*-Box (star-box)
Towards Reliability and Consistency in Dropbox-like File Synchronization Services

Yupu Zhang, Chris Dragga,
Andrea Arpaci-Dusseau, Remzi Arpaci-Dusseau

University of Wisconsin - Madison

Cloud-Based File Synchronization Services

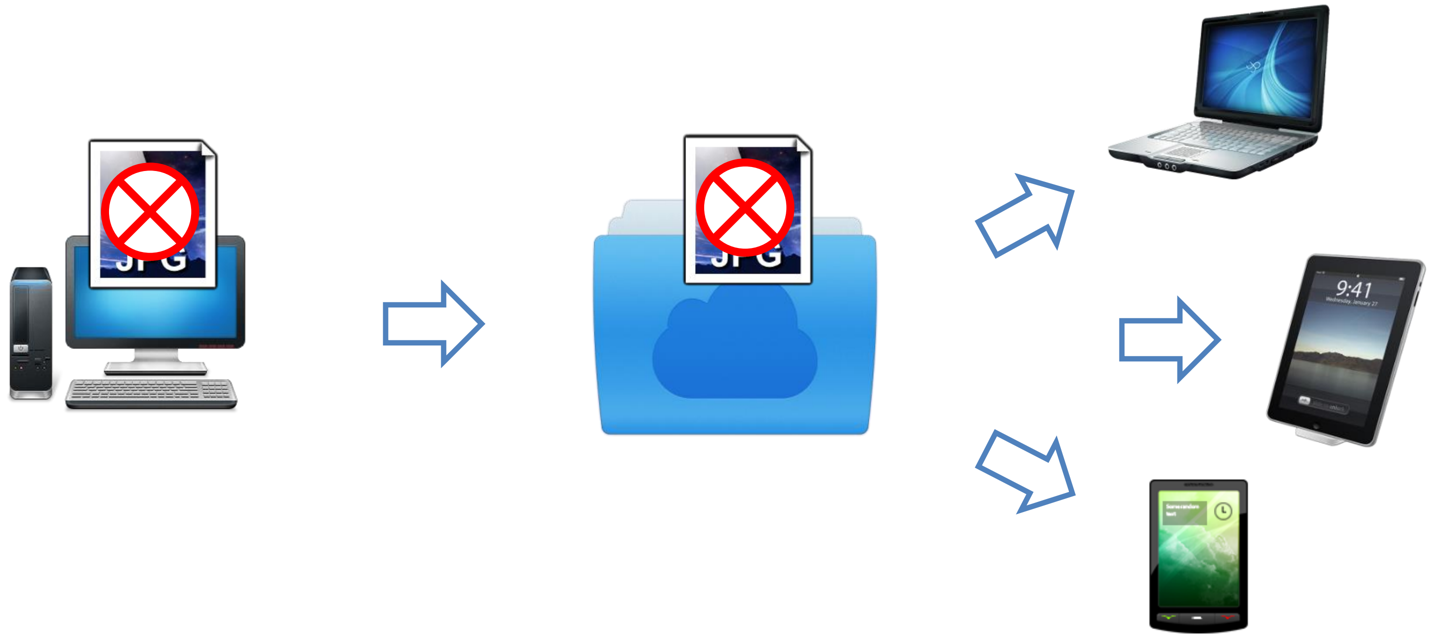
- Exploding in popularity
  - Numerous providers: Dropbox, Google Drive, SkyDrive …
  - Large user base: Dropbox has more than 100 million users
- Key benefit
  - Automatic synchronization across clients/devices
  - Reliable data storage on the server through replication

"your stuff is safe in Dropbox and will never be lost"

- Dropbox tour, page 1

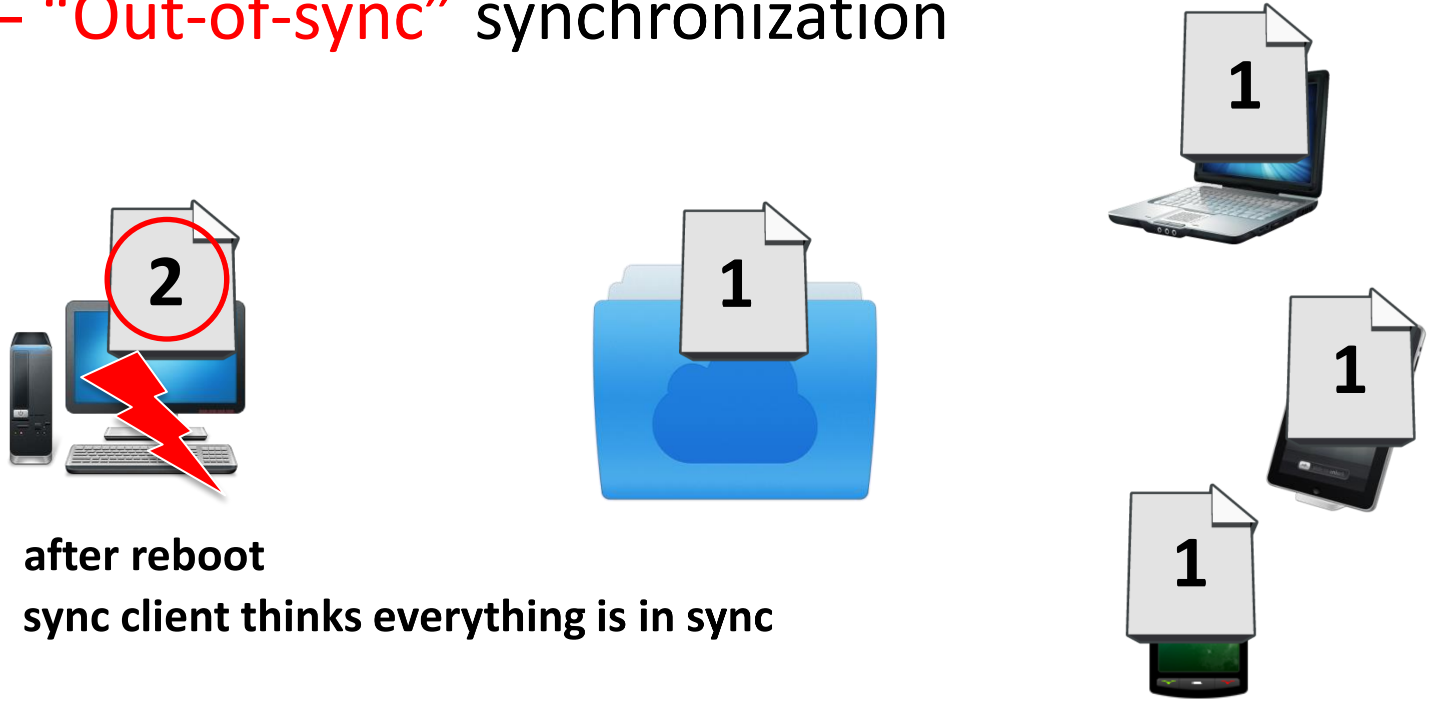
Is Your Data Really Safe?

