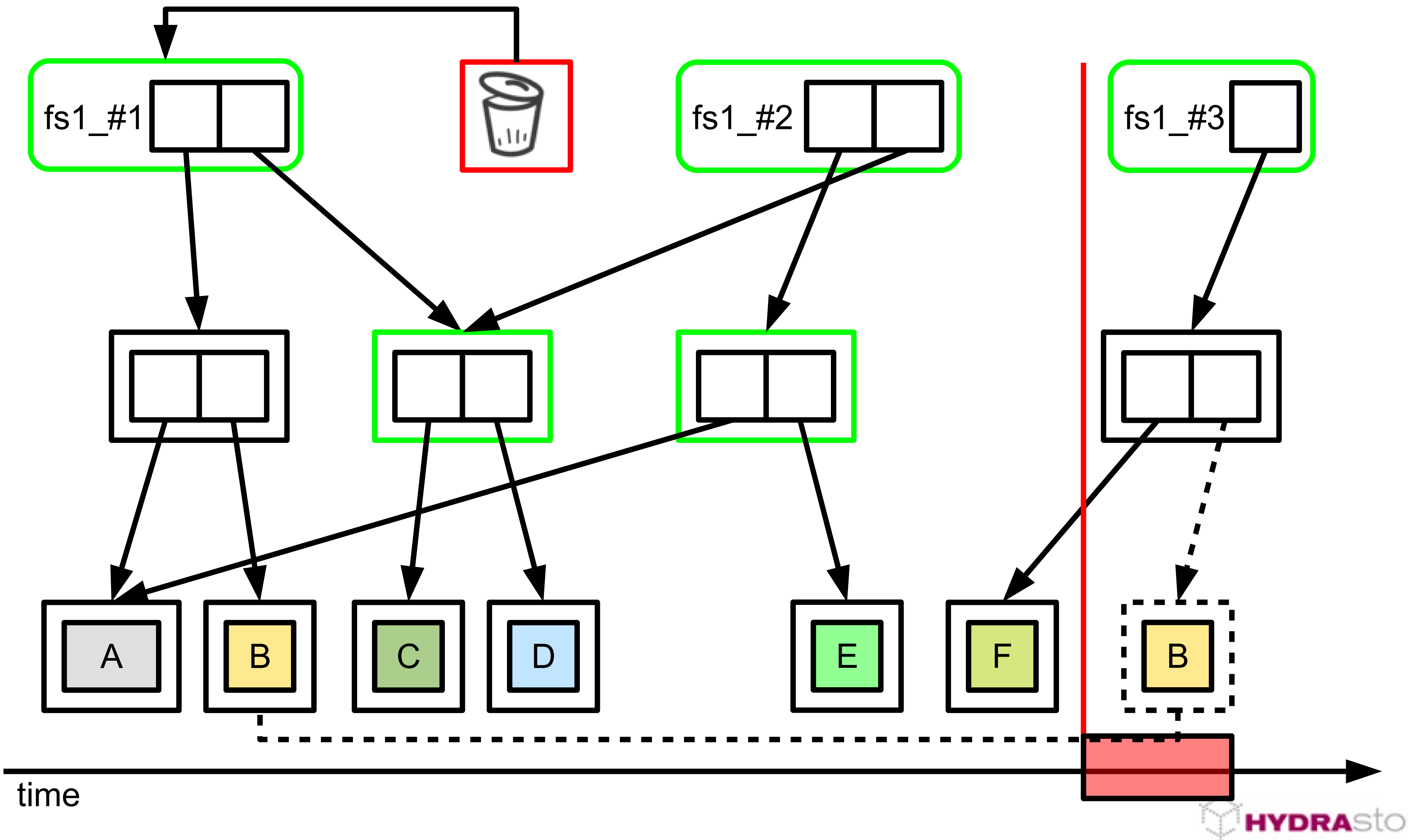
Challenges for deletion in CAS systems



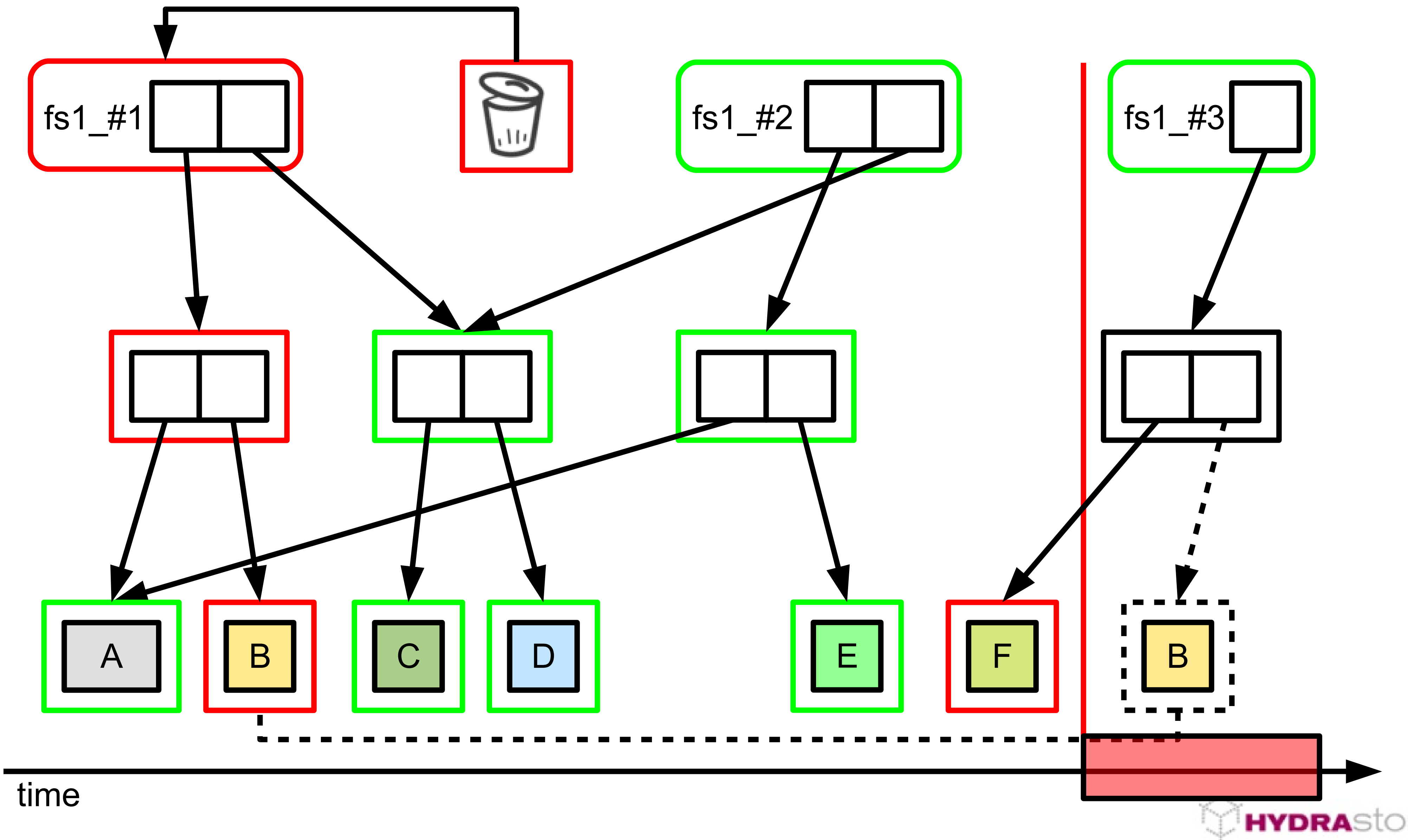
Challenges for deletion in CAS systems



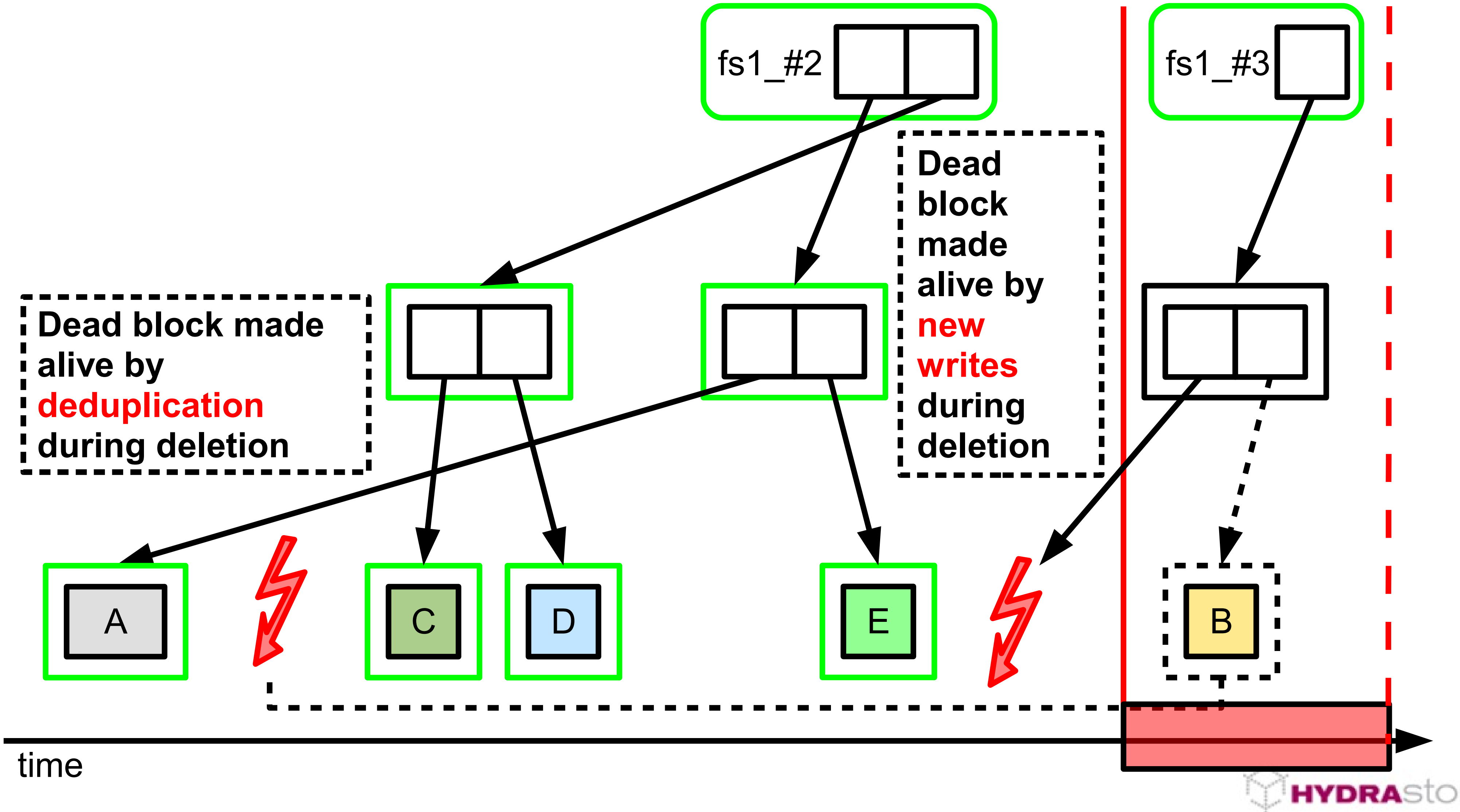
Challenges for deletion in CAS systems



Challenges for deletion in CAS systems



Challenges for deletion in CAS systems

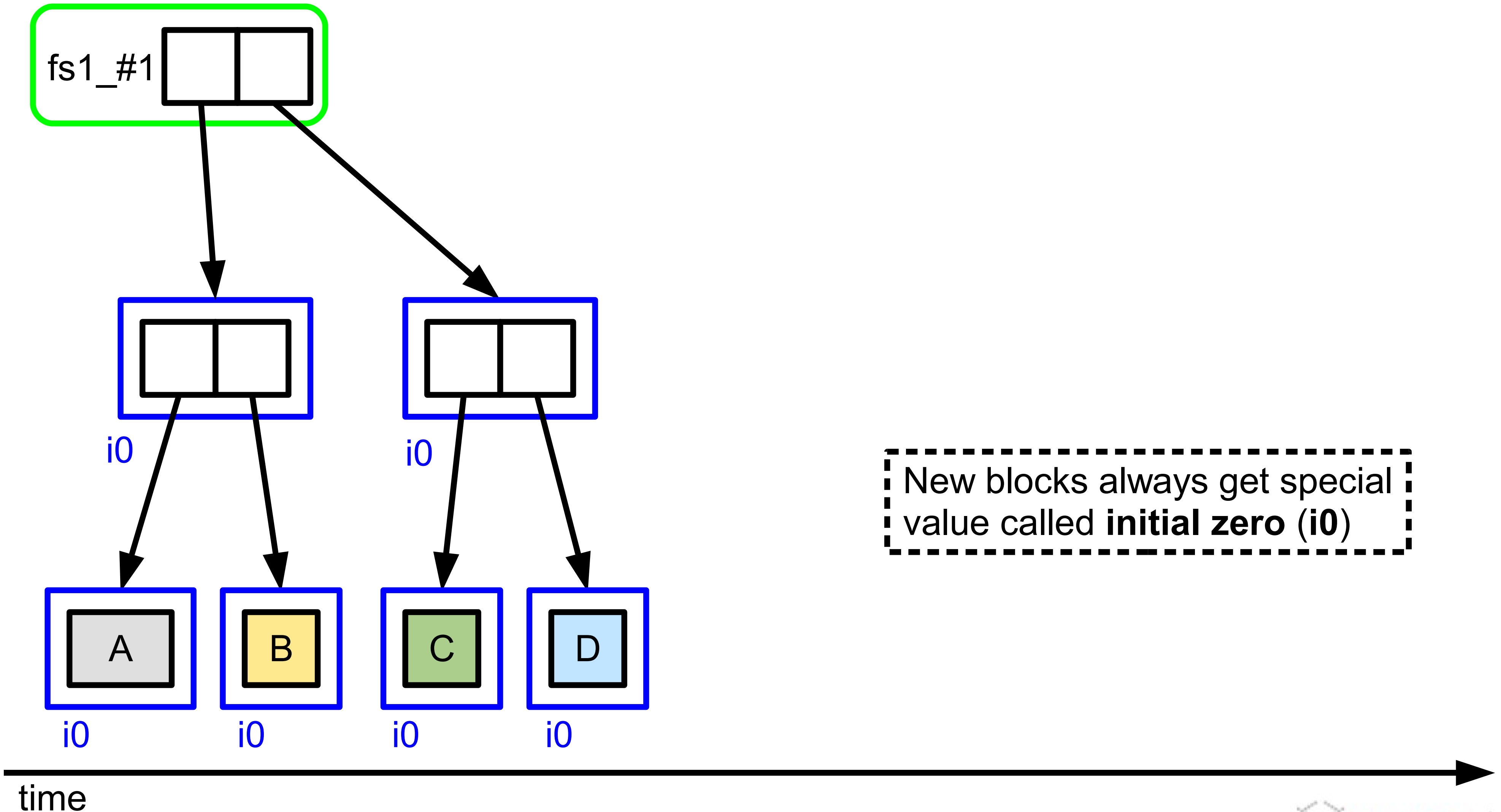


Base deletion algorithm in centralized CAS system

Base deletion algorithm

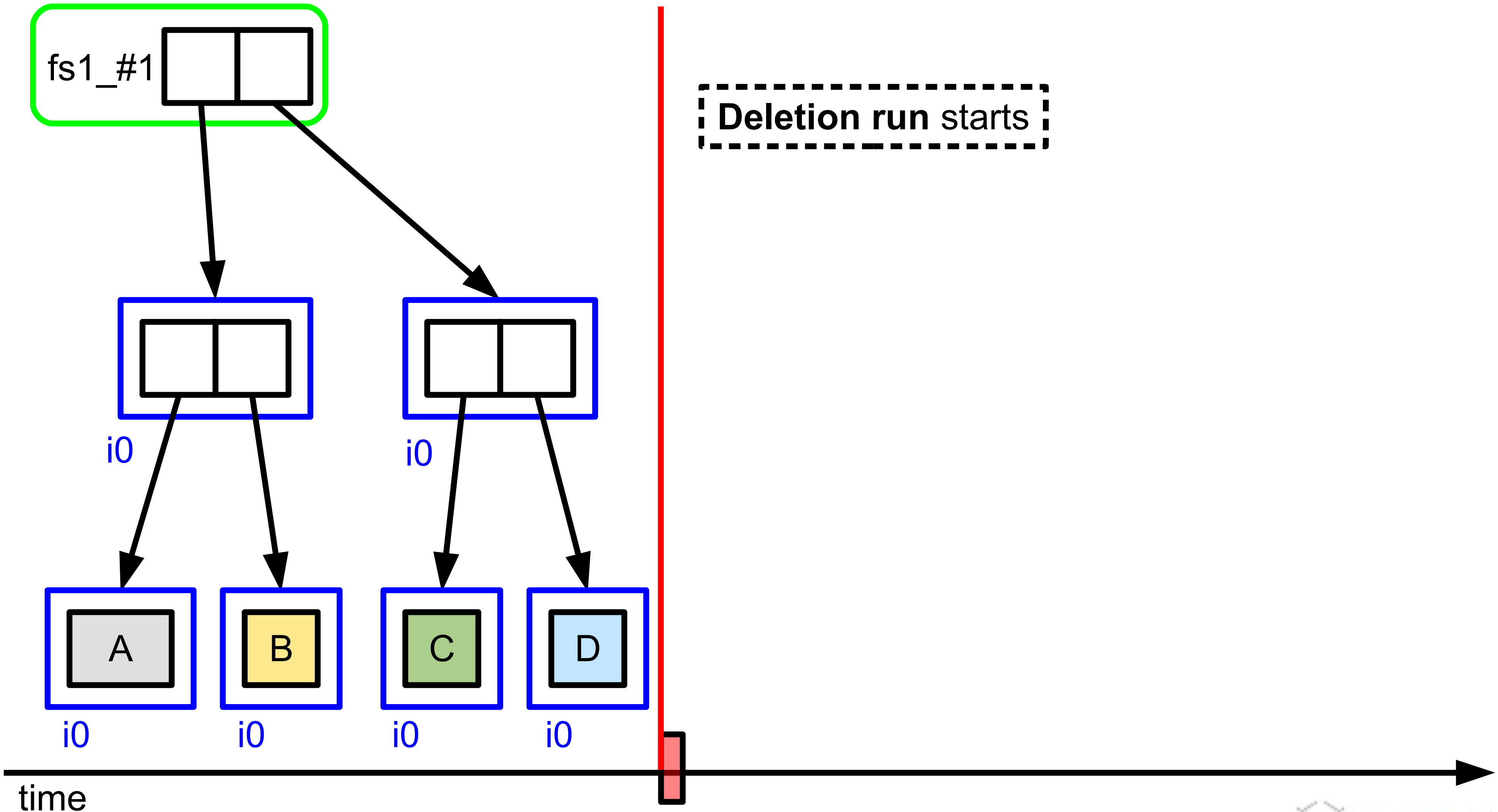
- Design decisions
  - **deferred reference counting**
  - **incremental**
- Limitations
  - no writes during deletion
- Two phases:
  - **garbage identification**
  - **space reclamation**
- **Deletion runs** started on demand
- Two versions of counters
  - **effective**
  - **temporary** (become effective when deletion run ends)

Base algorithm in steps

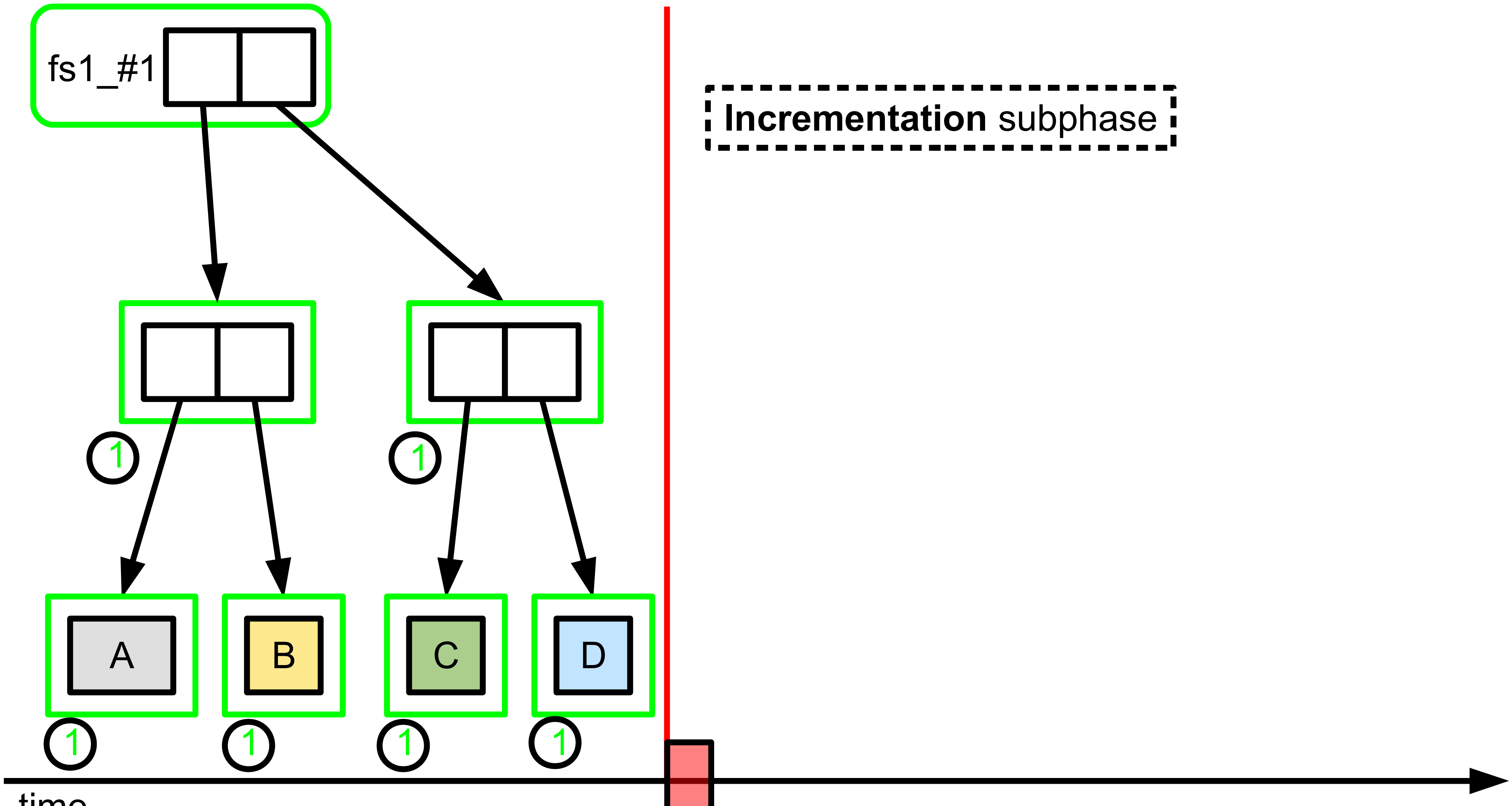


New blocks always get special value called **initial zero** (**i0**)