- Data corruption
  - Uploaded from local machine to cloud
  - Propagated to other devices/clients



Is Your Data Really Safe?

- Crash consistency
  - Inconsistent data ends up everywhere
  - “Out-of-sync” synchronization



after reboot
sync client thinks everything is in sync

Your Data is NOT Really Safe

- False sense of safety
  - Many copies do NOT always make your data safe
- Why?
  - Semantic gap between local file system and cloud
  - Separately designed and loosely linked

Project *-Box (star-box)

- Goal
  - Close the gap between local file system and cloud
  - Provide * without too much infrastructure changes
- * represents desired properties
  - e.g., reliable, consistent, fast, private …
- Currently focus on two properties
  - Reliable: Data corruption
  - Consistent: Crash consistency

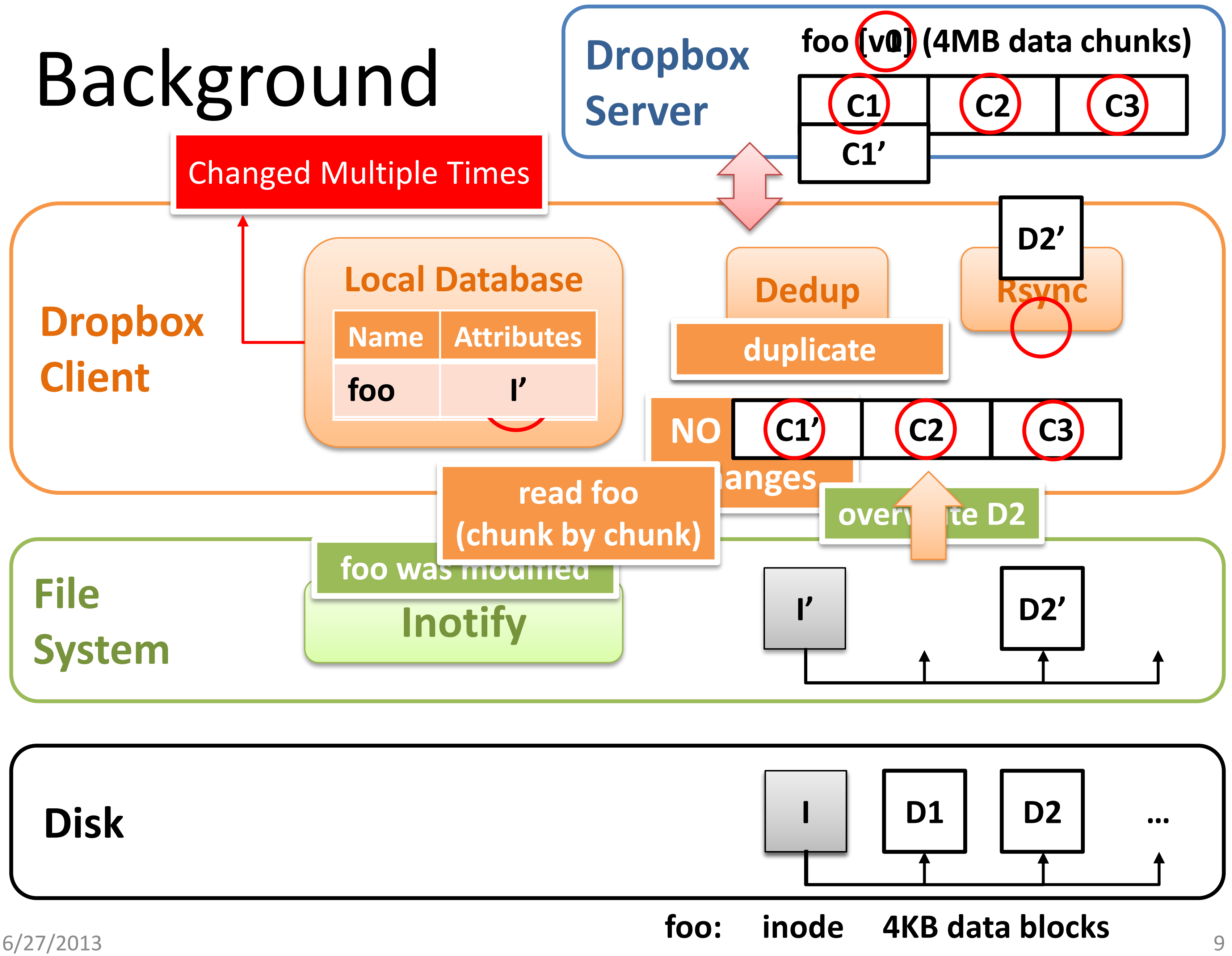
Outline

- Introduction
- **<u>Data Corruption</u>**
- Crash Consistency
- Current Status
- Conclusion

Corruption Problem

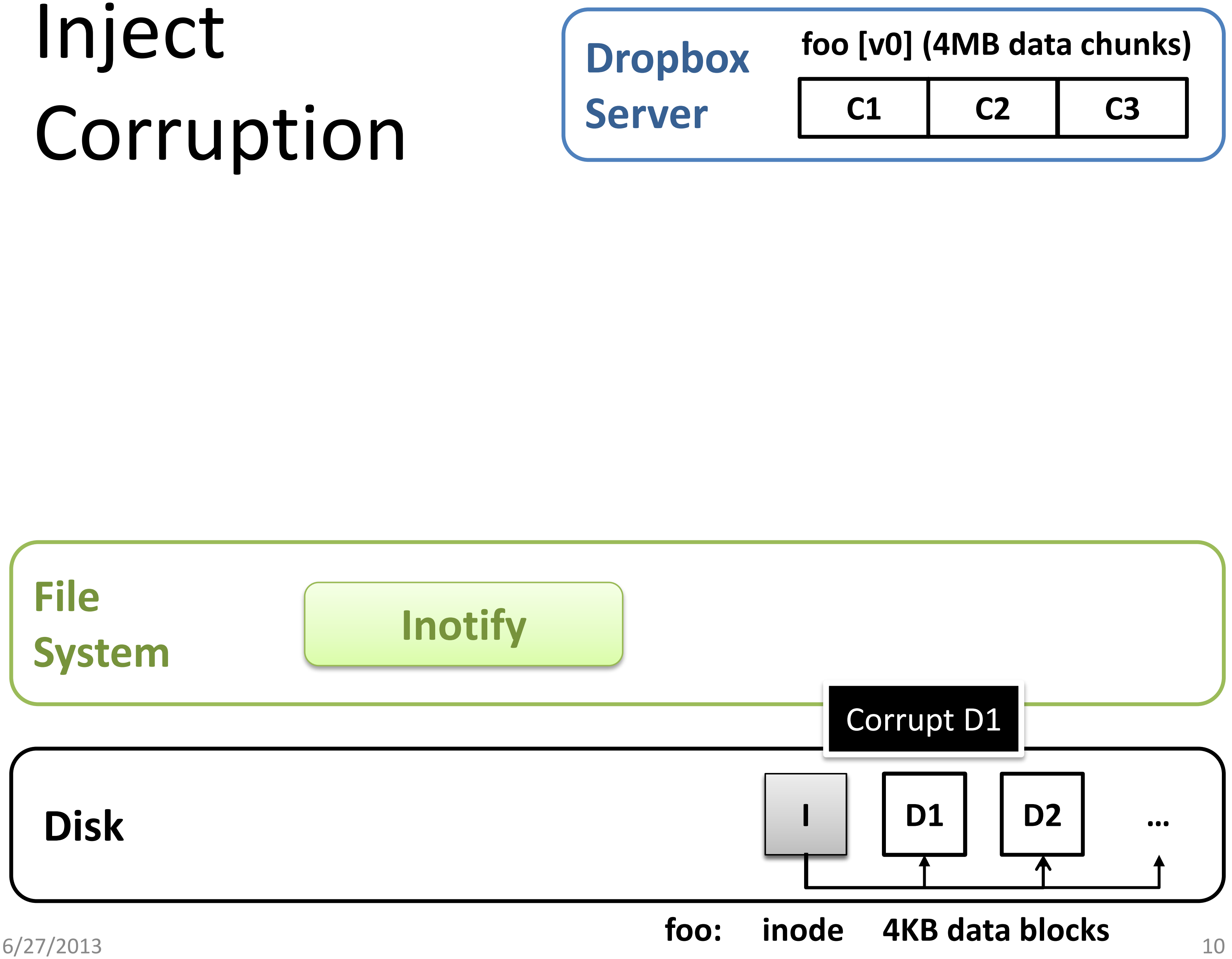
- Data corruption is not uncommon
  - Comes from disk media, firmware, controllers [Bairavasundaram07, Anderson03]
  - Remains local w/o synchronization
- With synchronization
  - Corruption may propagate and pollute other copies
- Synchronization is a double-edged sword
  - Make sure synchronized data is good