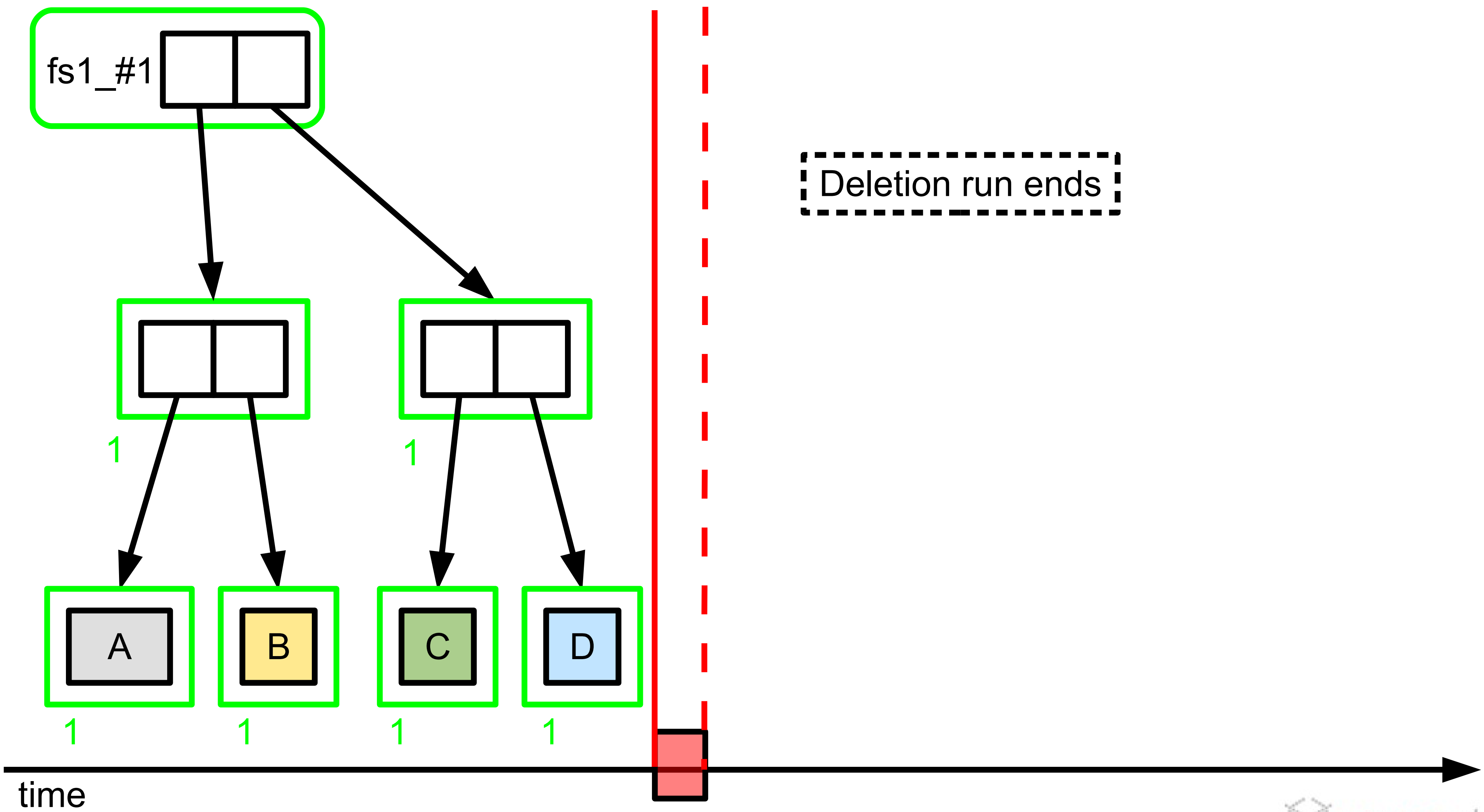
Base algorithm in steps



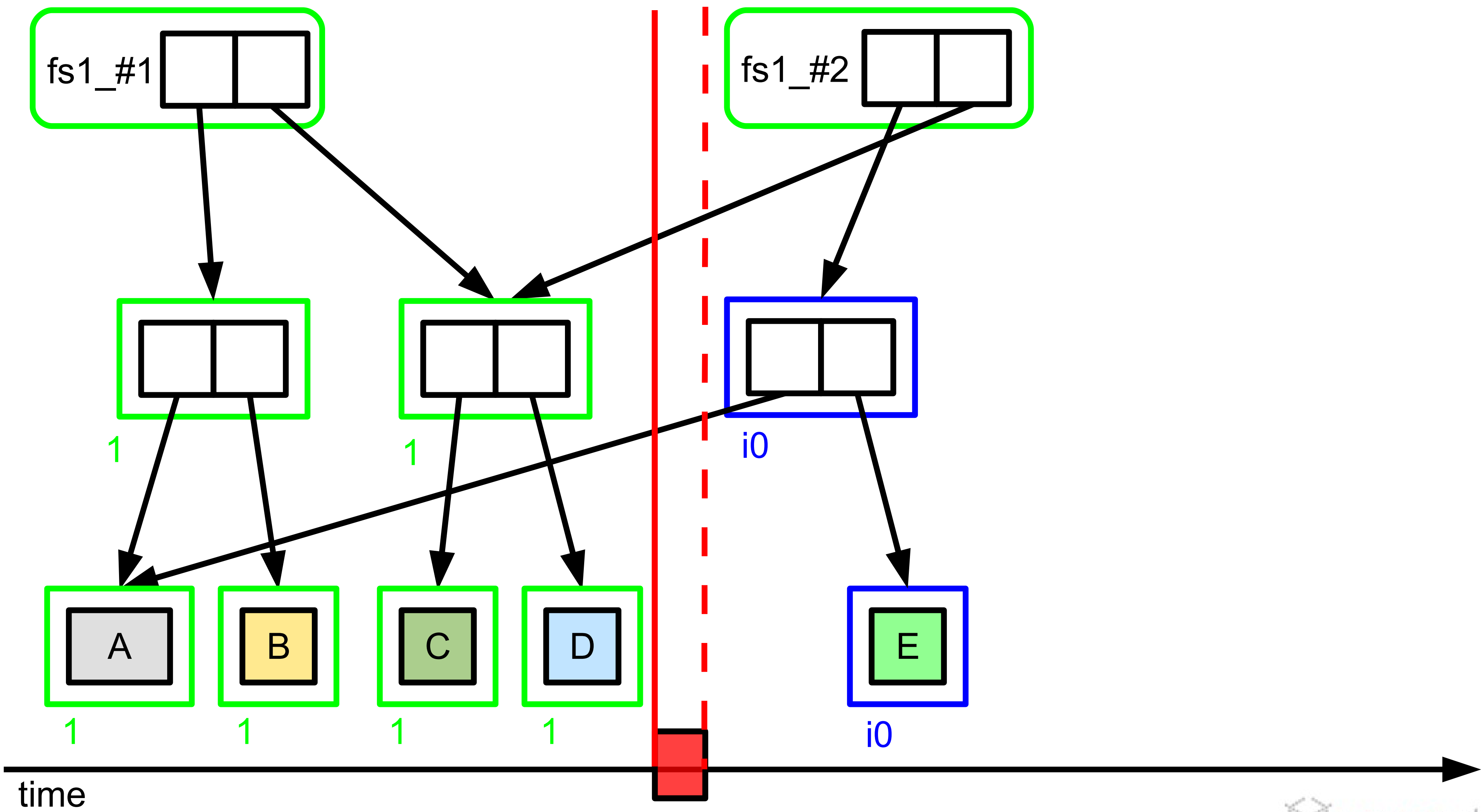
Base algorithm in steps



Base algorithm in steps

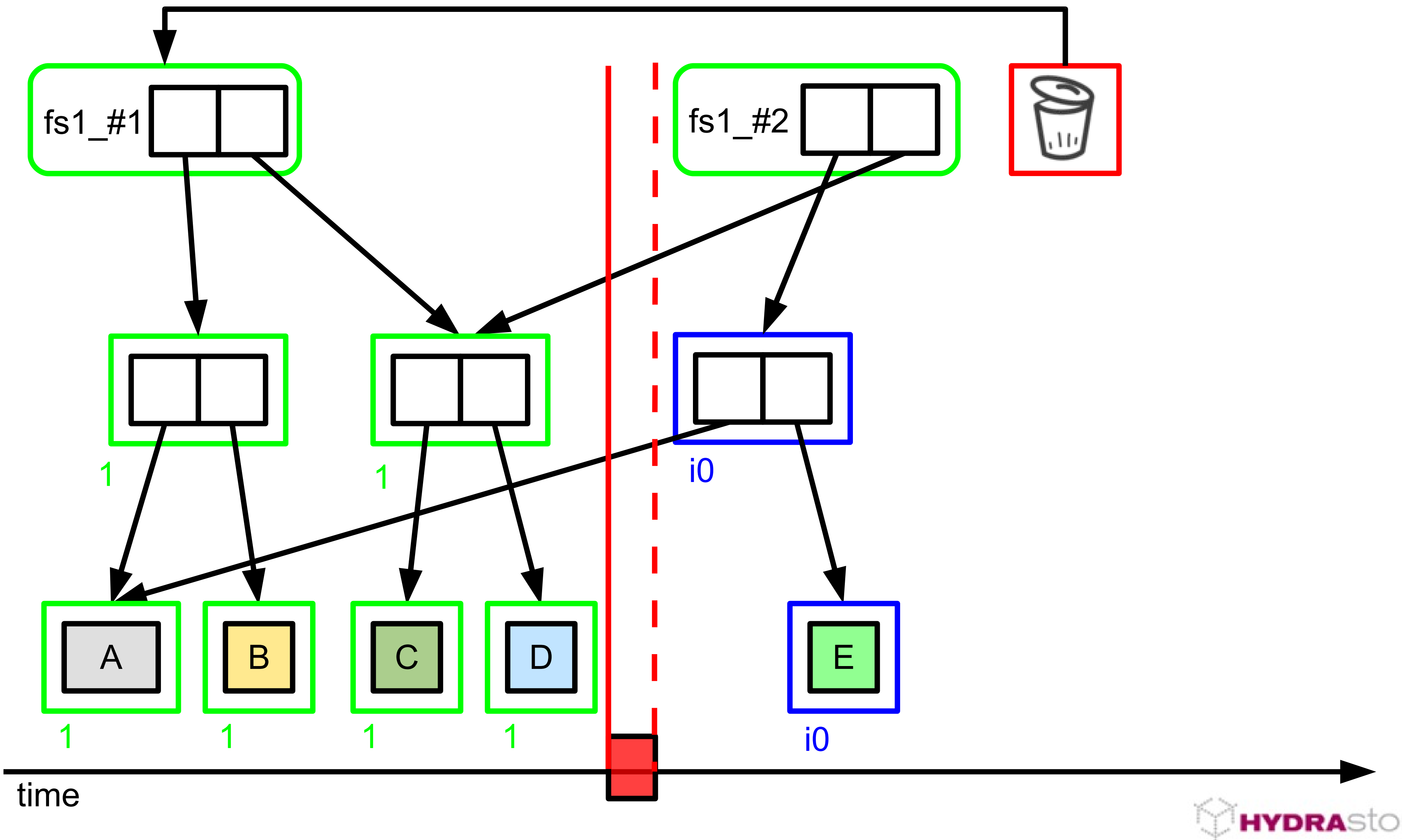


Base algorithm in steps

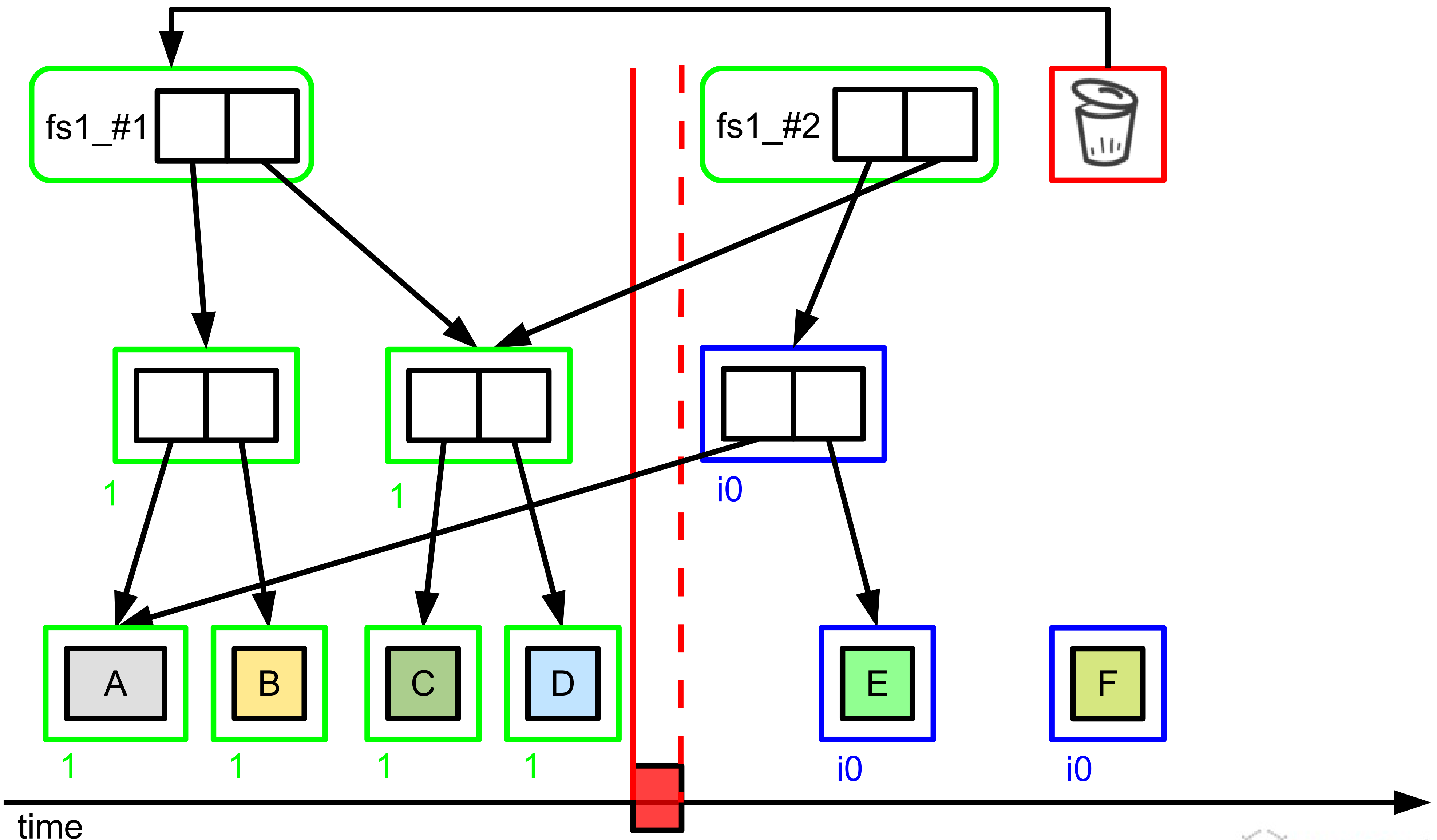


HYDRAstor

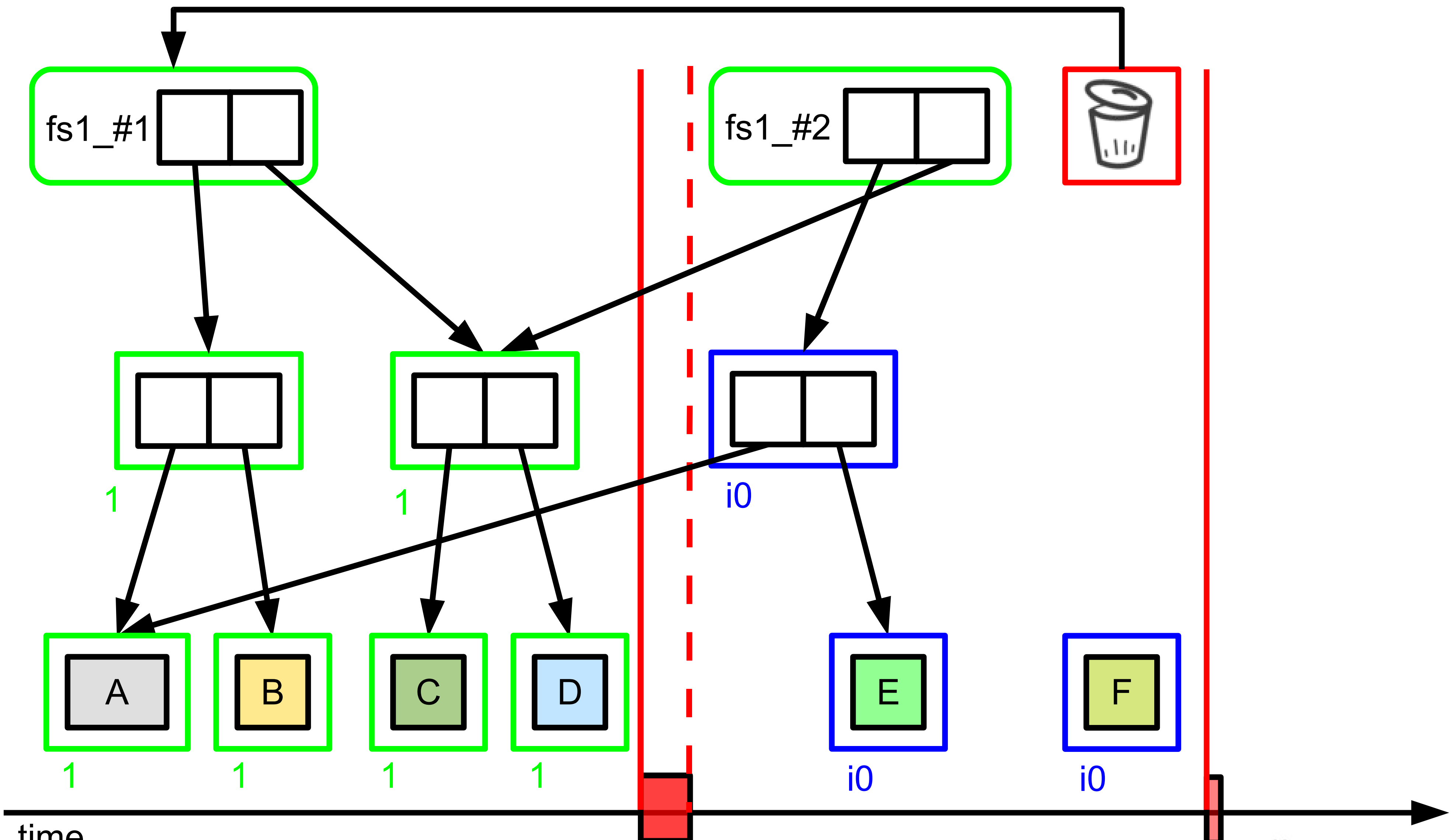
Base algorithm in steps



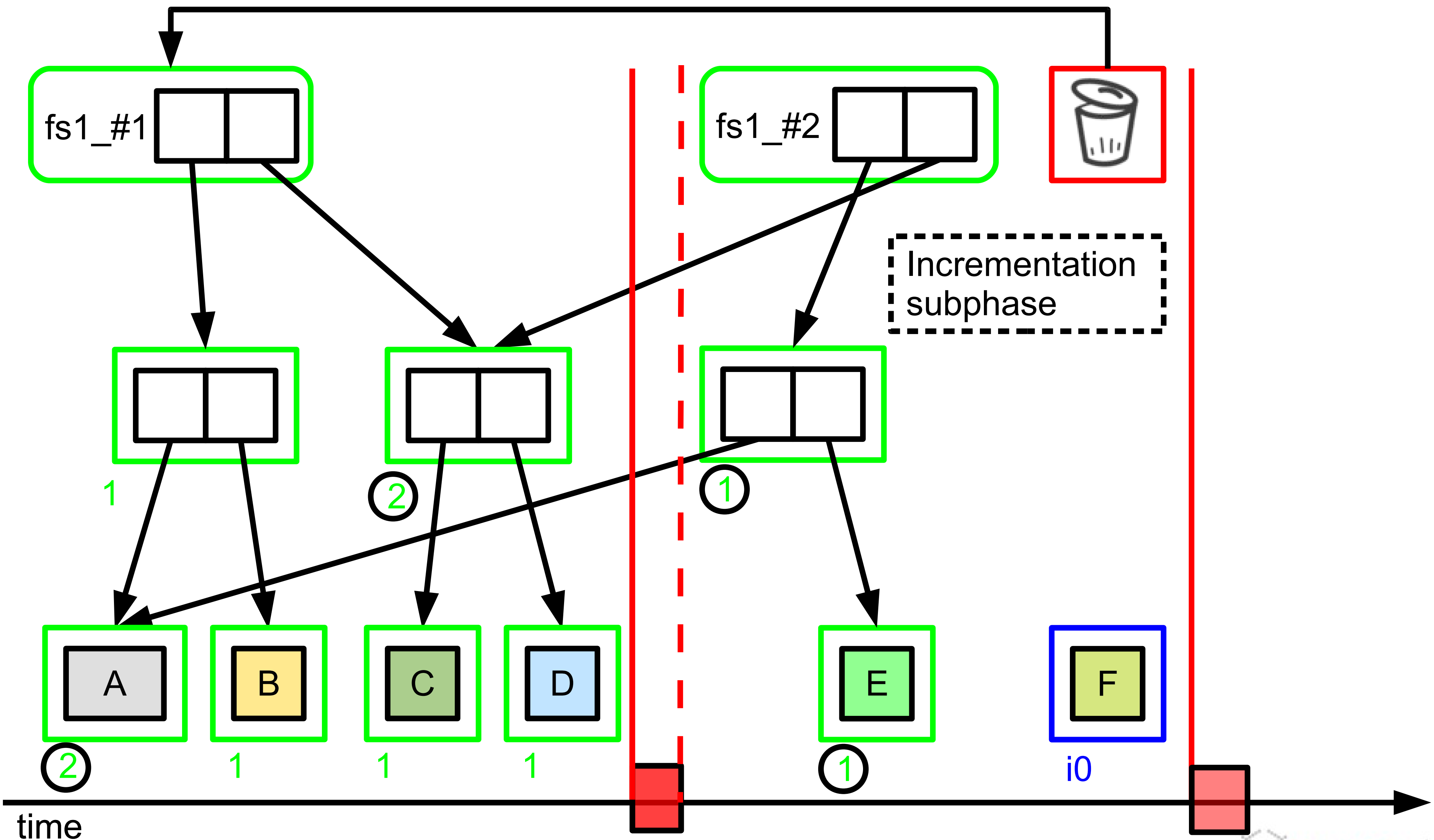
Base algorithm in steps



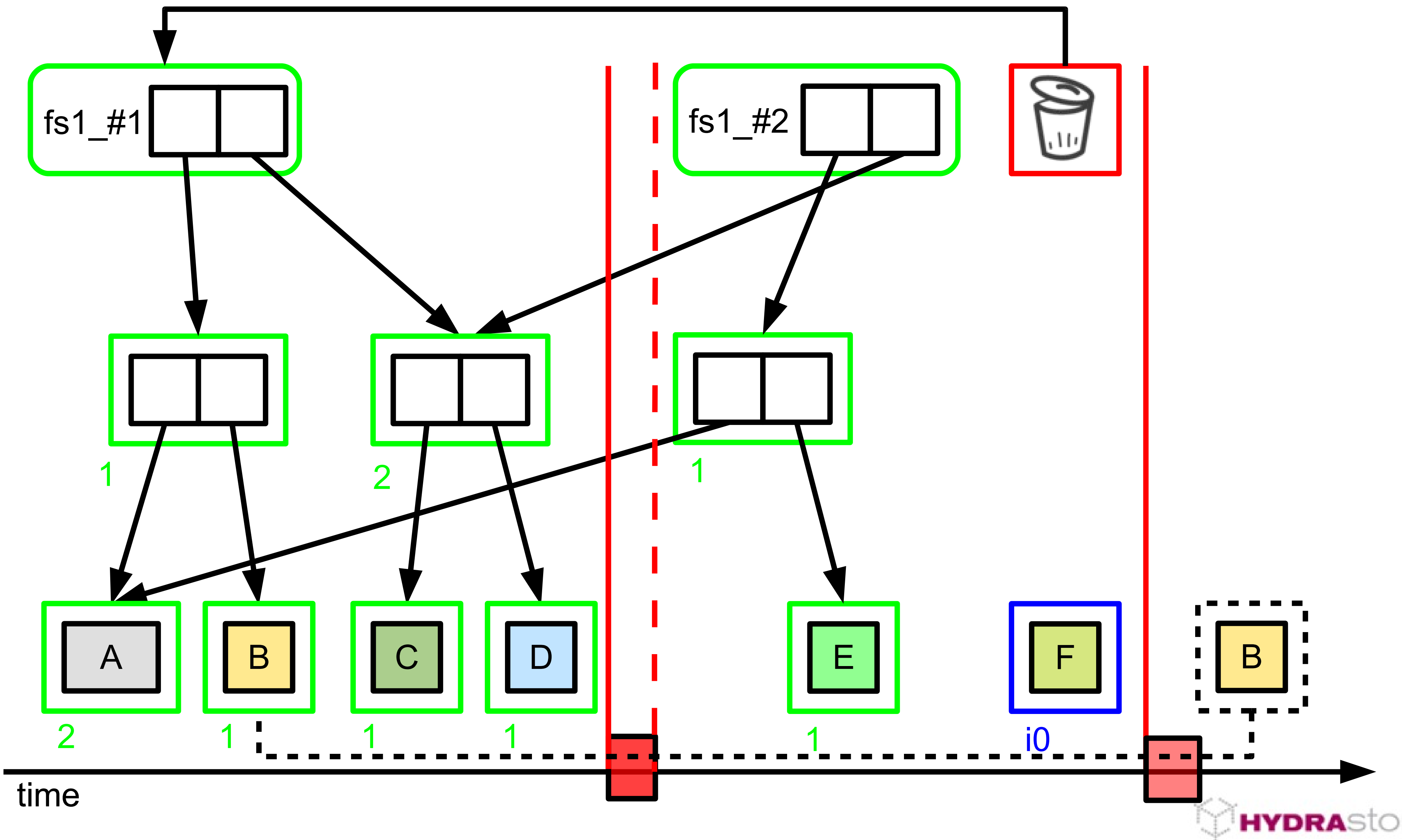
Base algorithm in steps



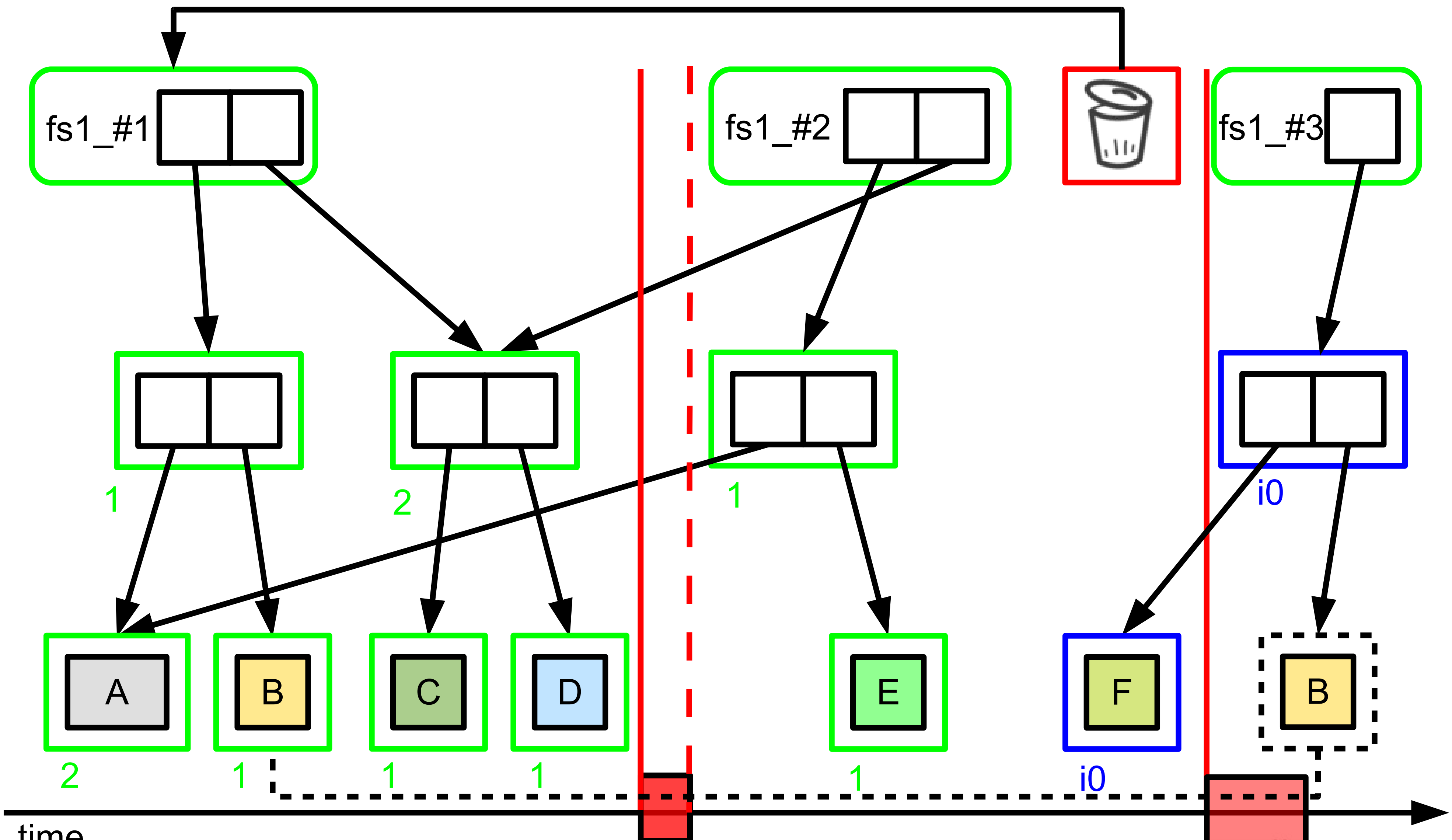
Base algorithm in steps



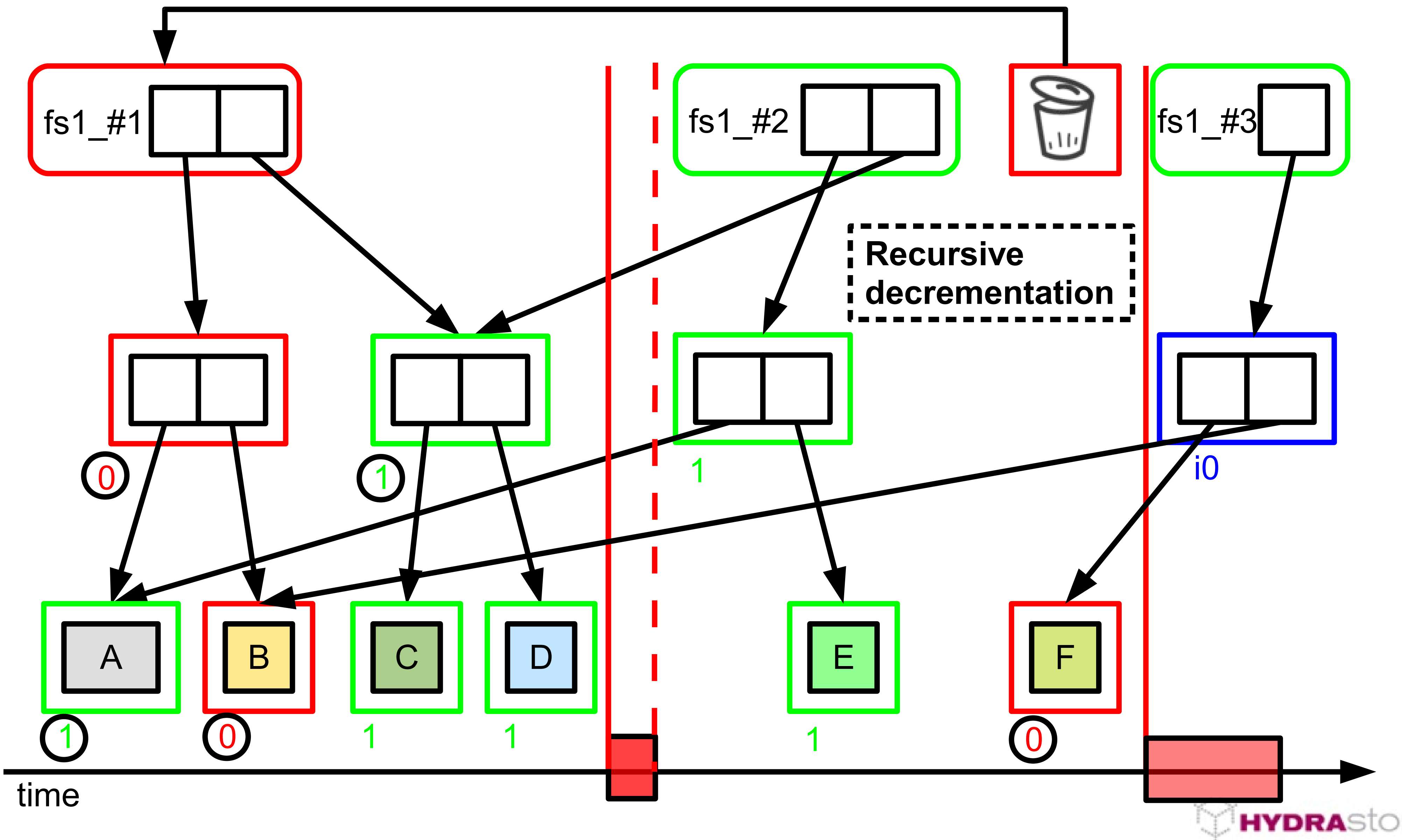
Base algorithm in steps



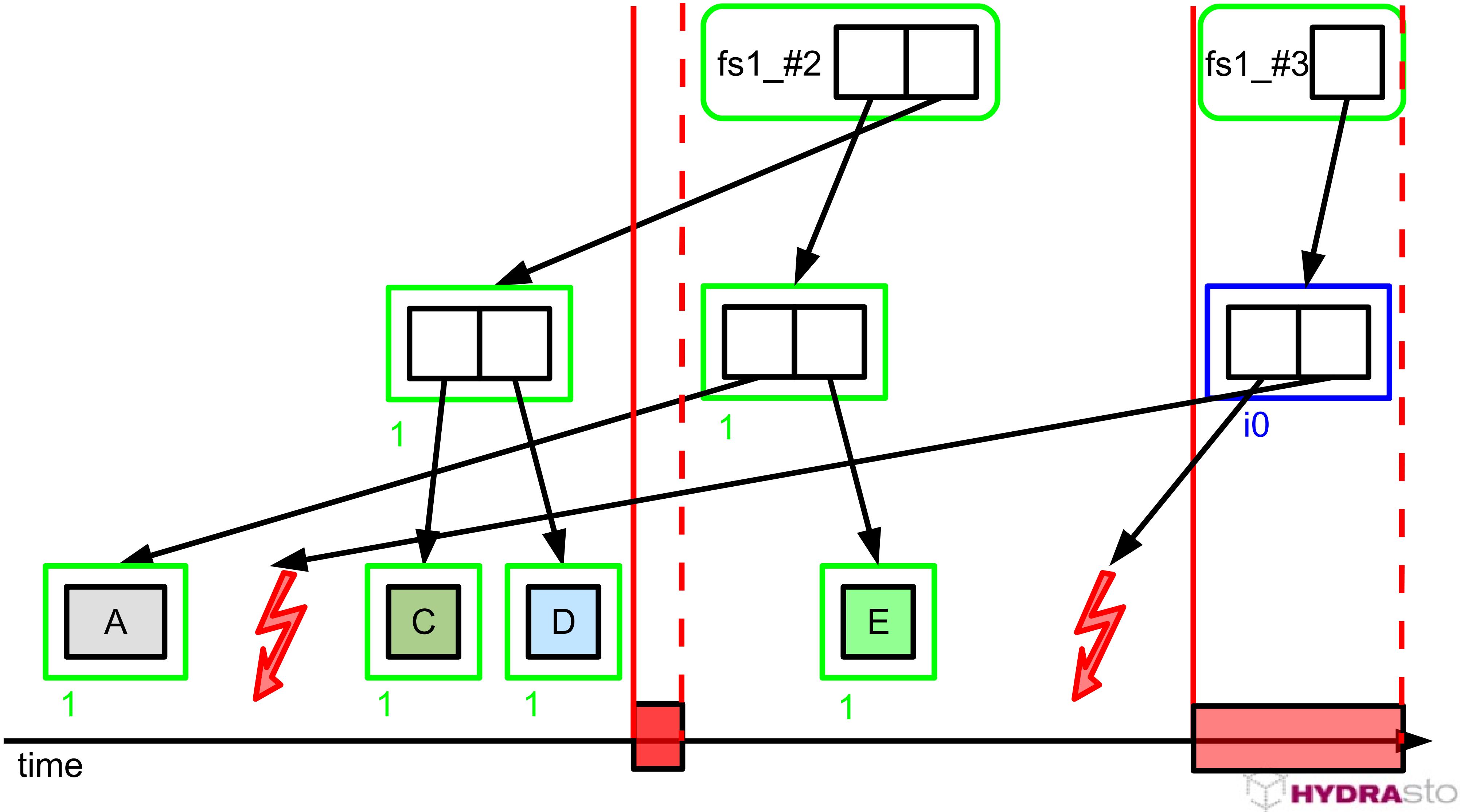
Base algorithm in steps



Base algorithm in steps

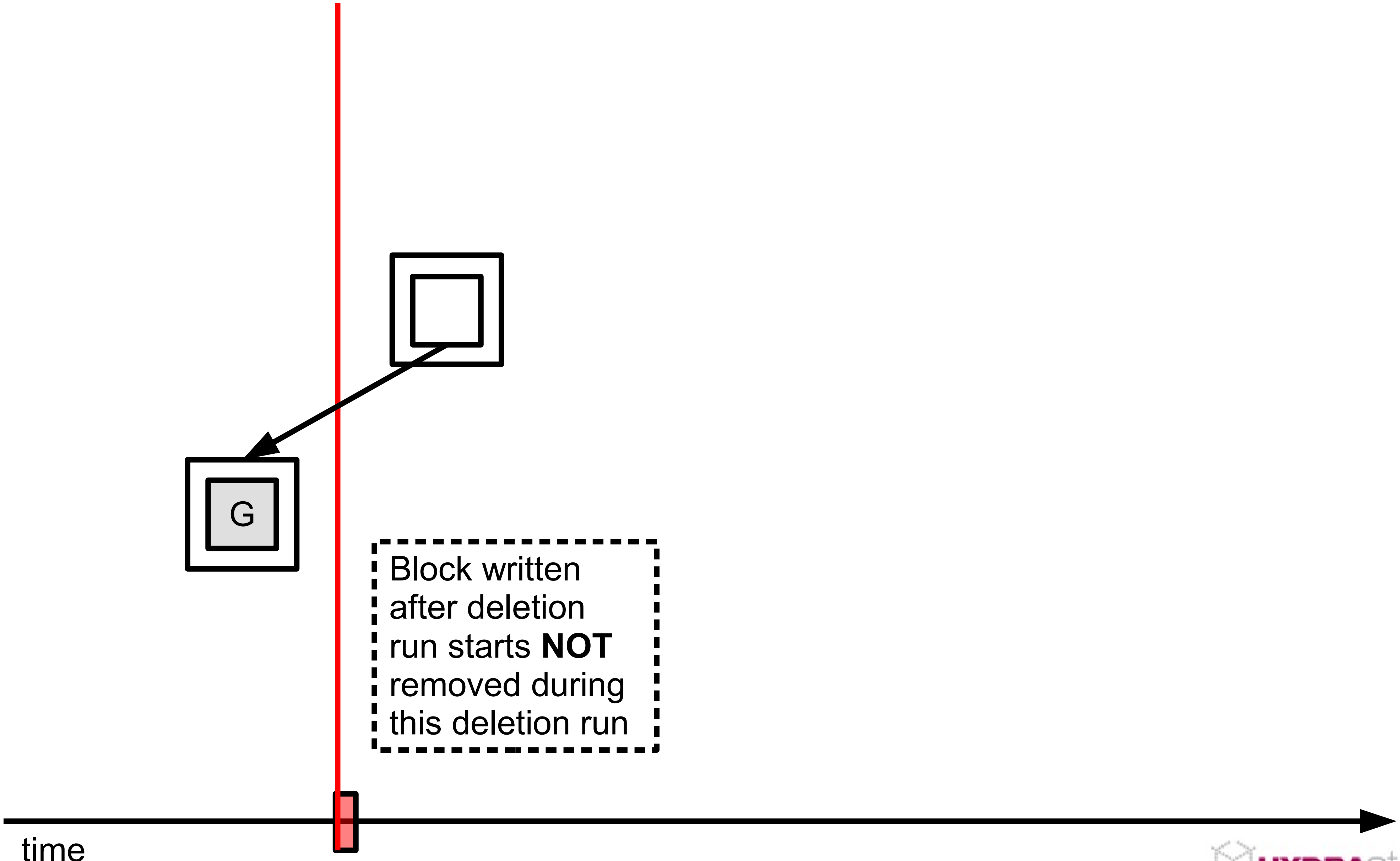


Problems with base algorithm

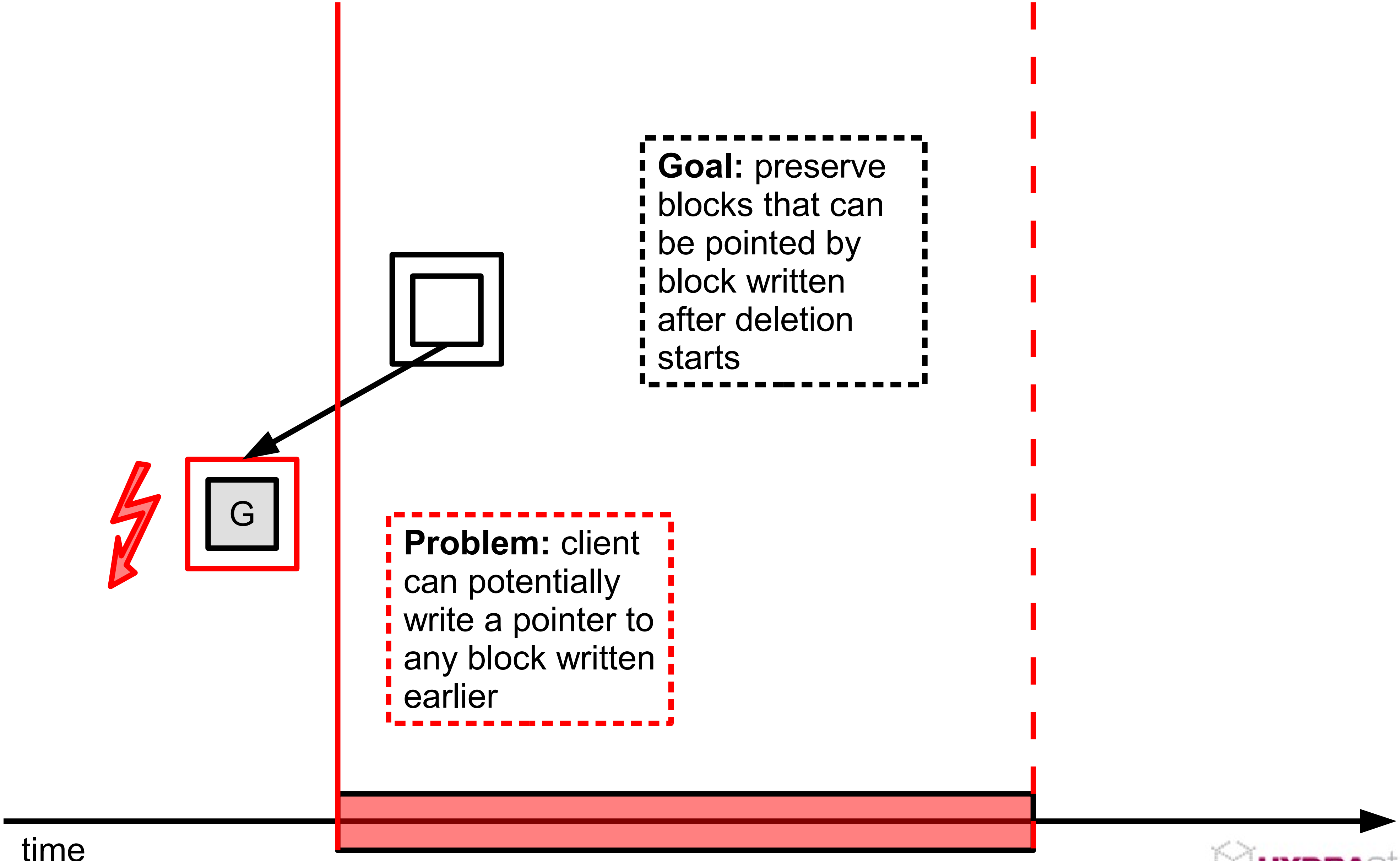


Supporting new writes with deduplication enabled

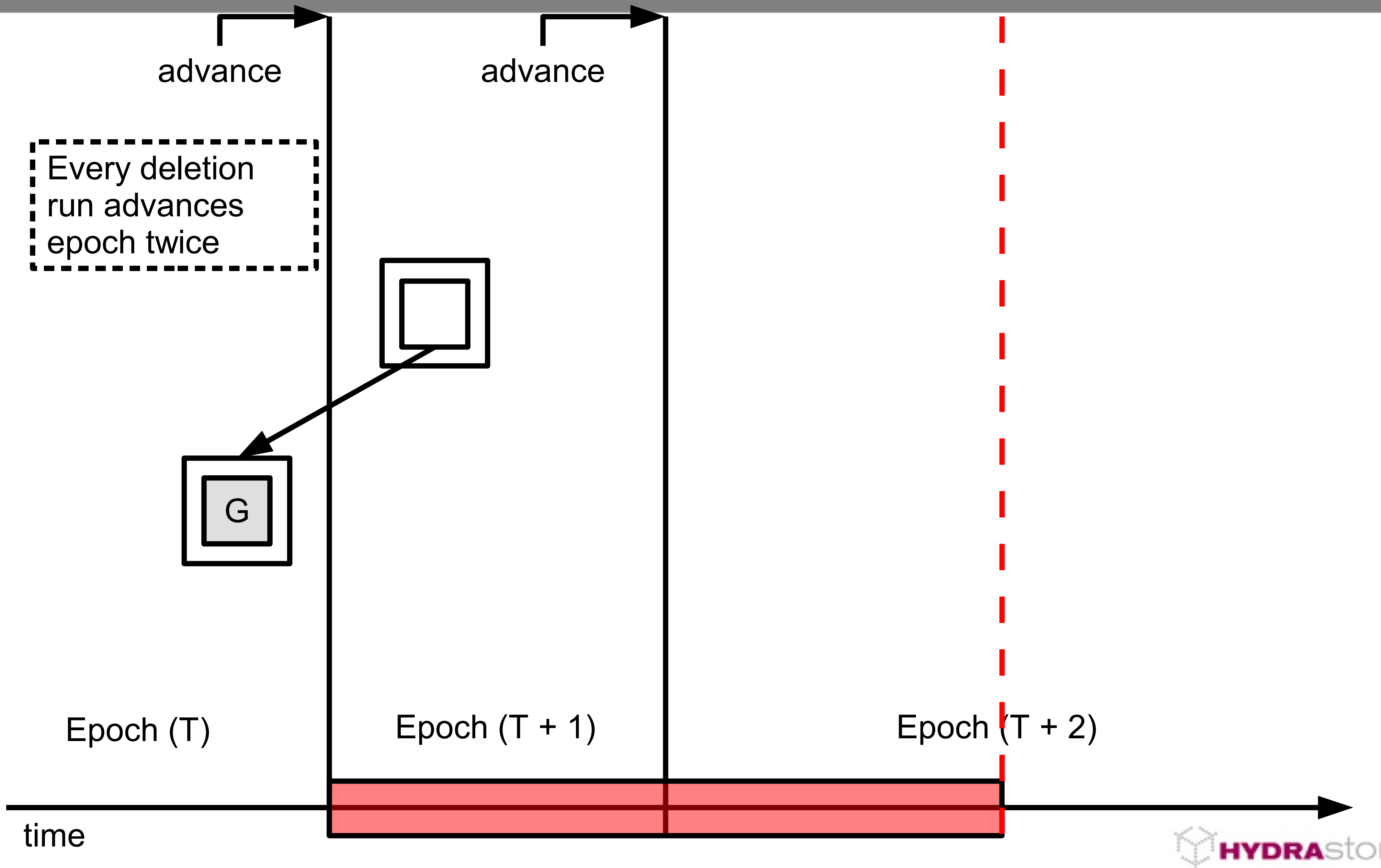
Supporting new writes during deletion



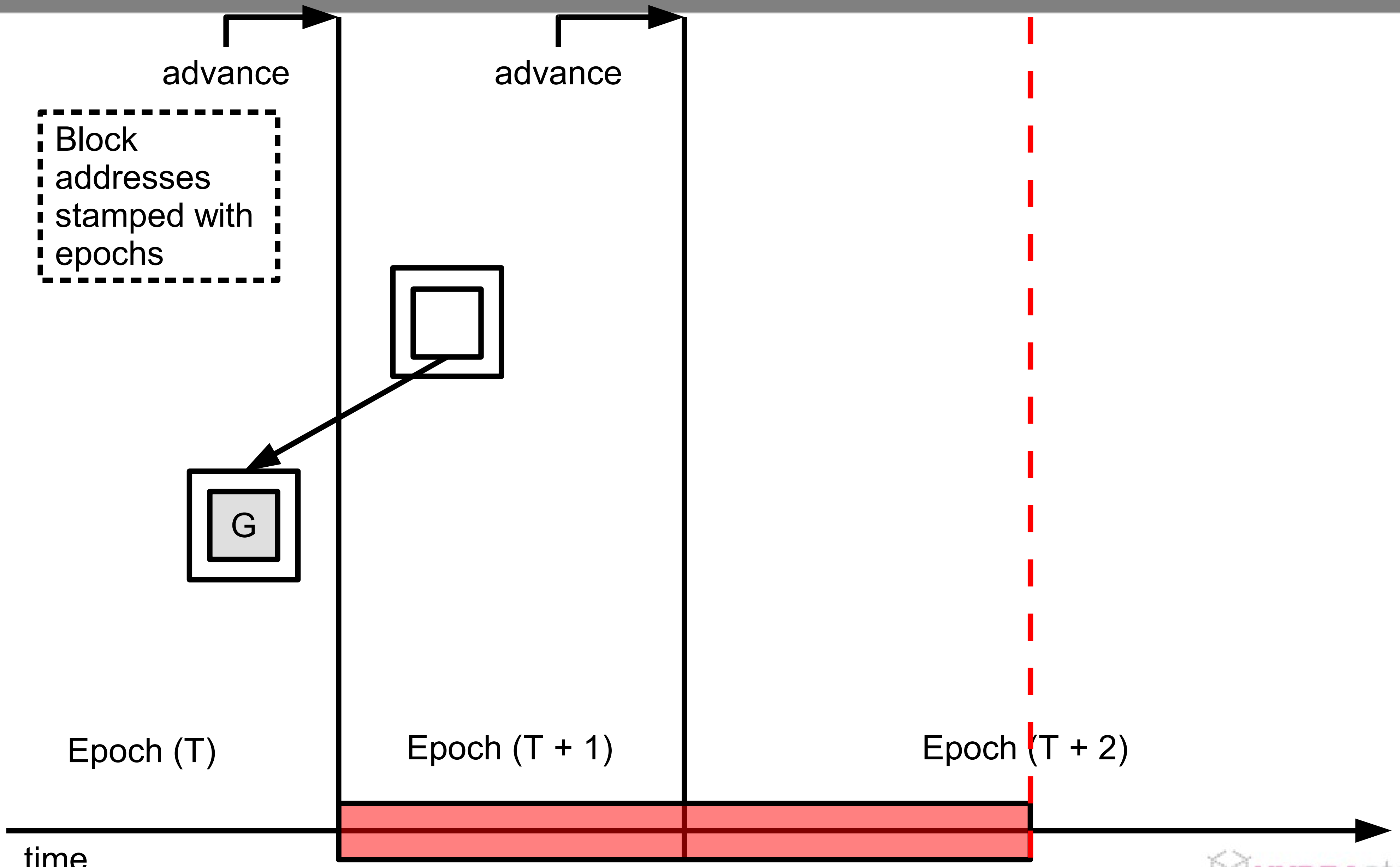
Supporting new writes during deletion



Supporting new writes during deletion

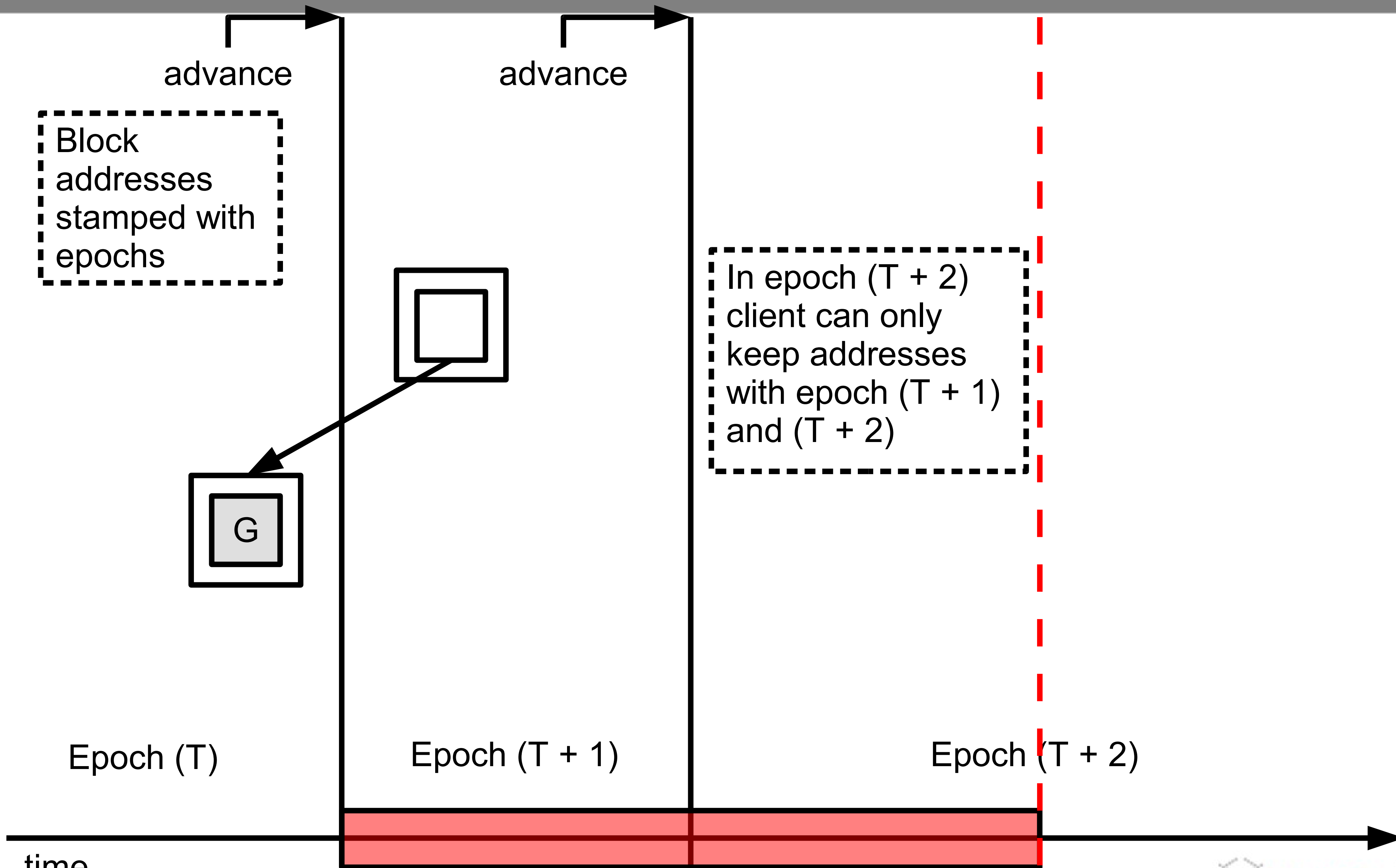