Background

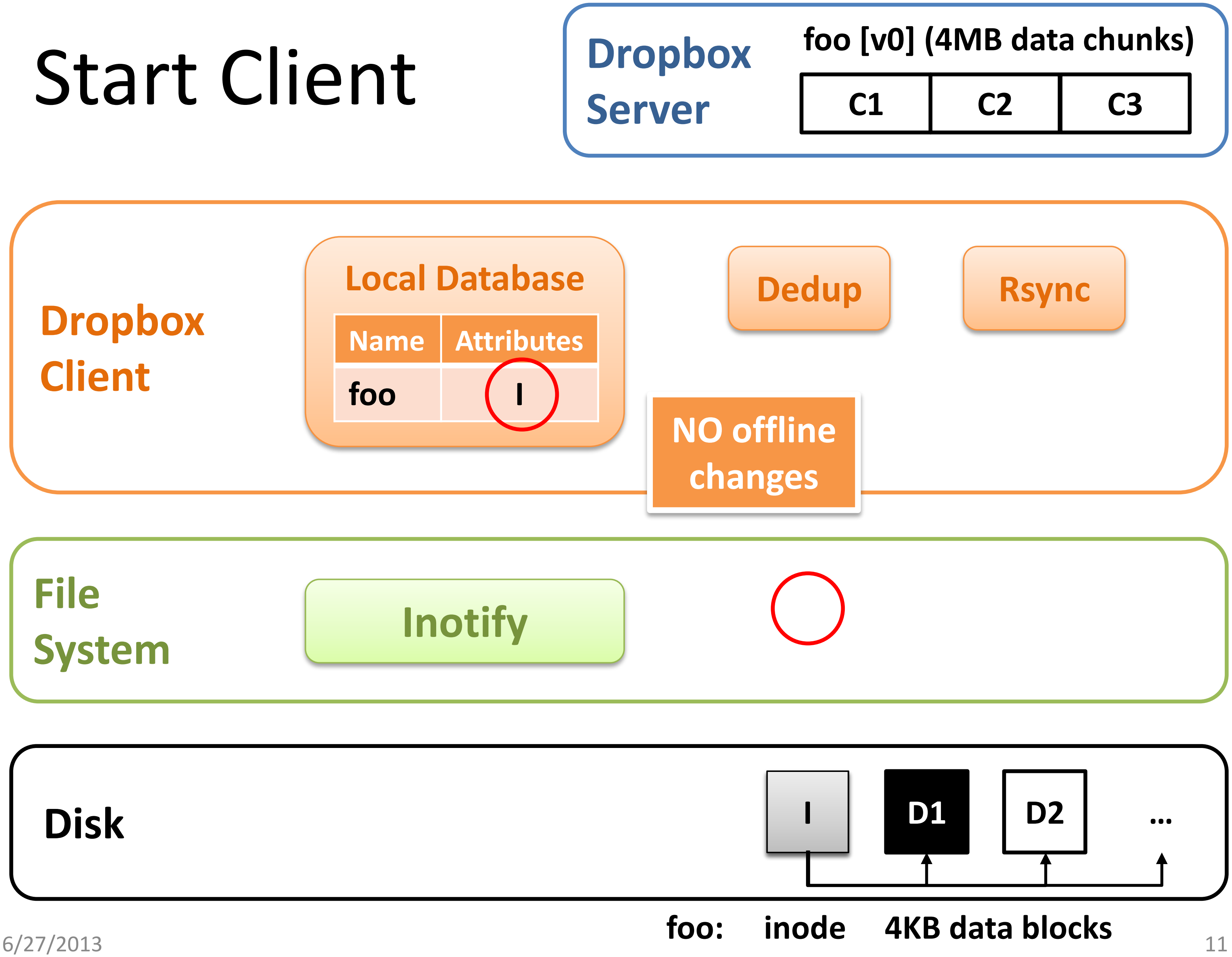


Dropbox Server
foo [v1] (4MB data chunks)
C1
C2
C3
C1'
Changed Multiple Times
Dropbox Client
Local Database
Name
Attributes
foo
I'
Dedup
Rsync
D2'
duplicate
NO
C1'
C2
C3
anges
read foo (chunk by chunk)
overw te D2
File System
foo was modified
Inotify
I'
D2'
Disk
I
D1
D2
...
foo: inode 4KB data blocks

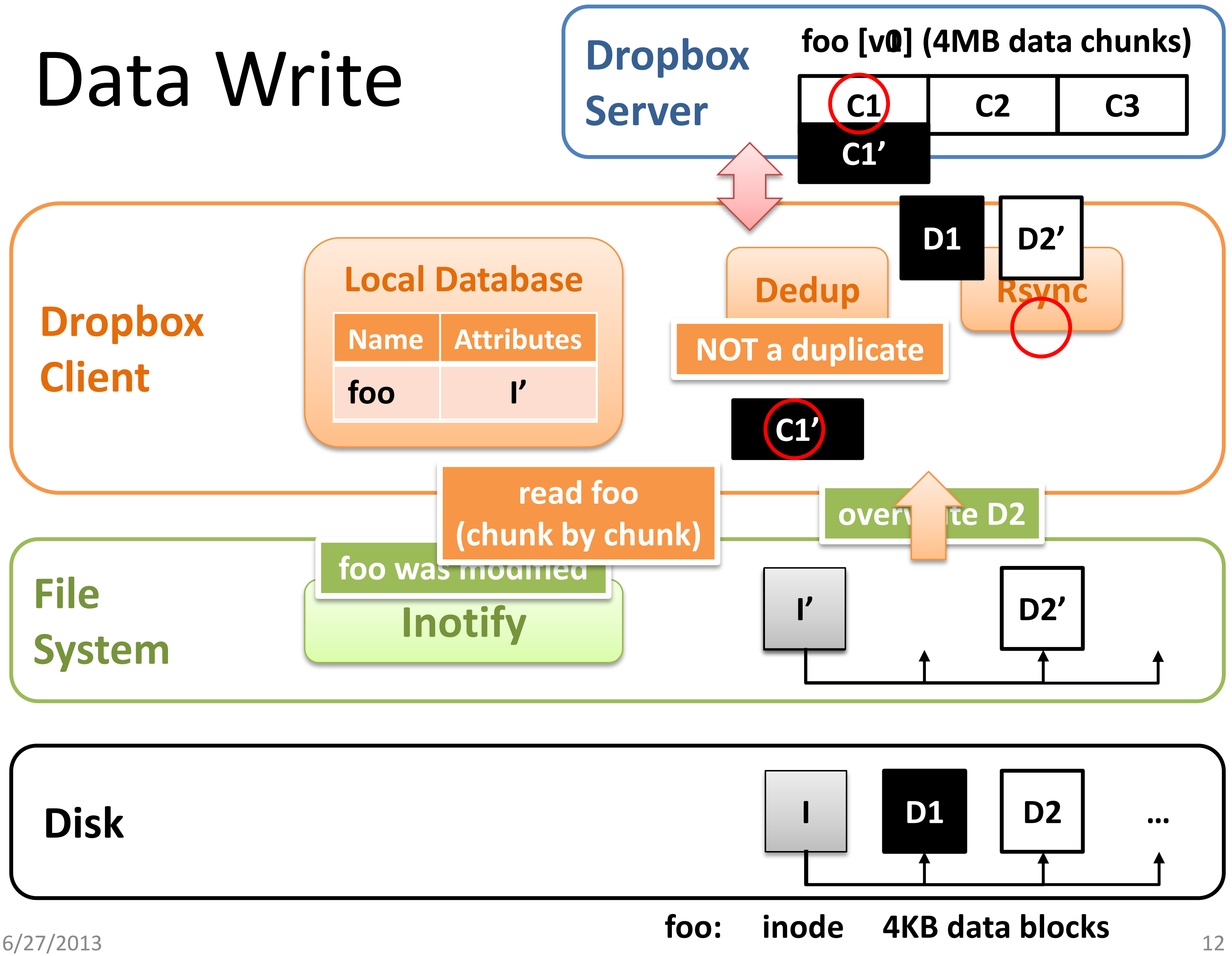
Inject Corruption



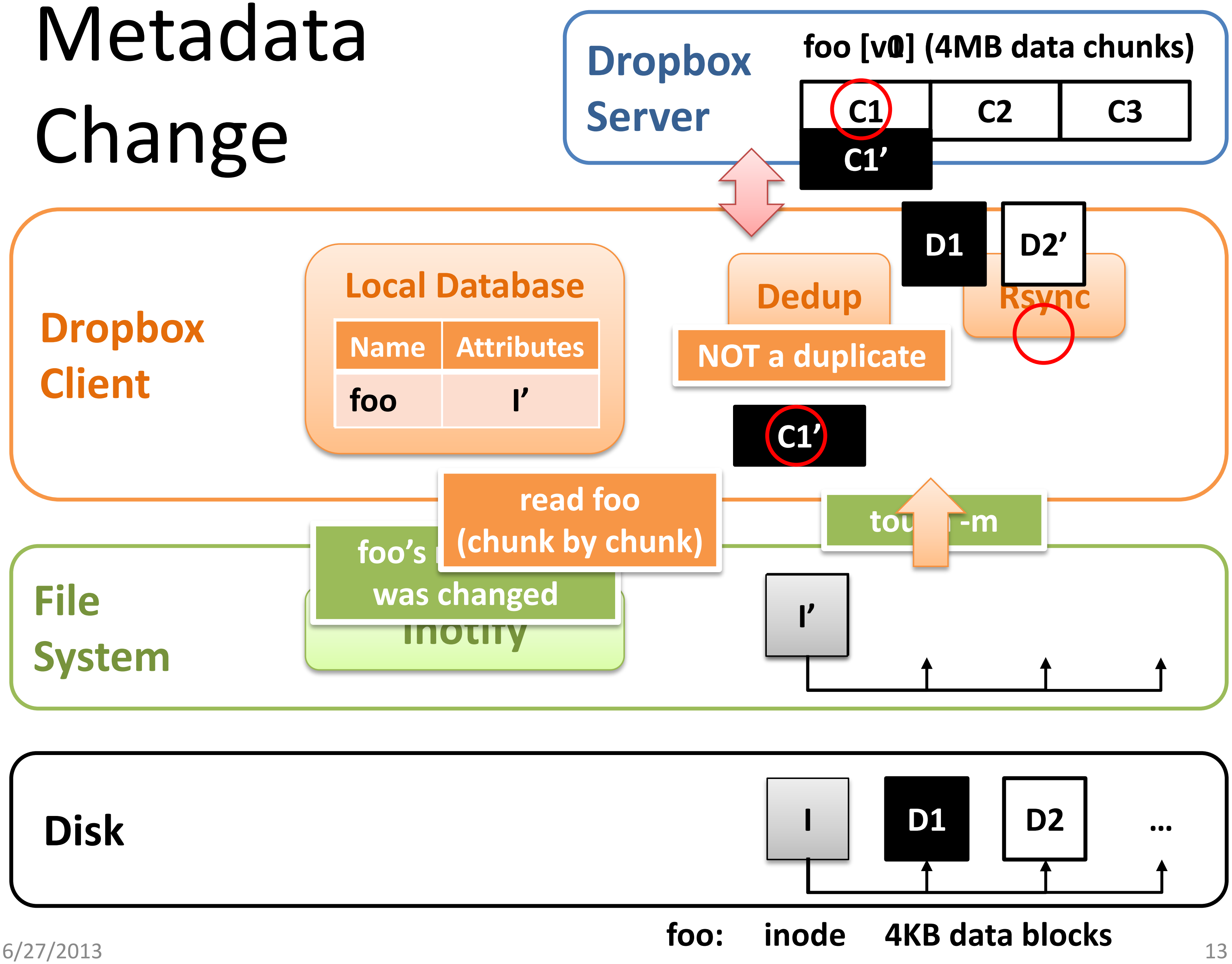
Start Client



Data Write


Dropbox Server
foo [v1] (4MB data chunks)
C1
C2
C3
C1'
D1
D2'
Dropbox Client
Local Database
Name
Attributes
foo
I'
Dedup
Rsync
NOT a duplicate
C1'
read foo (chunk by chunk)
overwrite D2
File System
foo was modified
Inotify
I'
D2'
Disk
I
D1
D2
...
foo: inode 4KB data blocks

Metadata Change



More Results

L: Local corruption
G: Global corruption

FS	Service	Data Writes	Metadata Changes		
			mtime	ctime	atime
ext4 (Linux)	Dropbox	L G	L G	L G	L
	ownCloud	L G	L G	L	L
	FileRock	L G	L G	L	L
HFS+ (Mac OS X)	Dropbox	L G	L G	L	L
	ownCloud	L G	L G	L	L
	GoogleDrive	L G	L G	L	L
	SugarSync	L G	L	L	L
	Syncplicity	L G	L G	L	L

- Corruption is propagated when there is a change to file data
- Even if there is no data change, corruption may still be uploaded

Summary

- Bad bits are promoted to resilient bad bits
  - ALL copies polluted
  - Cloud copies protected by checksum
- Fundamental problem, not implementation bugs
  - FS monitoring services only provide file-level notification
  - Sync clients cannot tell legitimate changes from corruption
- Redundant data on the cloud is not fully utilized
  - If corruption can be detected, local FS can recover from corruption using cloud copies