Supporting new writes during deletion

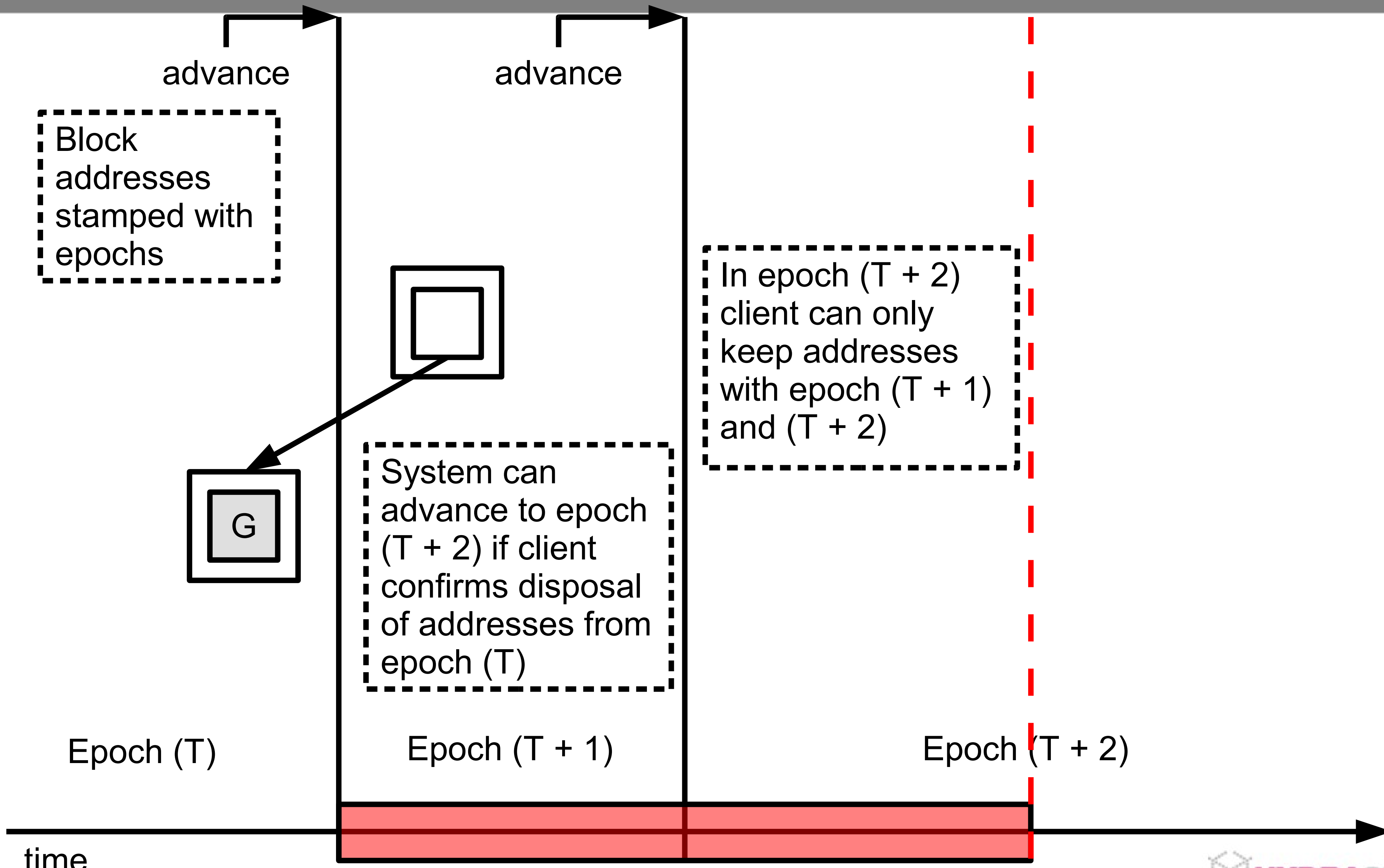


HYDRAstor

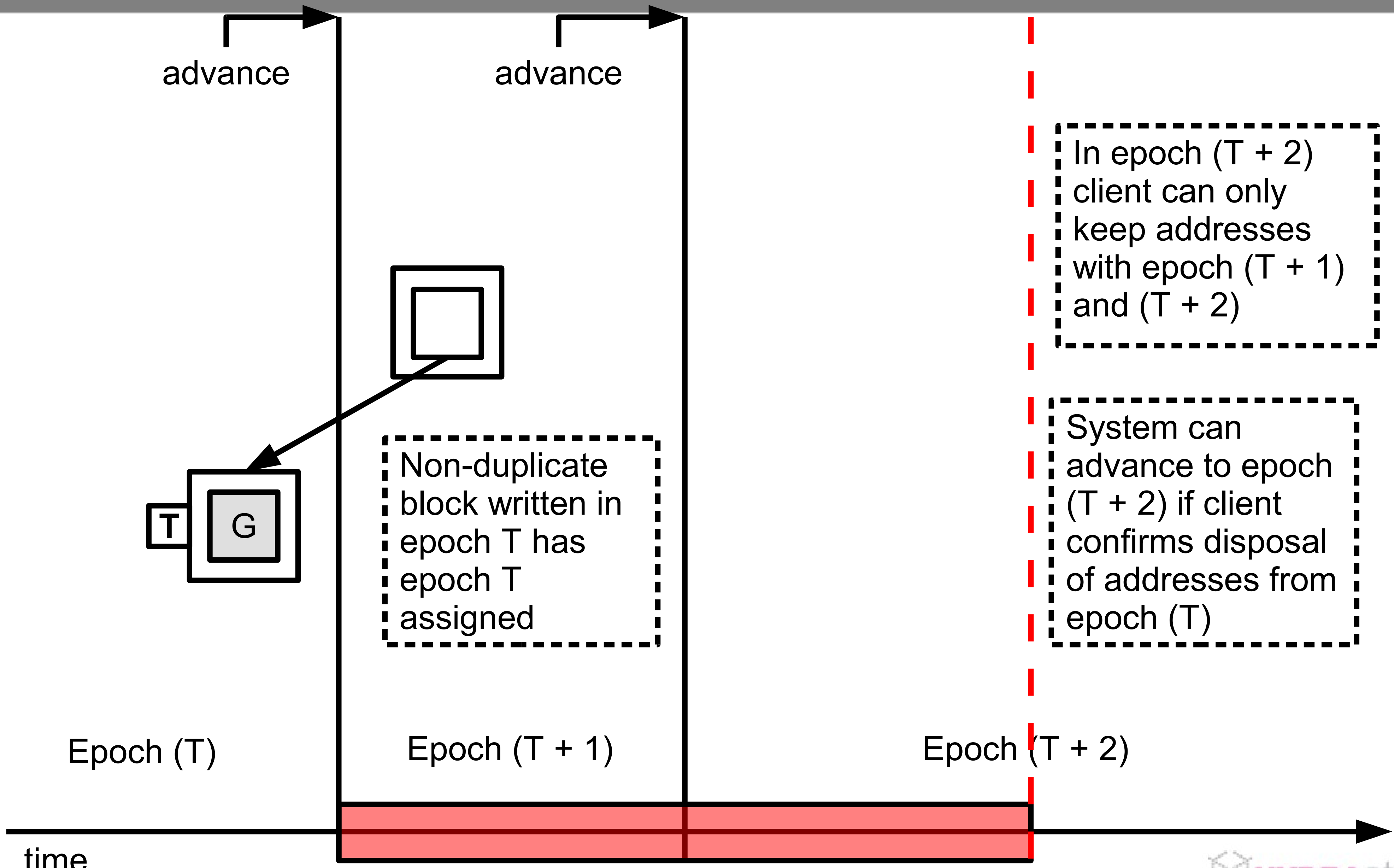
Supporting new writes during deletion



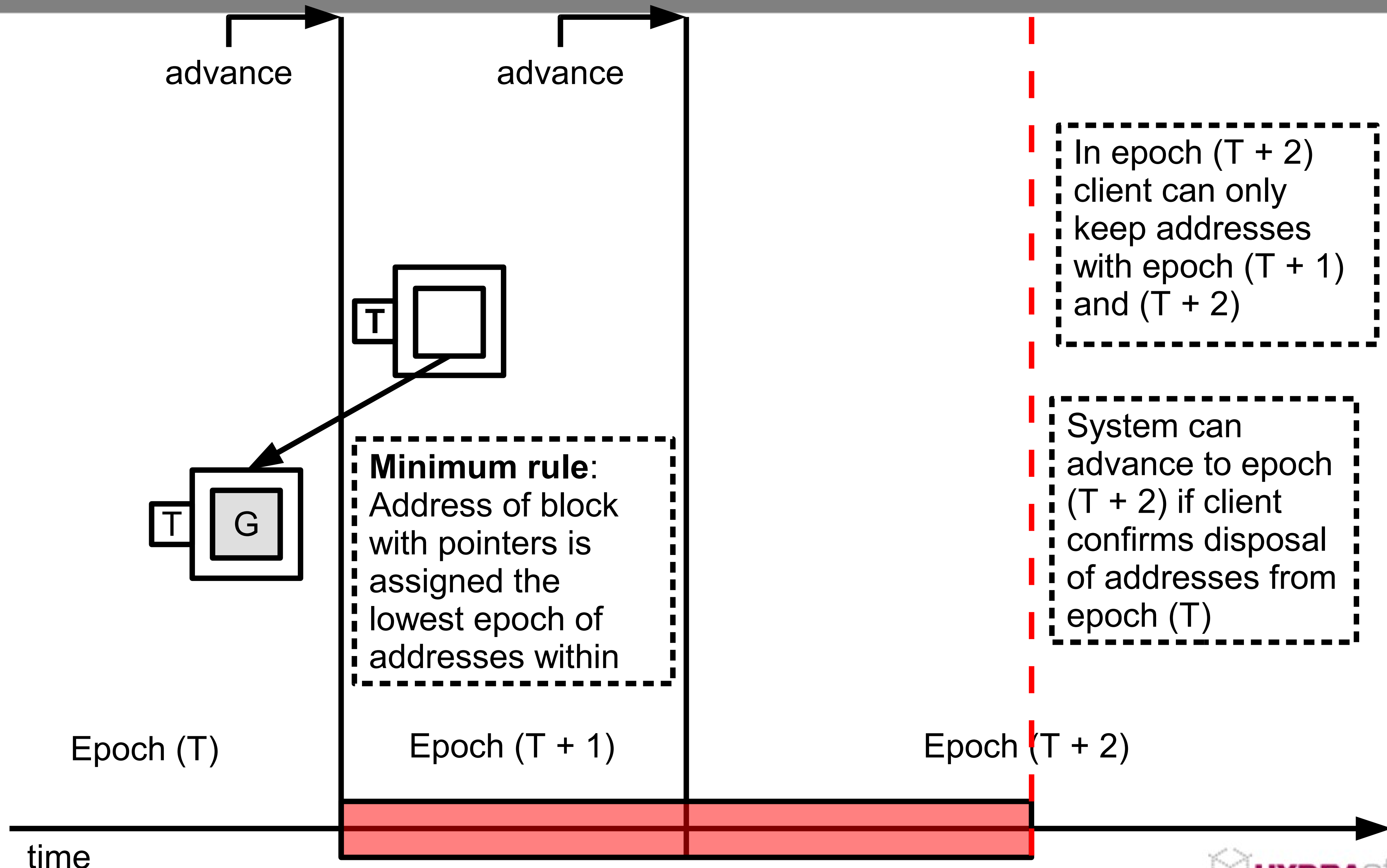
Supporting new writes during deletion



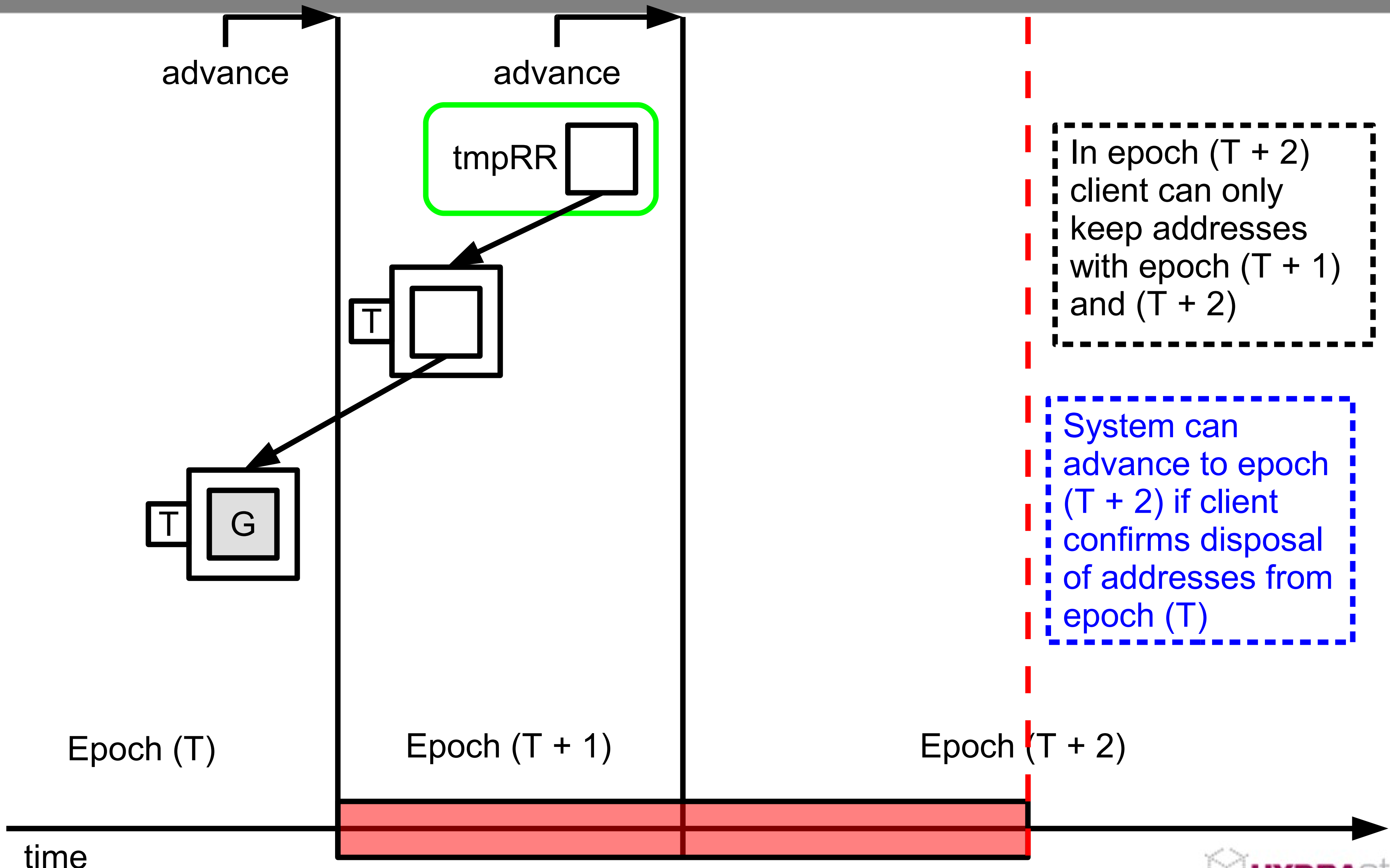
Supporting new writes during deletion



Supporting new writes during deletion

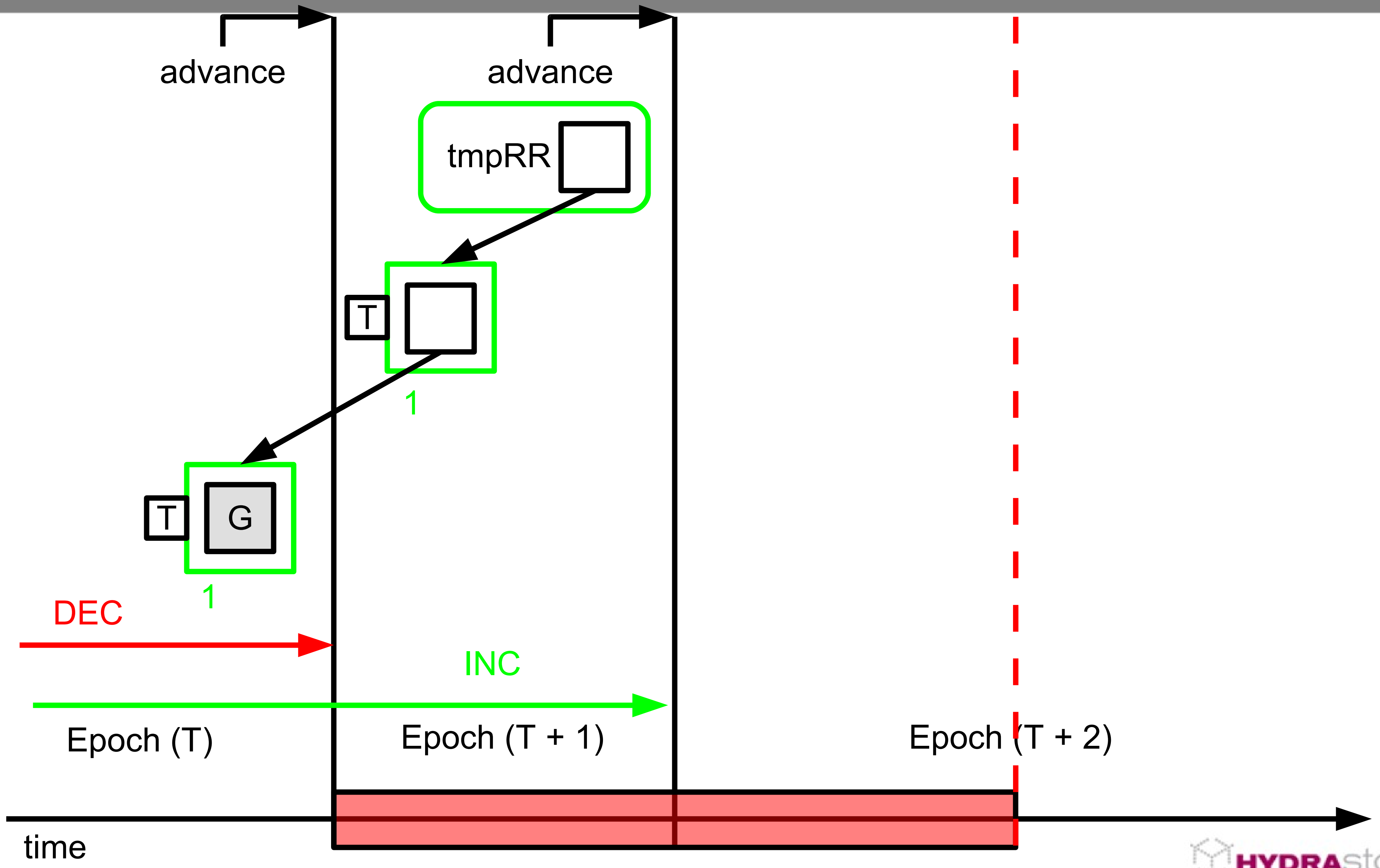


Supporting new writes during deletion

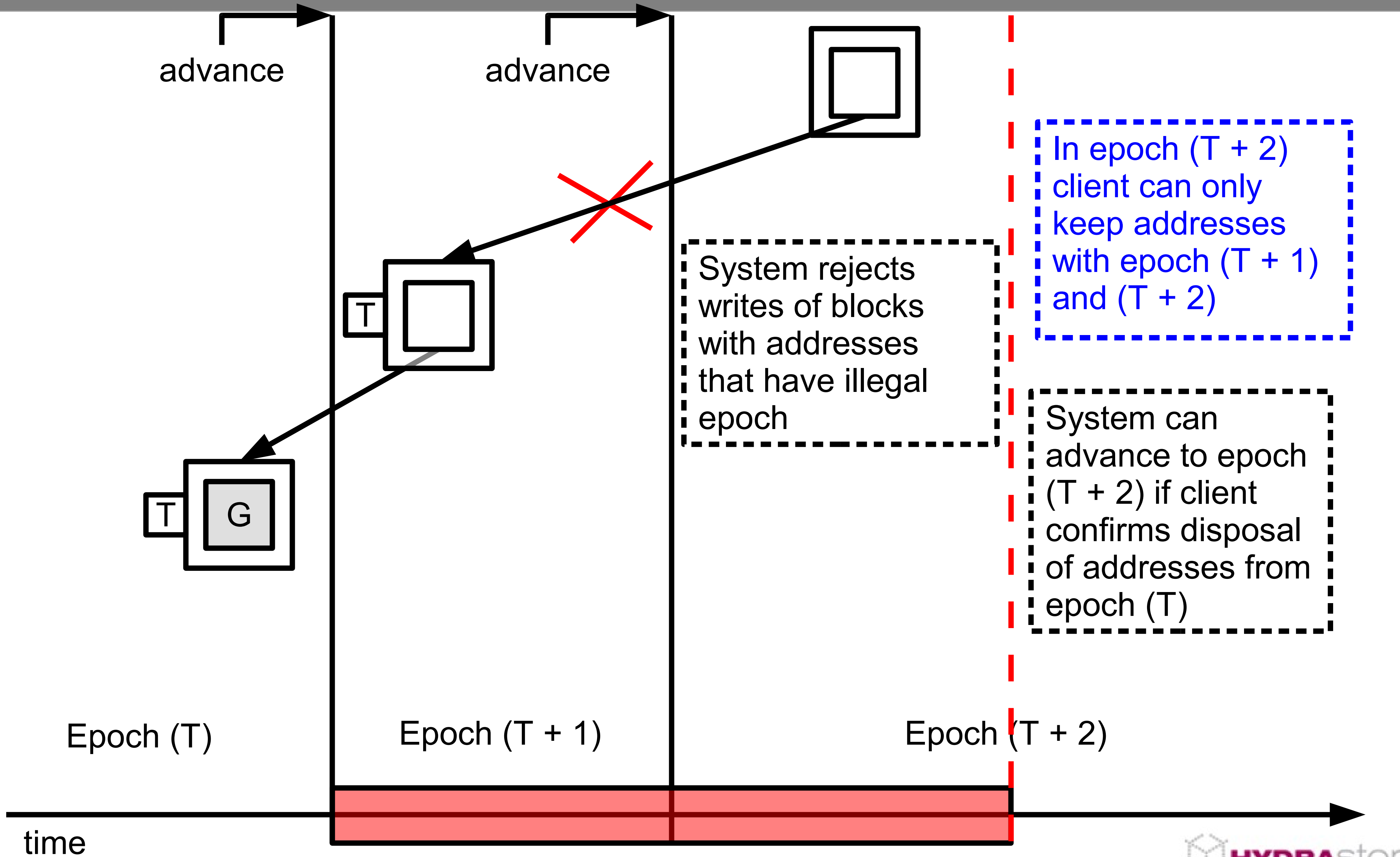


HYDRAstor

Supporting new writes during deletion

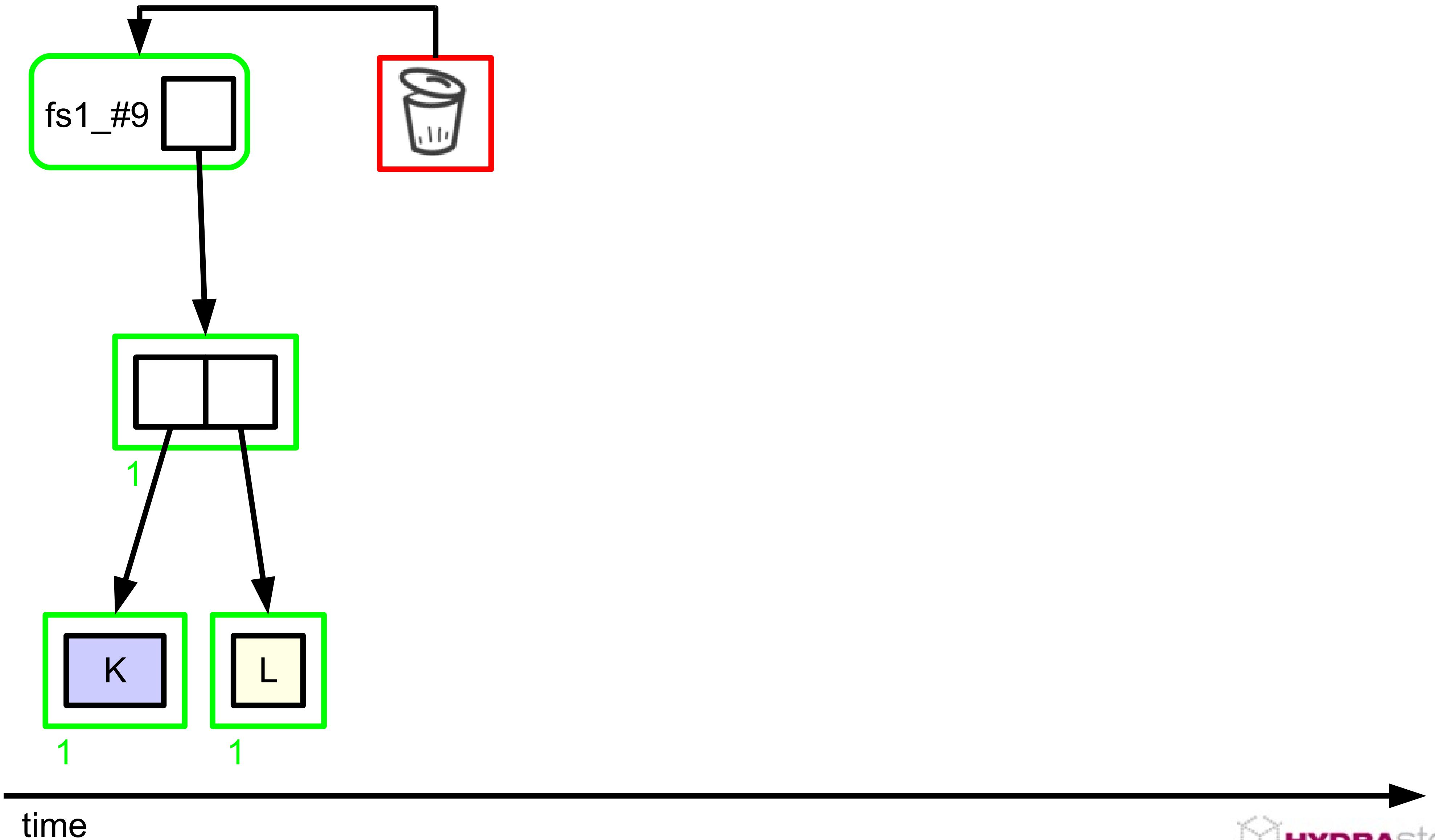


Supporting new writes during deletion

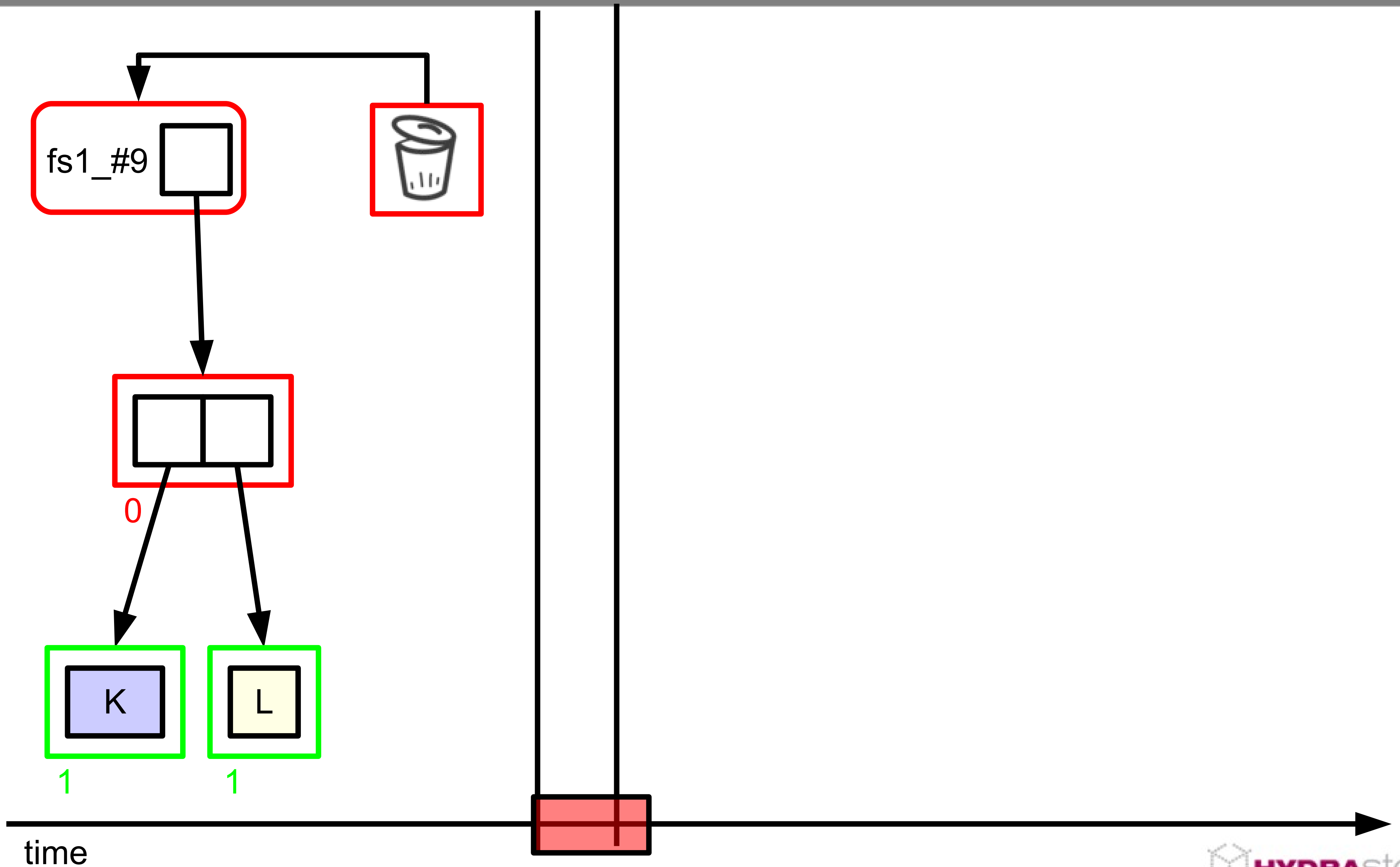


Supporting deduplication during deletion

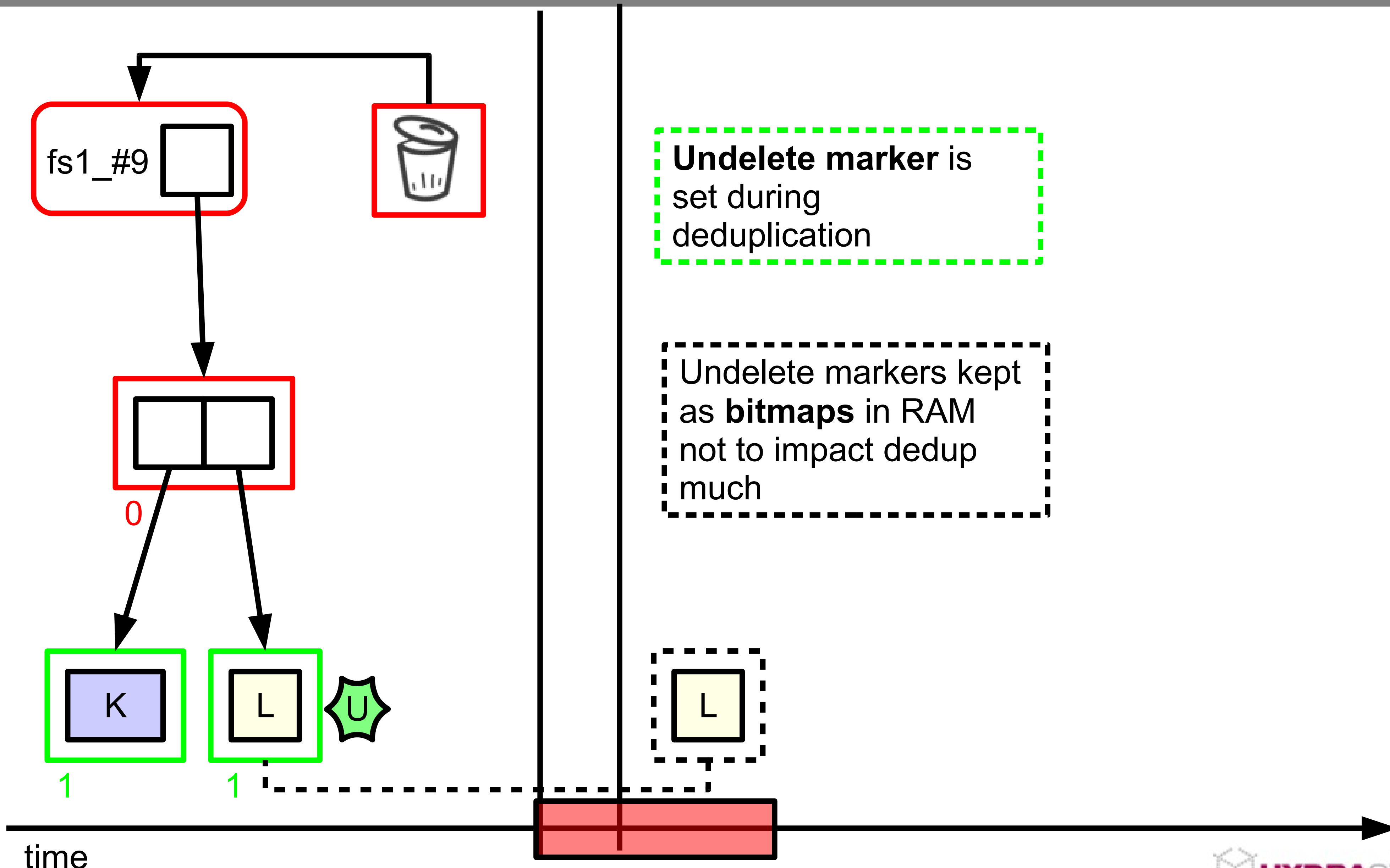
Supporting deduplication during deletion



Supporting deduplication during deletion



Supporting deduplication during deletion

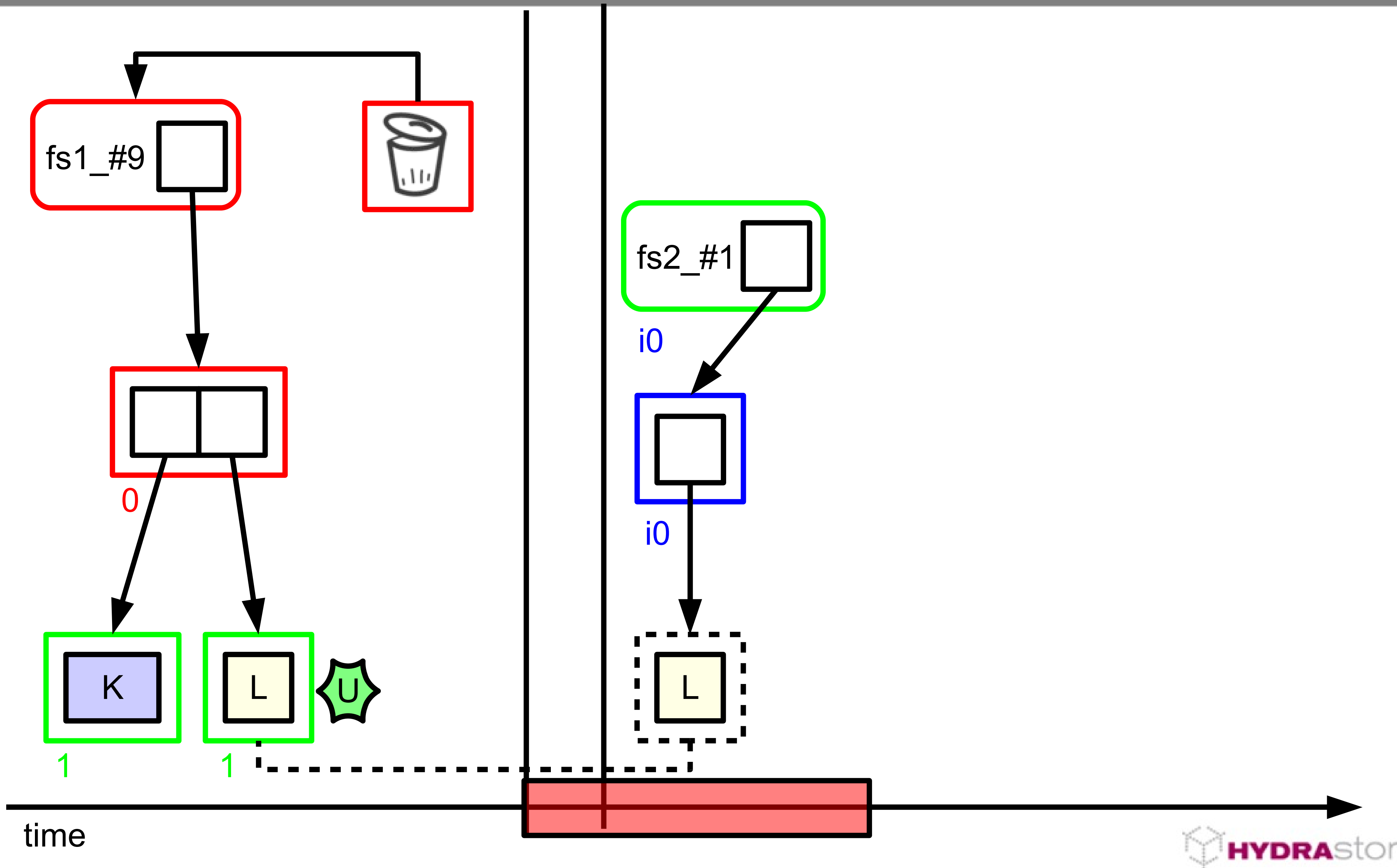


Undelete marker is set during deduplication

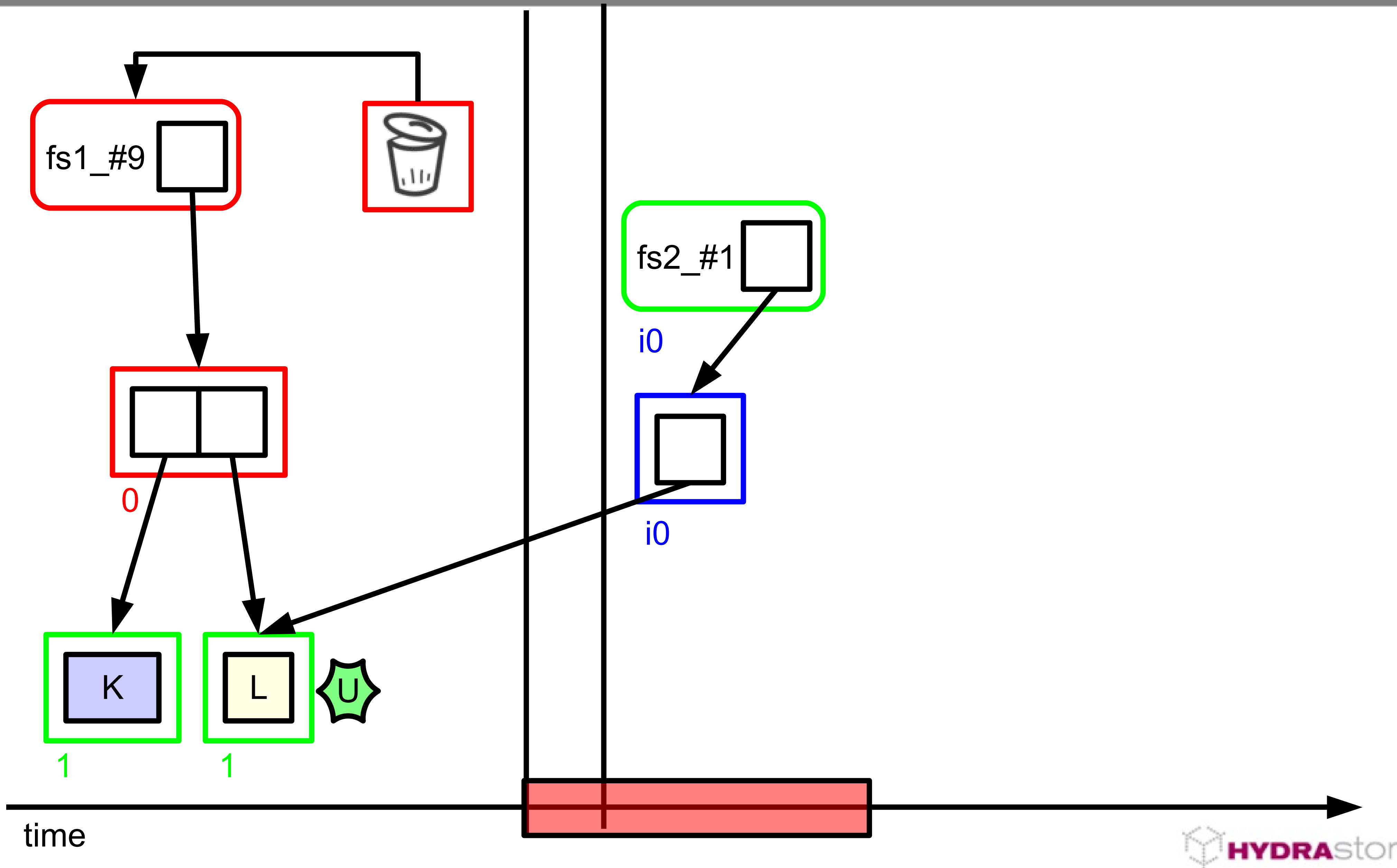
Undelete markers kept as **bitmaps** in RAM not to impact dedup much