Outline

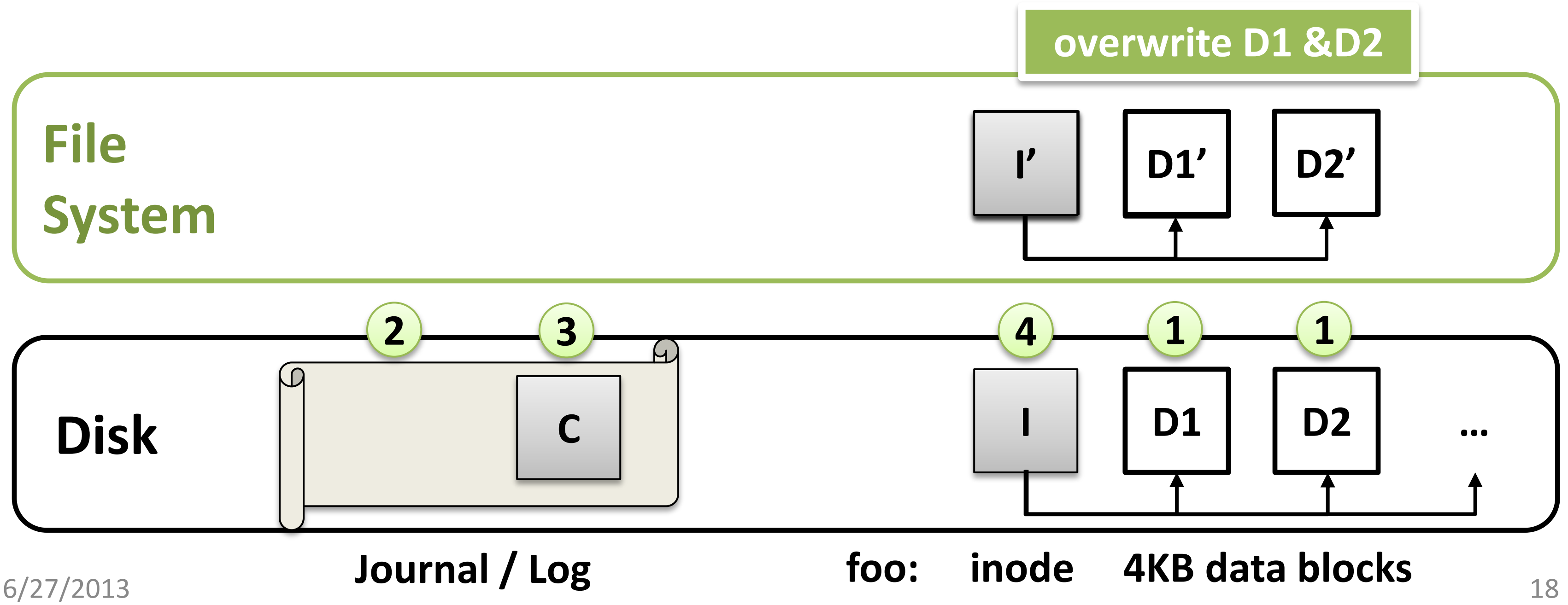
- Introduction
- Data Corruption
- **Crash Consistency**
- Current Status
- Conclusion

Crash Recovery Techniques

- Copy-on-write (e.g., ZFS, btrfs)
  - Always roll back to a consistent version
- Journaling (e.g., ext4)
  - Data journaling mode
    - Both data and metadata are logged
    - Provide data consistency
  - Ordered journaling mode
    - Only journal metadata
    - Data blocks are written before metadata is logged
    - Cannot guarantee data consistency

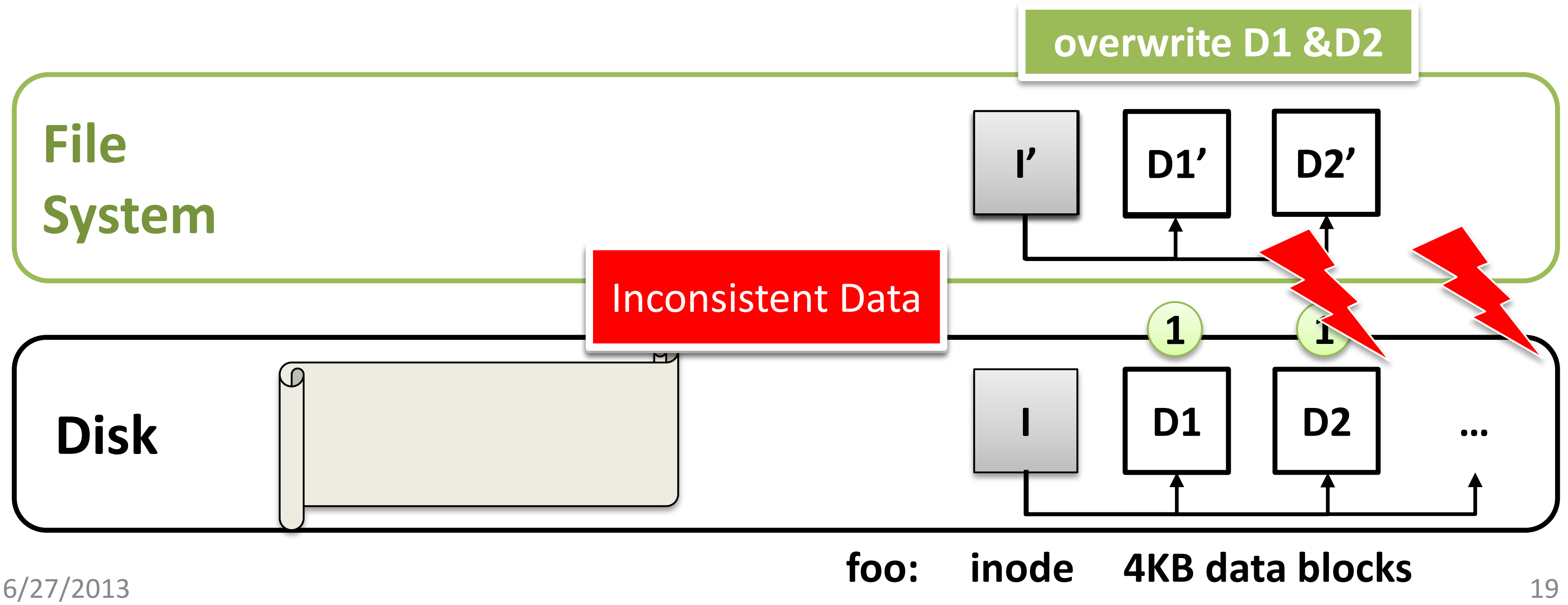
Ordered Mode

1. Write dirty data blocks to home locations
2. Write metadata blocks to journal
3. Write journal commit block to the journal
4. Checkpoint journaled metadata blocks to home locations