HYDRAstor

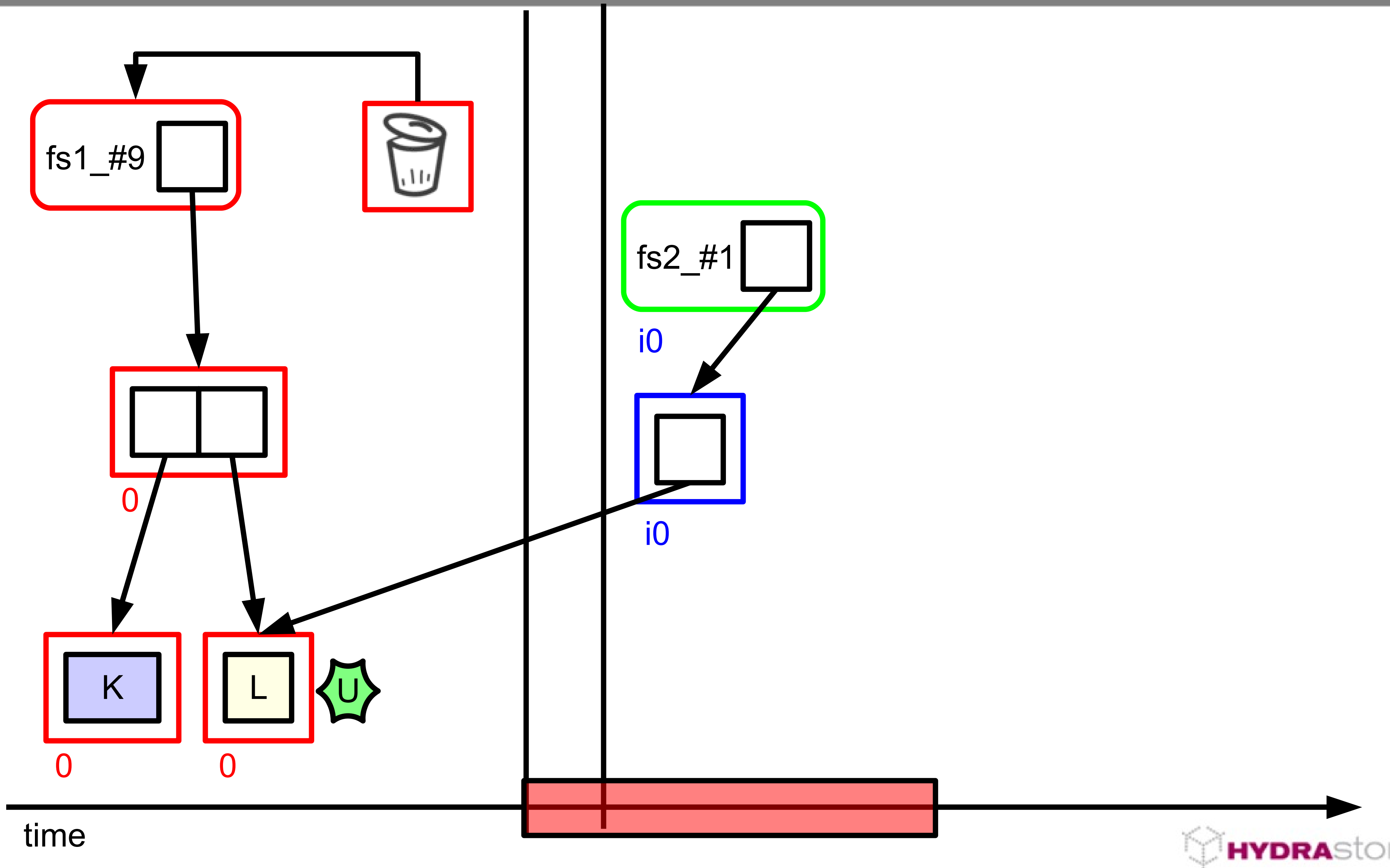
Supporting deduplication during deletion



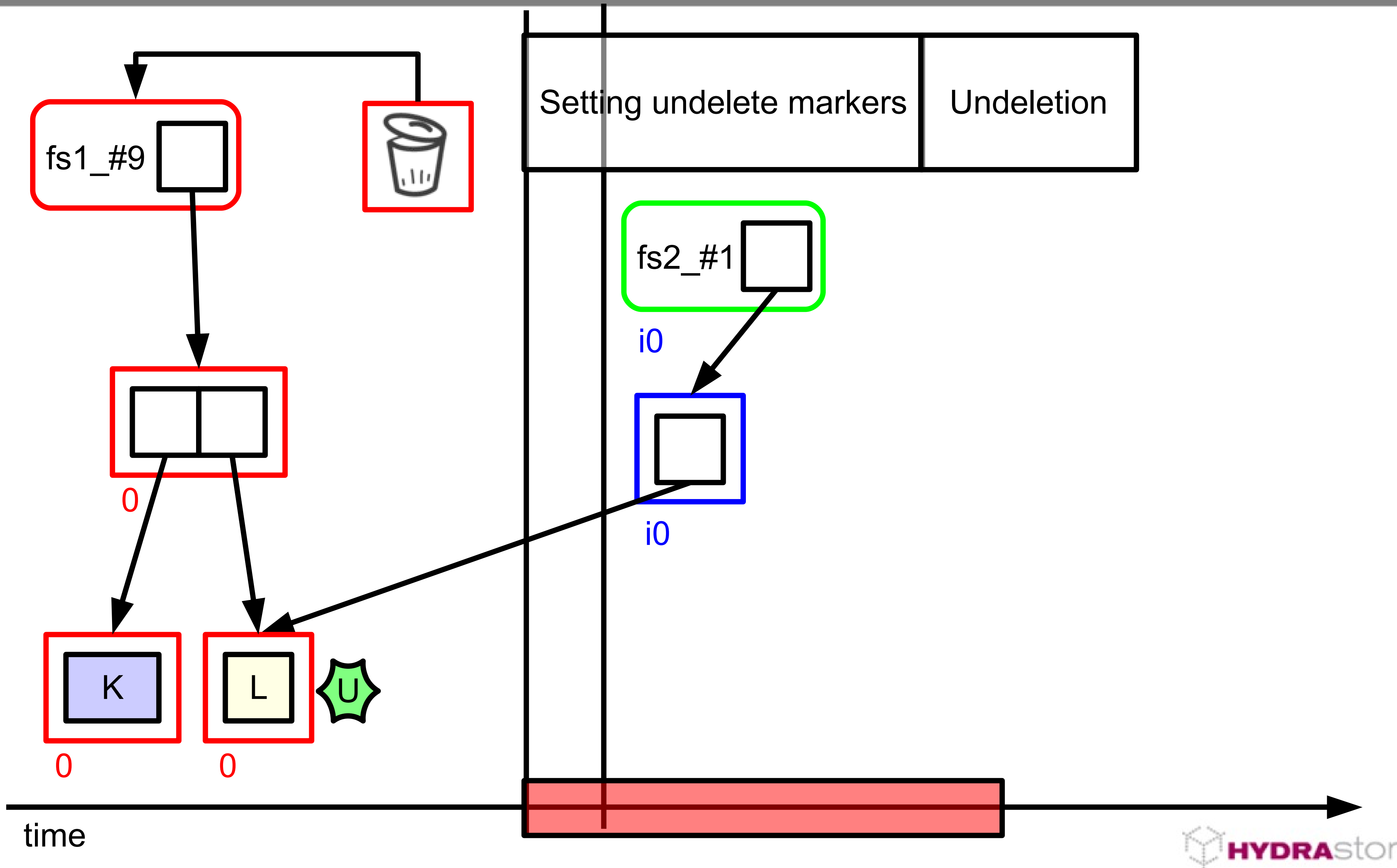
Supporting deduplication during deletion



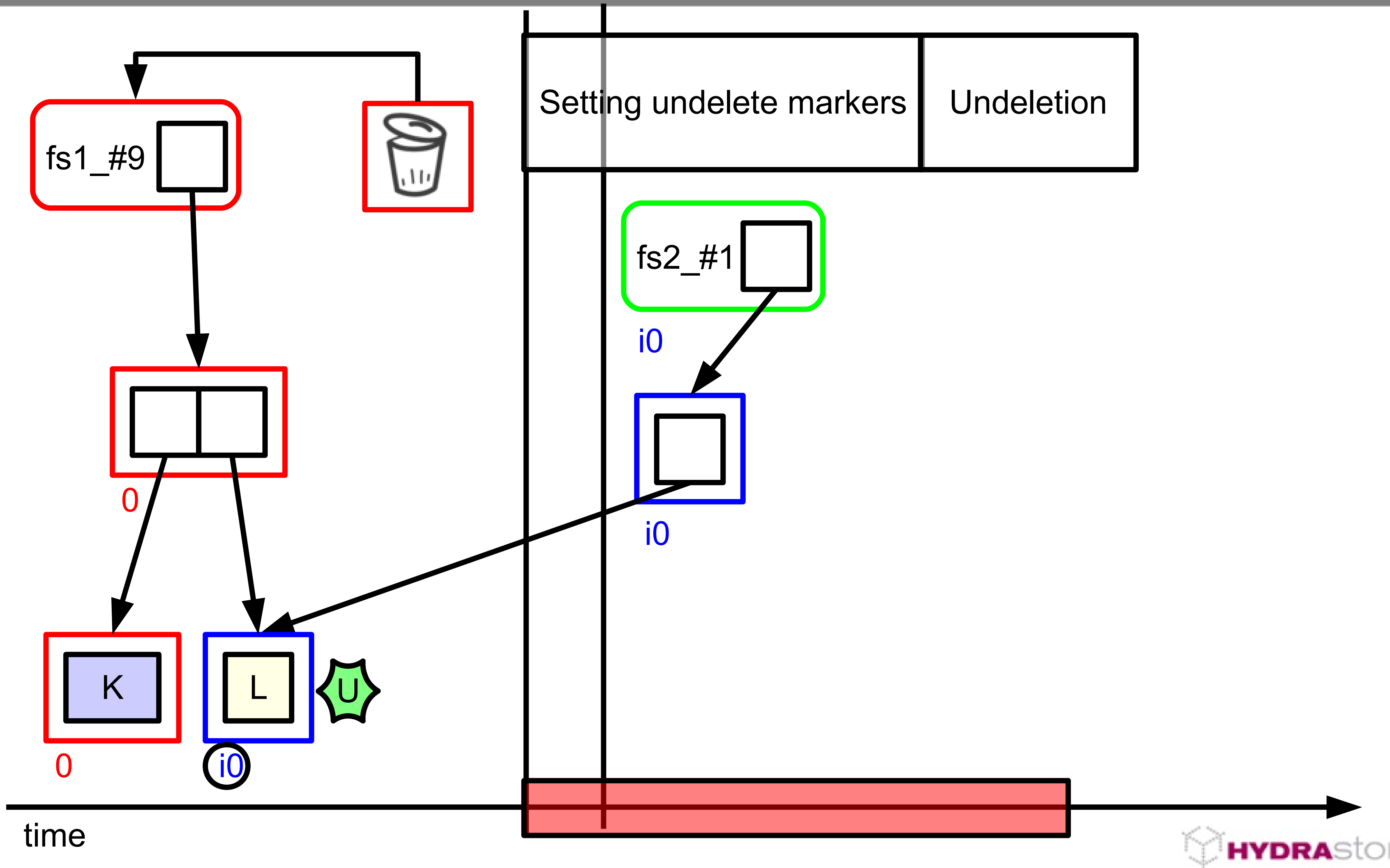
Supporting deduplication during deletion



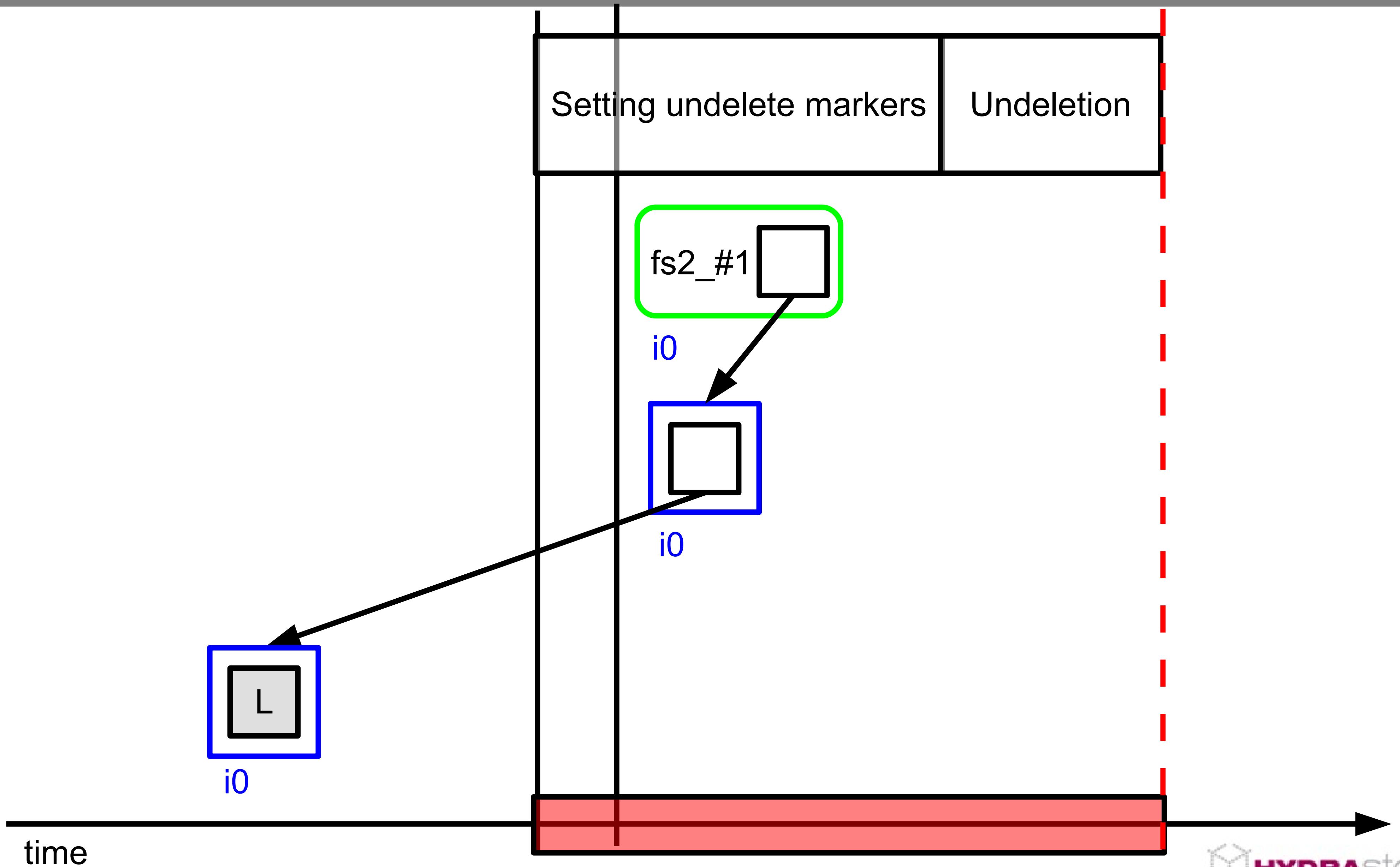
Supporting deduplication during deletion



Supporting deduplication during deletion



Supporting deduplication during deletion



Deletion in a distributed architecture

New challenges introduced by distribution

- Assumptions:
  - block data distributed
- Dynamic configuration
  - nodes can be added/deleted
- Consistency of blocks preservation/removal
- Handling failures
  - detached nodes with obsolete counters
  - continue on failure of some nodes

Key techniques to support distribution

- Continue on failure of some nodes
  - redundancy of counter computation
  - redundancy of undelete markers

- Recognize obsolete counters
  - stamping counters with epochs
  - counter epochs refreshed when deletion ends

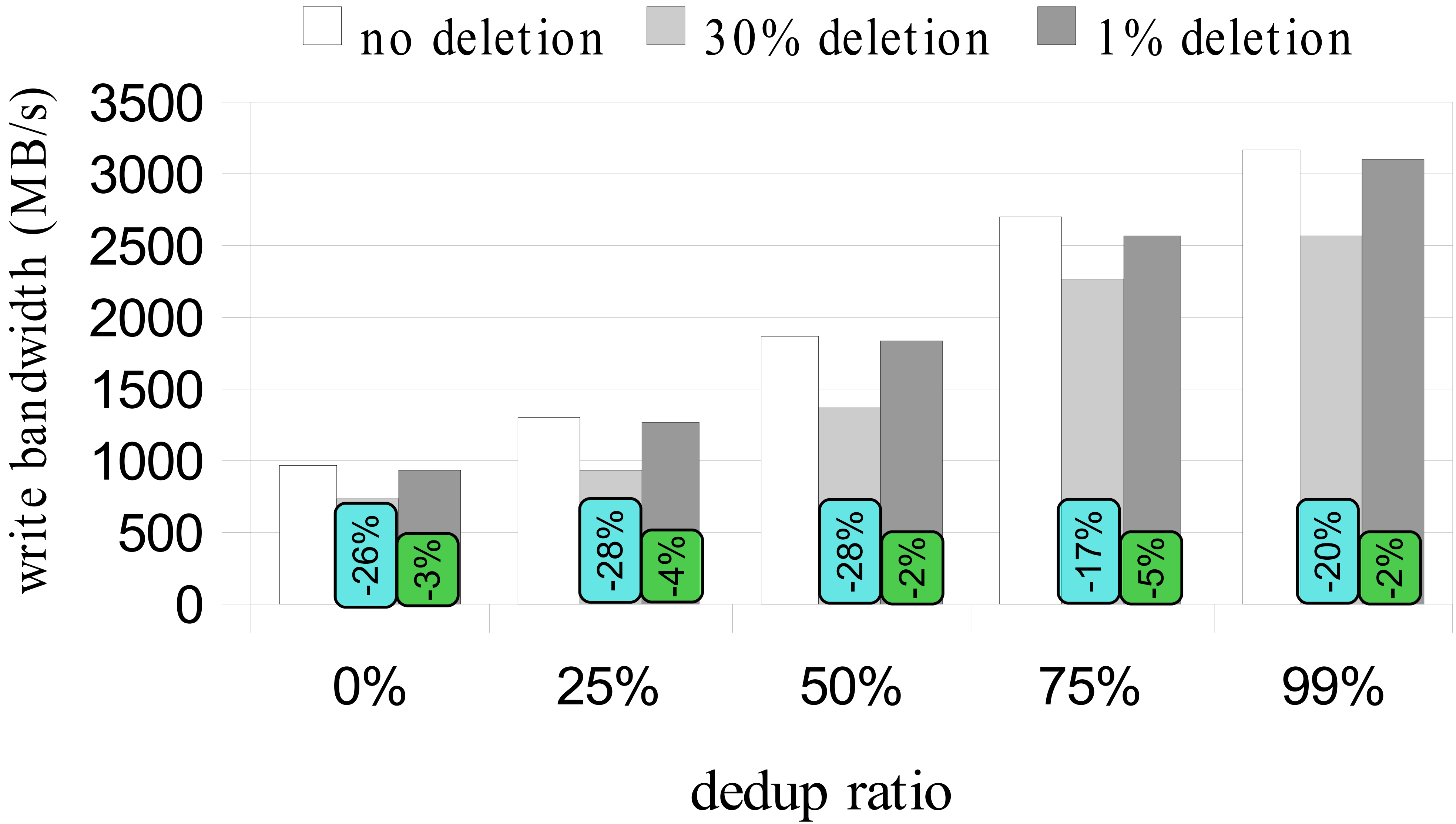
HYDRAstor

Implementation in HYDRAstor

Performance evaluation in HYDRAstor

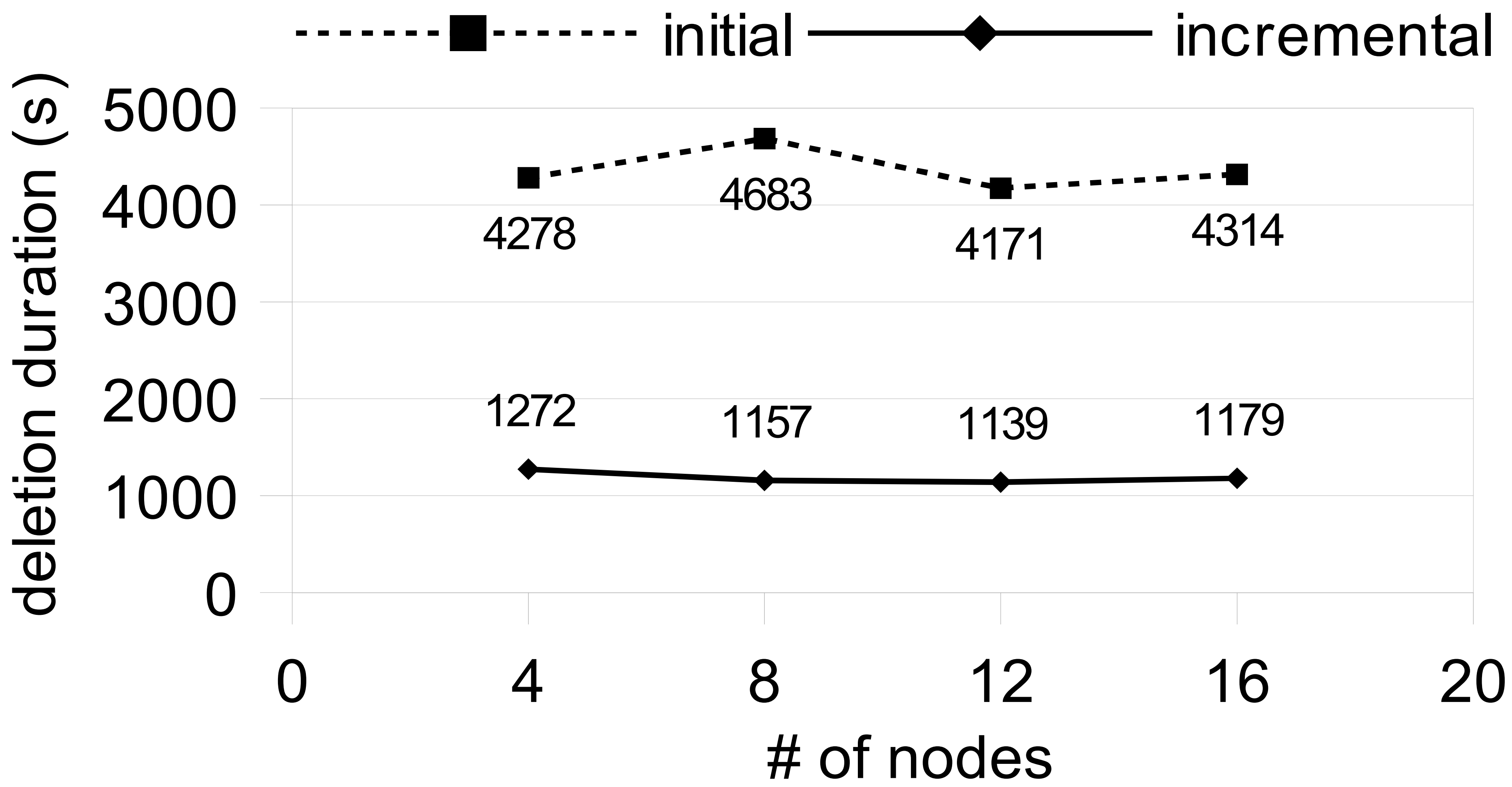
- Configurable resource division
  - default: 30% deletion, 70% user operations
  - minimal: less than 1% for deletion
- Experiments used 4 to 16 machines
  - 2^{nd} or 3^{rd} generation HYDRAstor servers
    - 2 quad core 2.4Ghz CPU
    - 12x1TB SATA disks
    - 24GB RAM
    - 4 GigE cards
- Measurements
  - bandwidth of reads and writes during deletion
  - deletion duration

Impact of garbage identification on writes



HYDRAstor

Scalability



Conclusions

Conclusions

- Problem
  - Deletion in distributed CAS systems
- Solution
  - concurrent deletion algorithm
  - writes with dedup enabled during deletion
  - failure tolerance
  - scalability
- Key techniques
  - epochs
  - undelete markers
- Implementation in HYDRAstor
  - fairly low performance impact

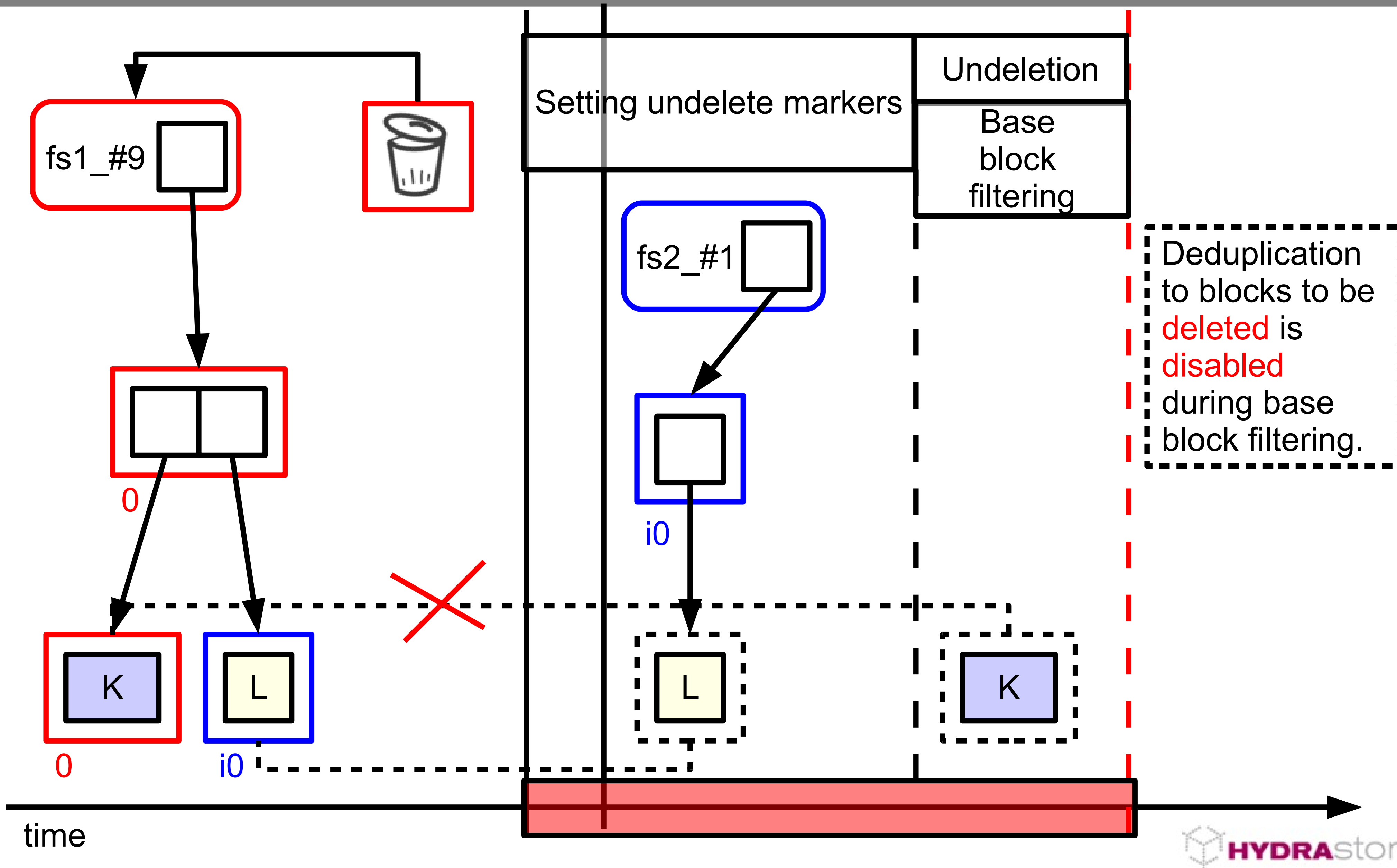
Questions?

strzelczak@9livesdata.com

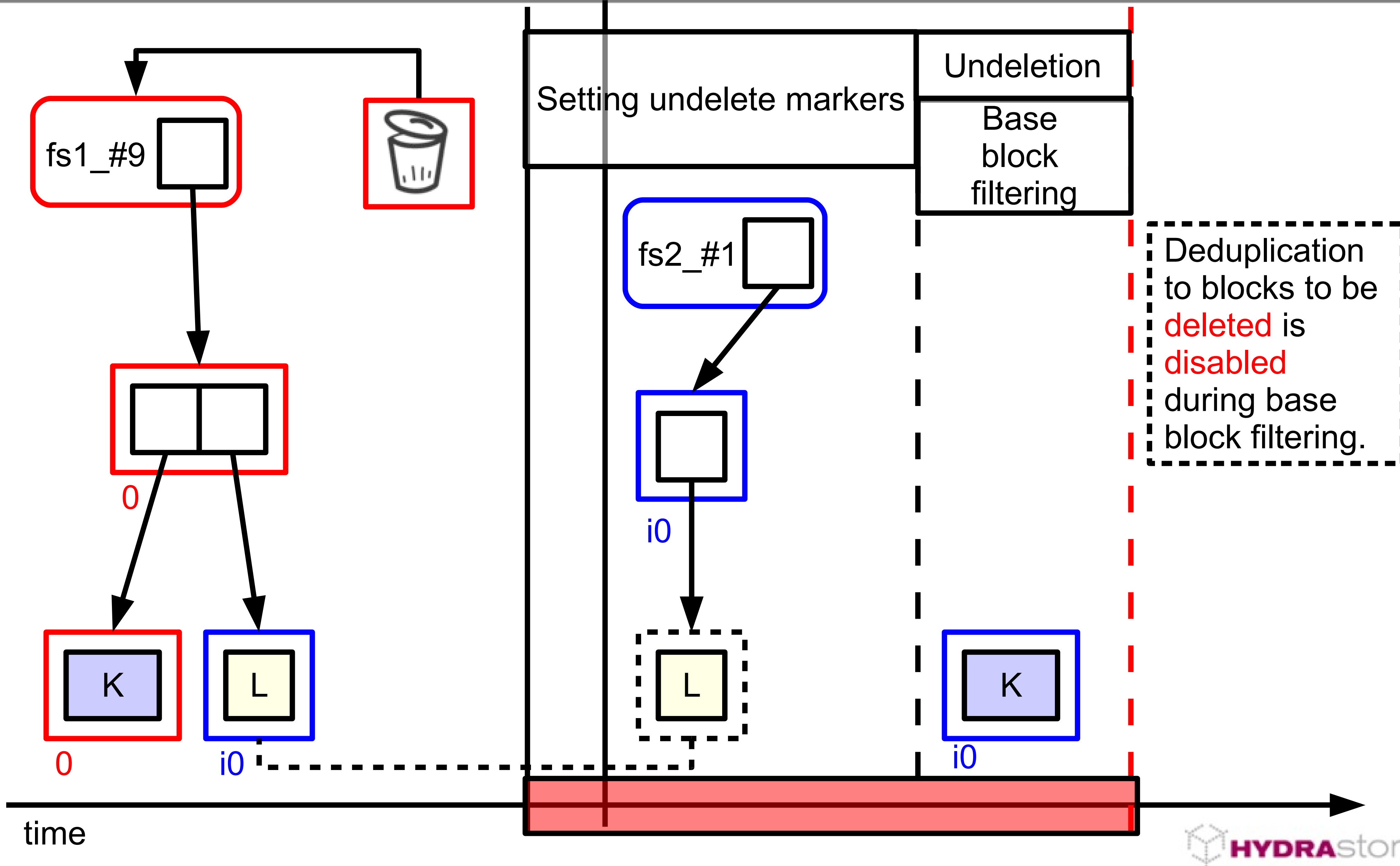
Appendix

Deduplication support during undeletion

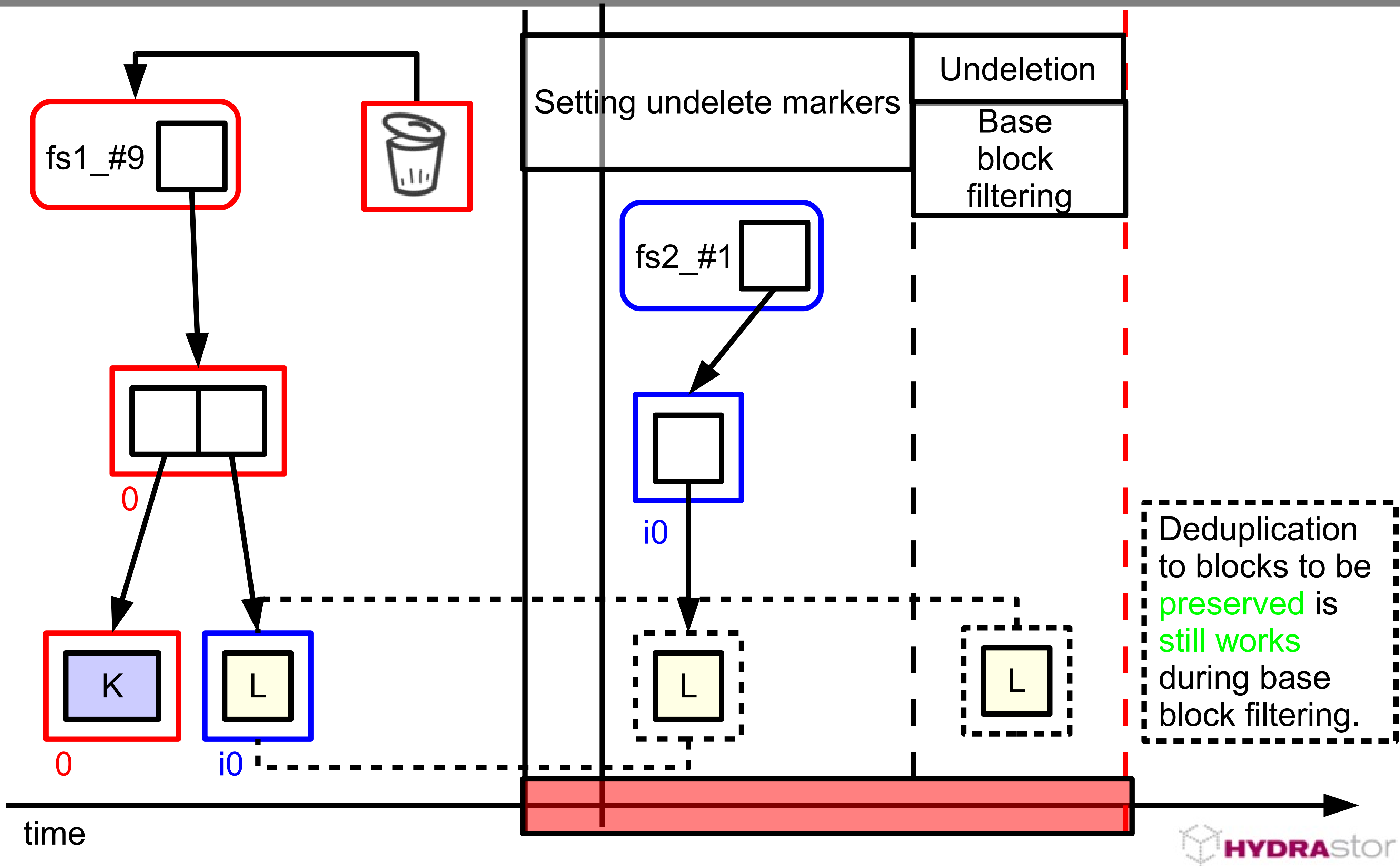
Supporting deduplication during deletion



Supporting deduplication during deletion

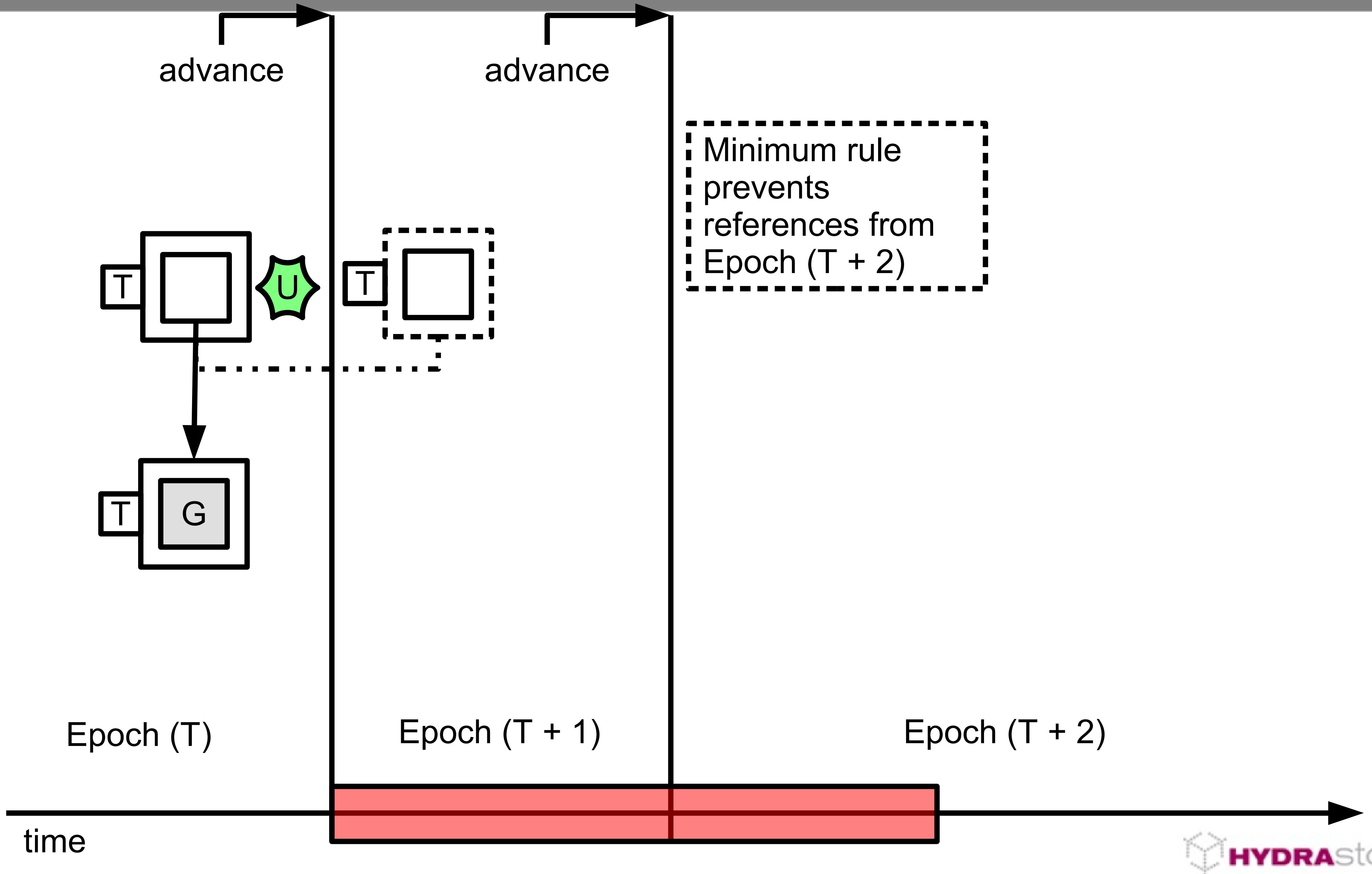


Supporting deduplication during deletion

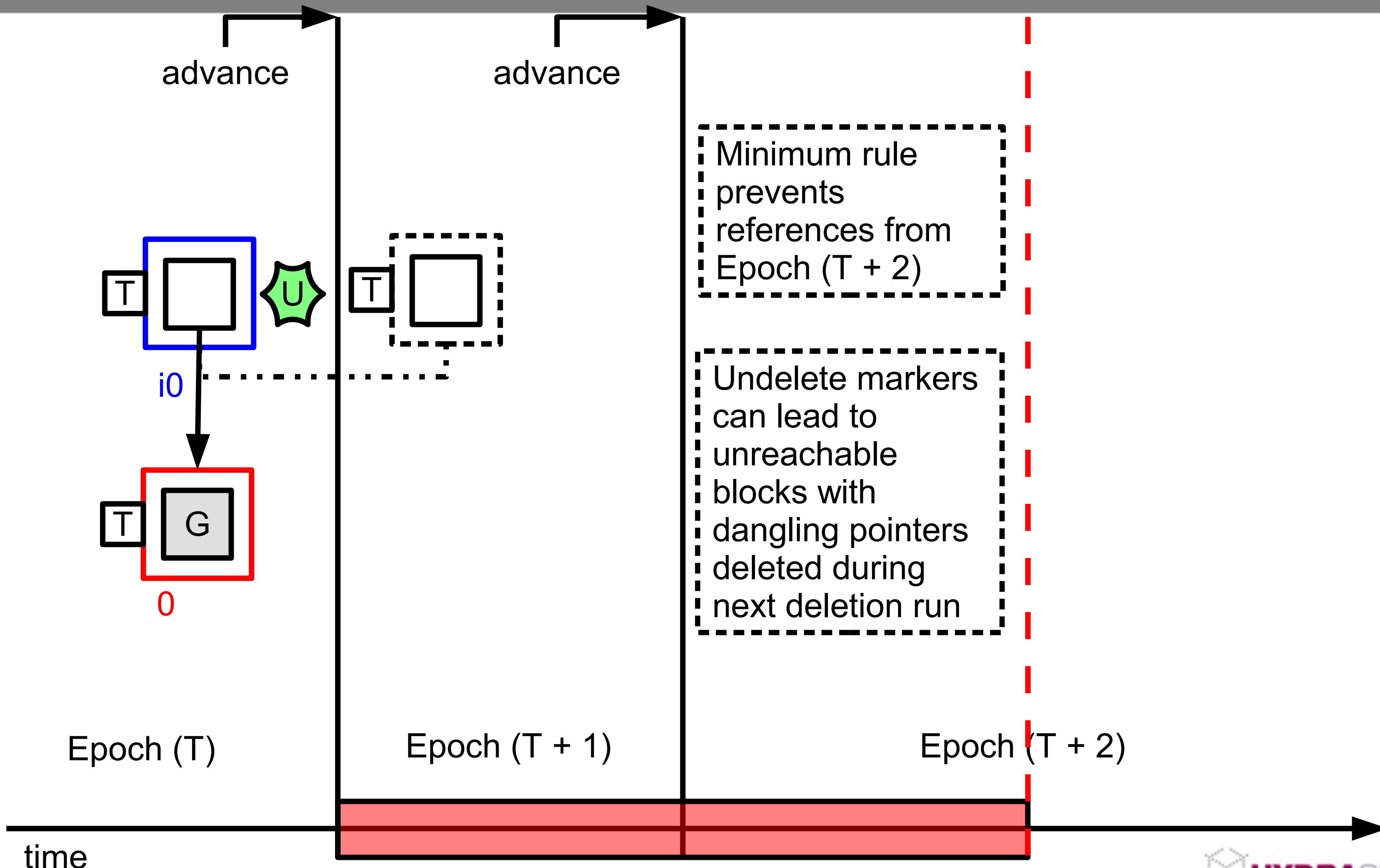


Deduplication of blocks with pointers during deletion

Deduplication of blocks with pointers durig deletion



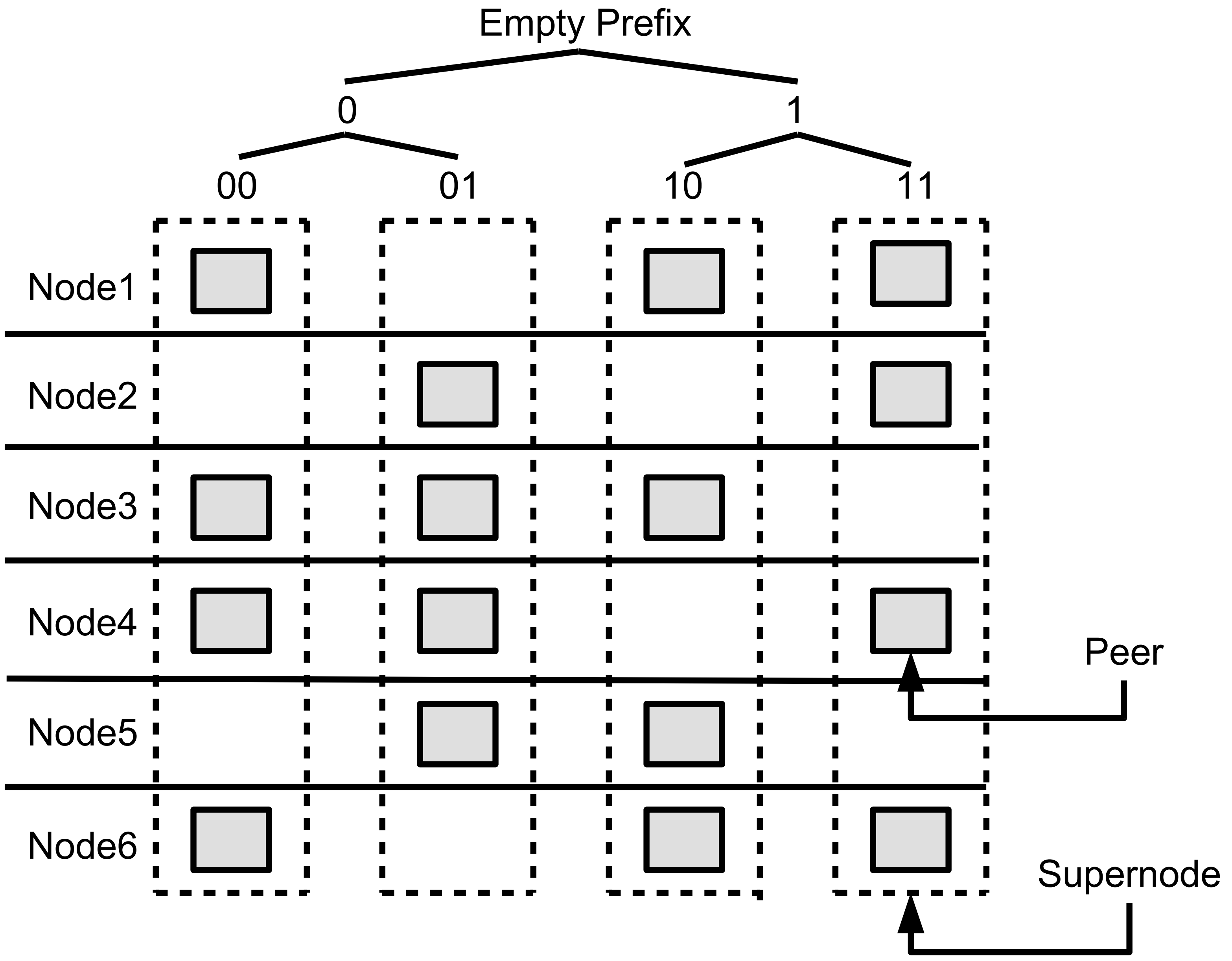
Deduplication of blocks with pointers durig deletion