Crash in Ordered Mode

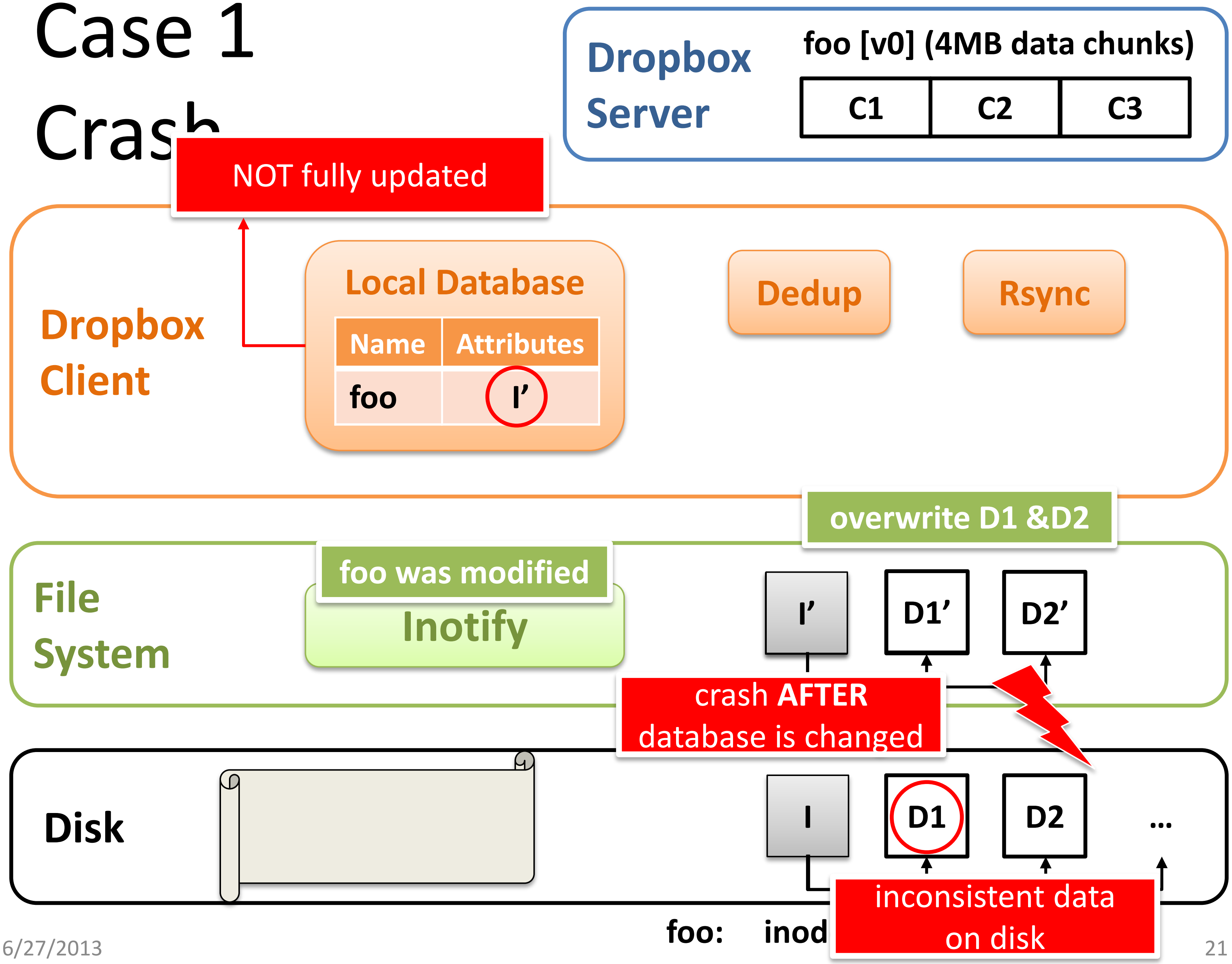
- Crash during step 1
  - Write dirty data blocks to home locations