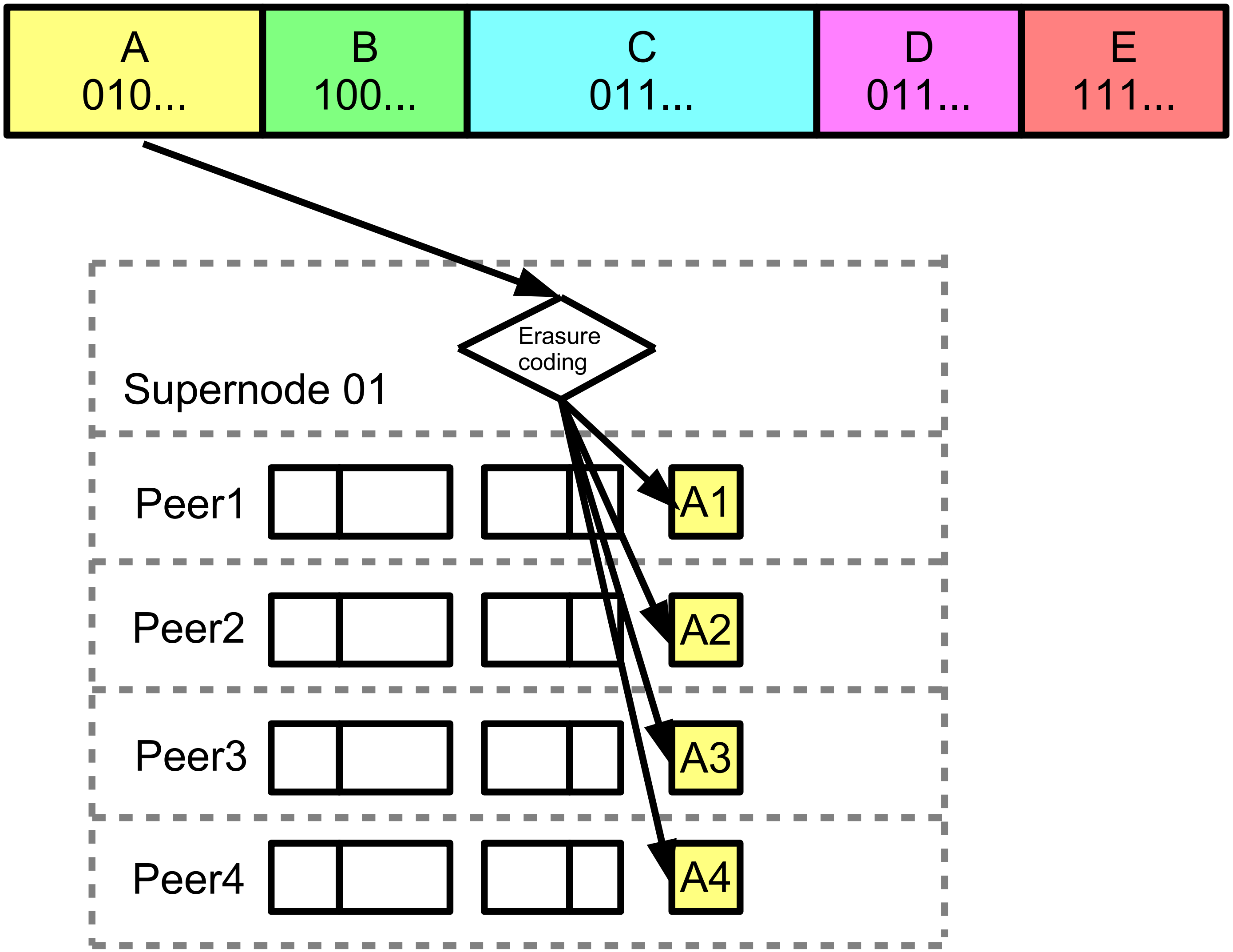
HYDRAstor

Distributed architecture

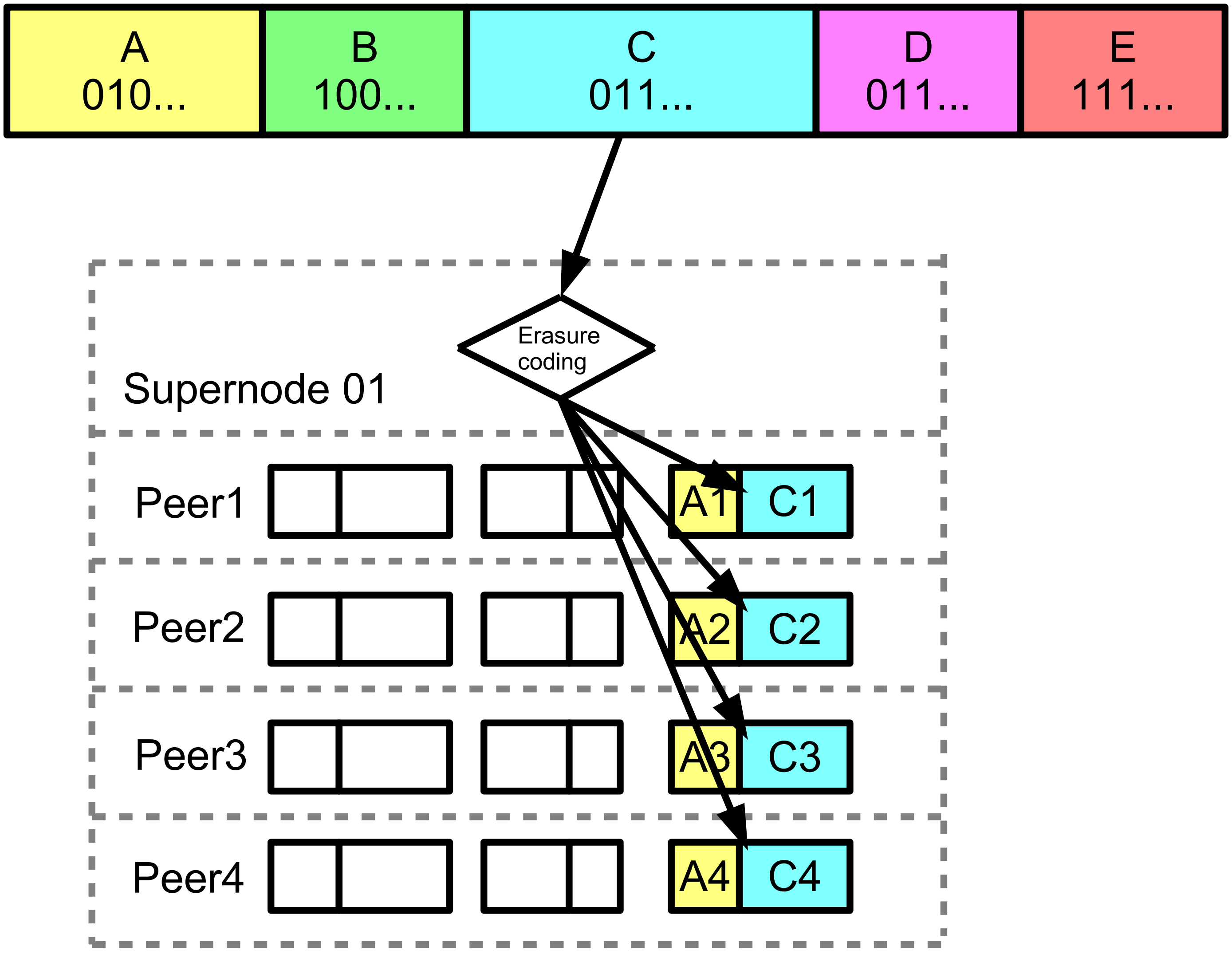
DHT, supernodes, peers



Container based organization

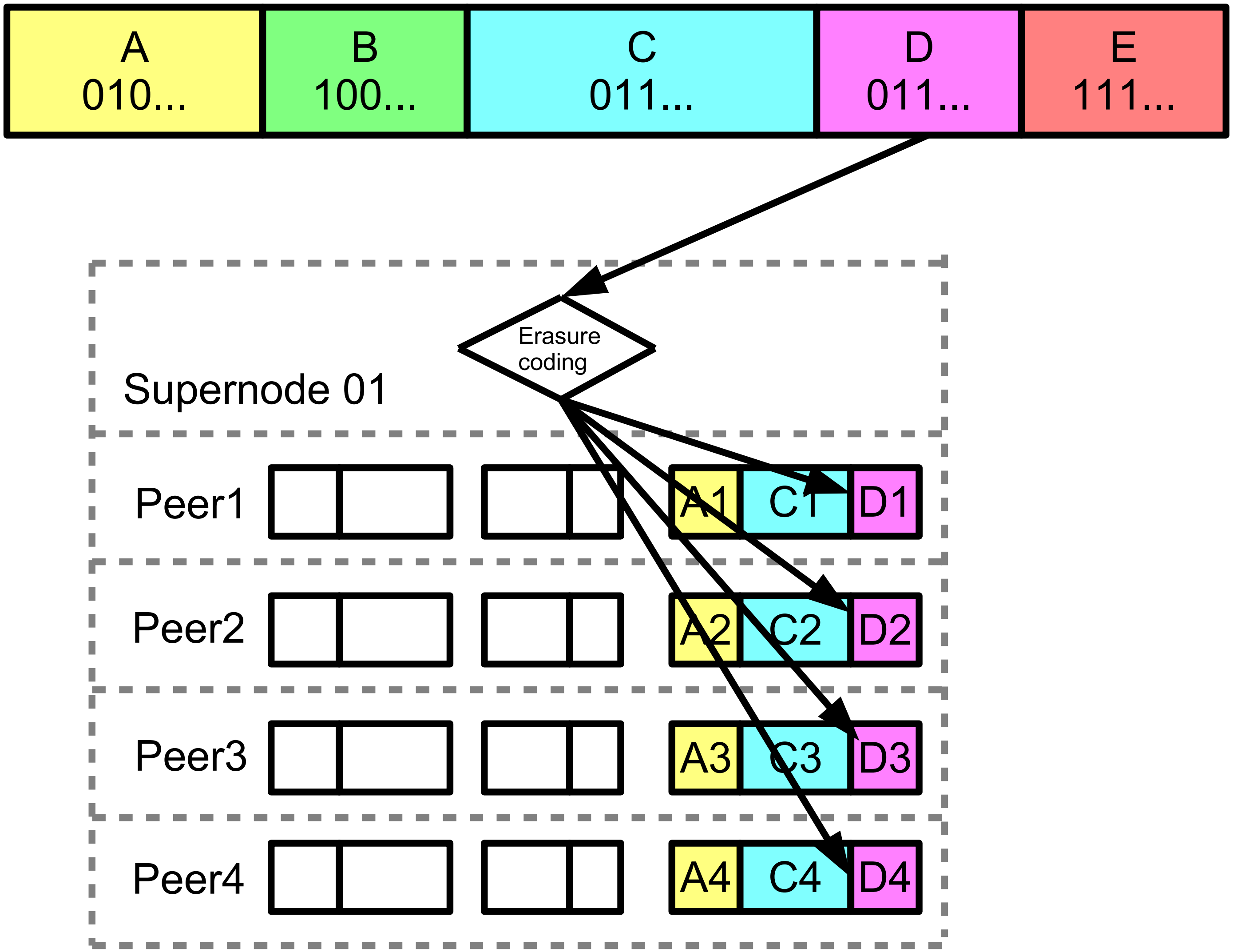


Container based data organization

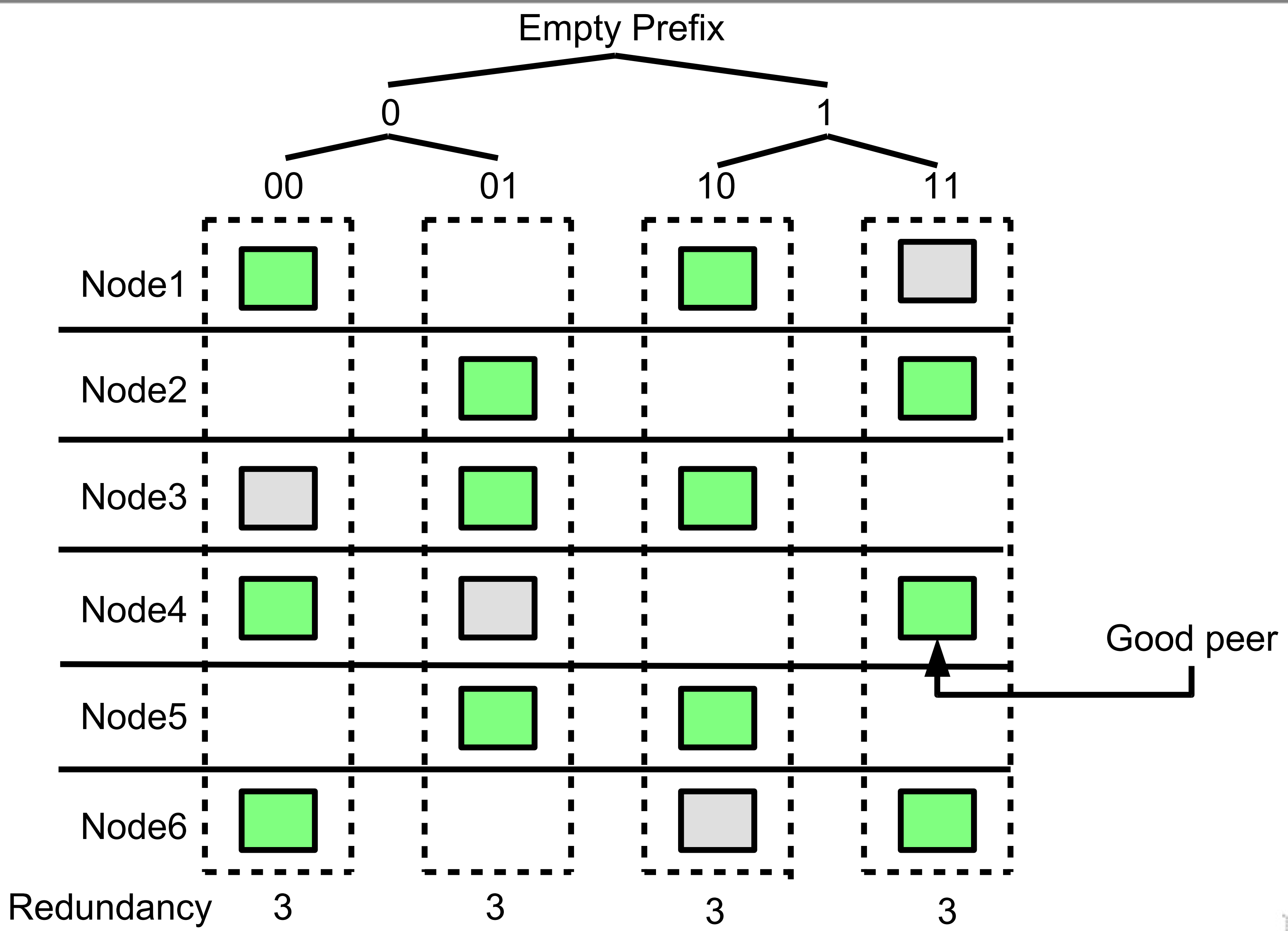


HYDRAstor

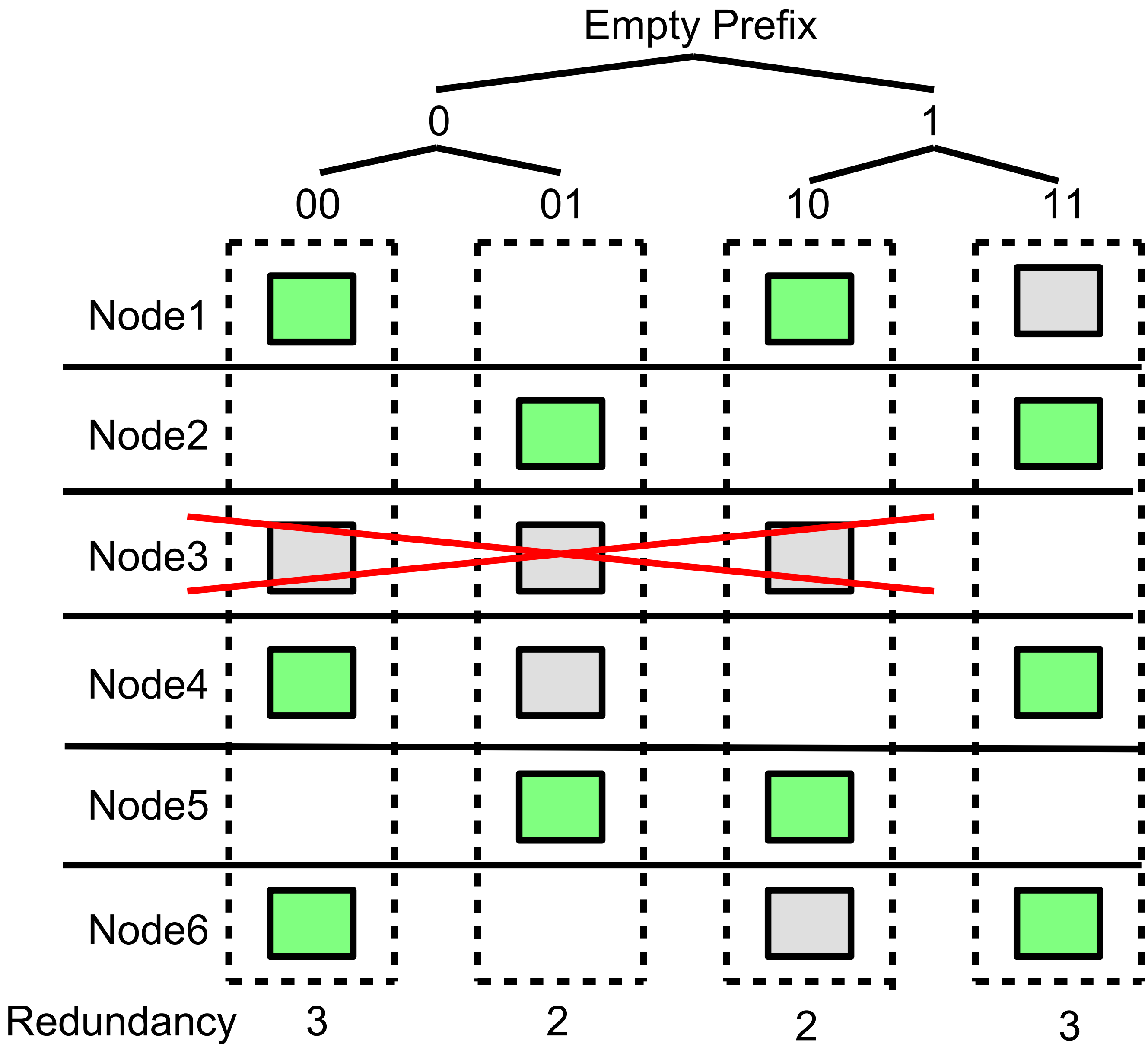
Container based organization



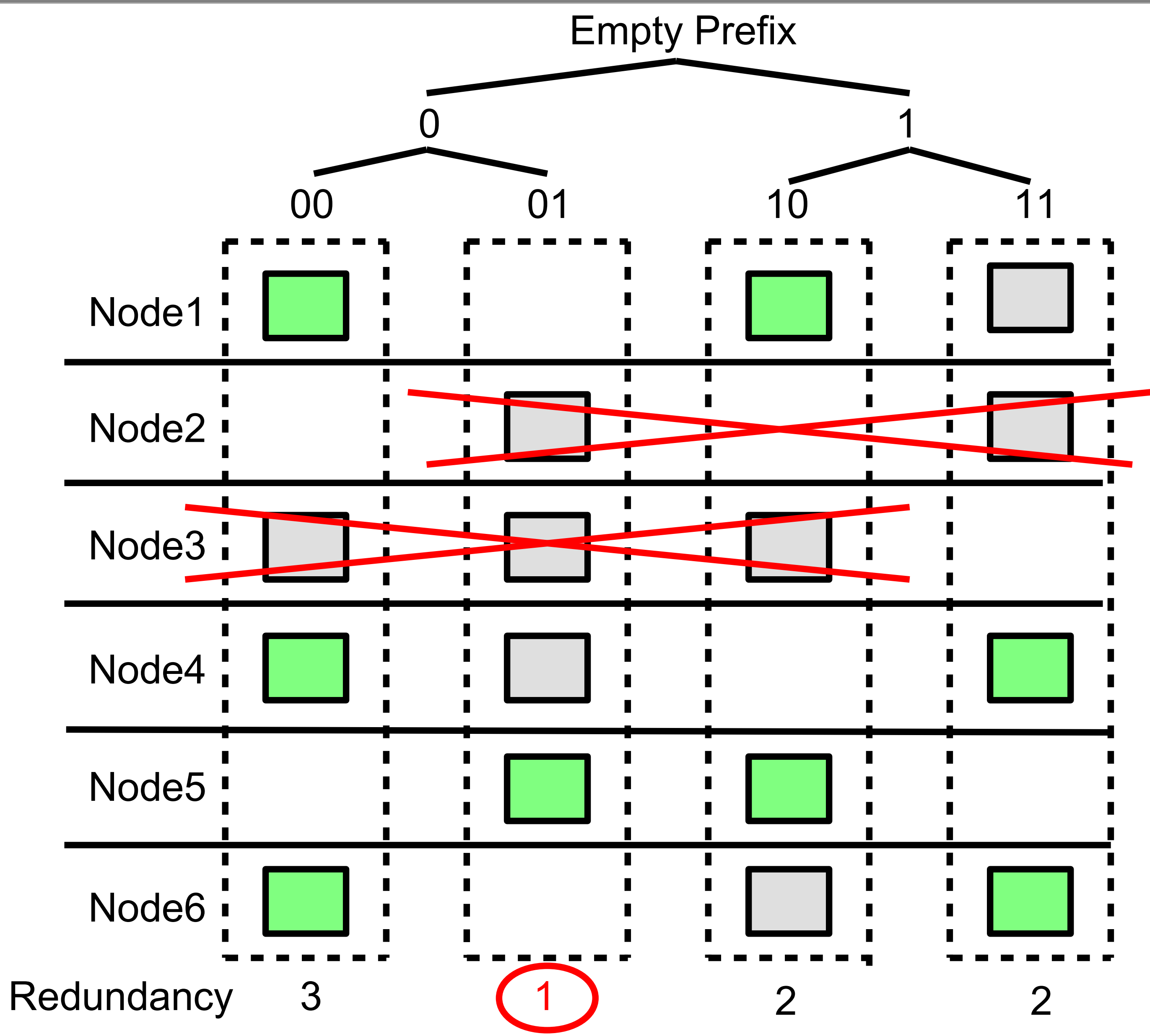
Redundancy of computation



Redundancy of computation



Redundancy of computation



Not enough good peers.
Deletion run is aborted.