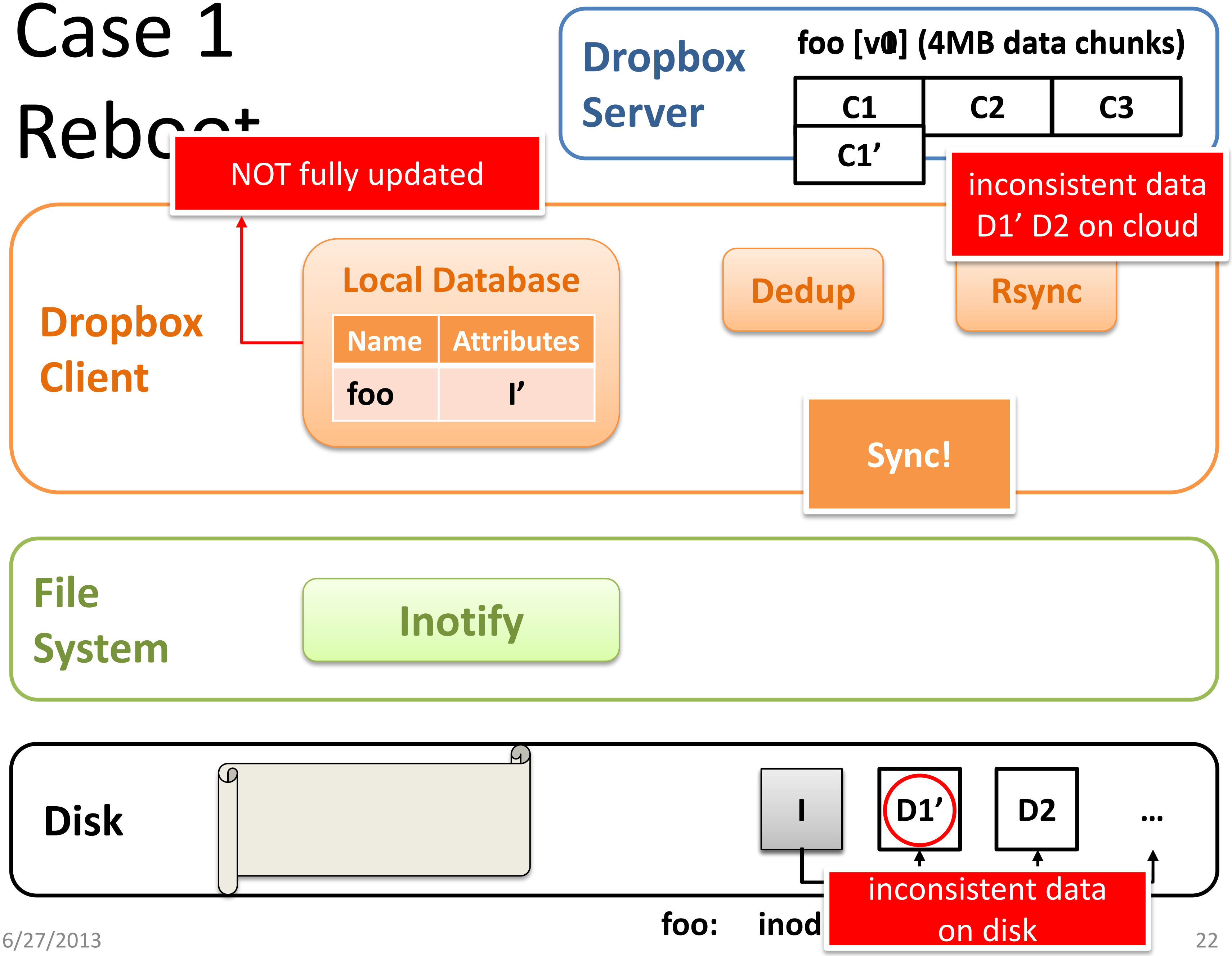
Ext4 Ordered Mode + Dropbox

- Case 1
  - Inconsistent data is propagated
- Case 2
  - Consistent data is NOT synchronized

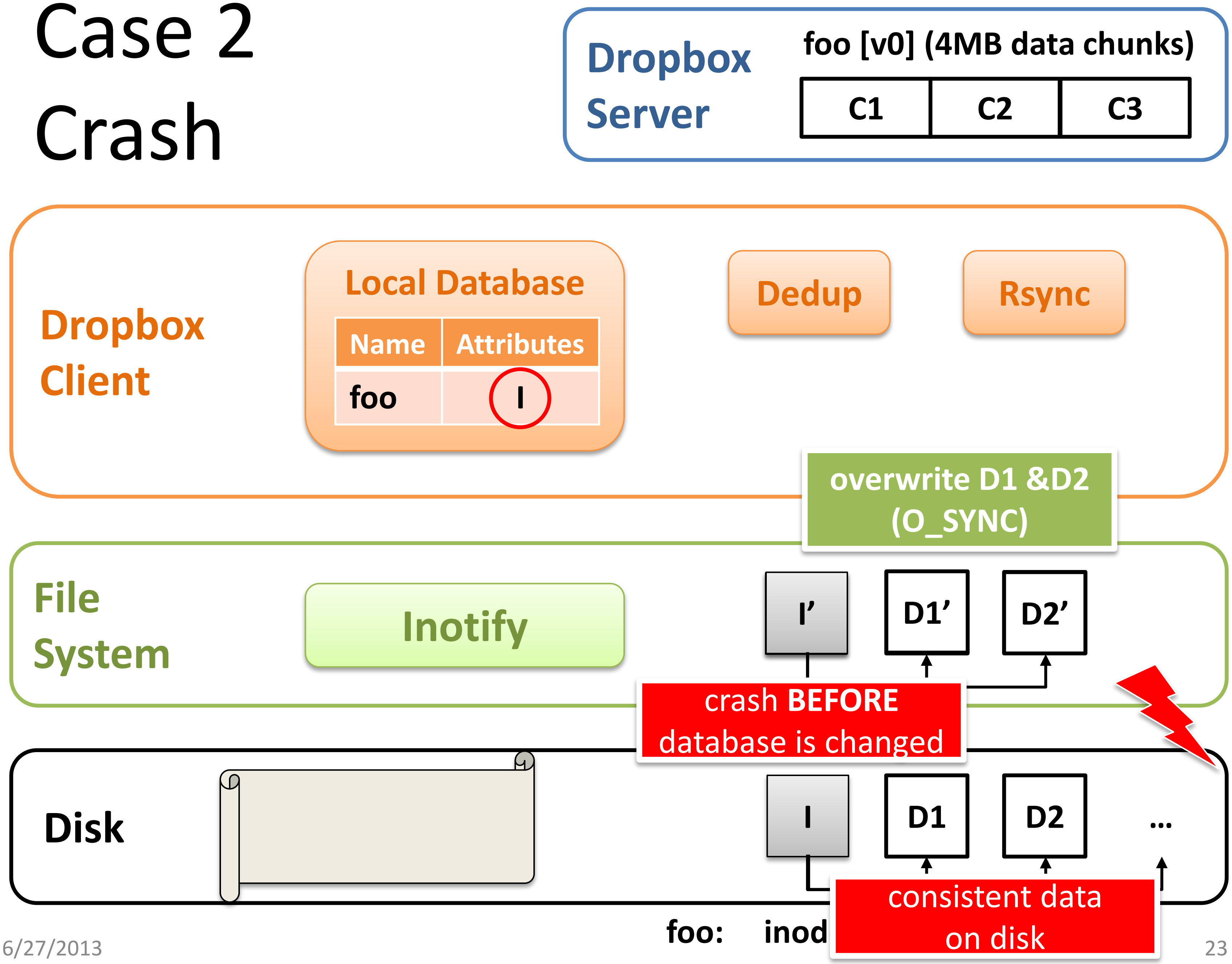
Case 1 Crash


Dropbox Server
foo [v0] (4MB data chunks)
C1
C2
C3
NOT fully updated
Dropbox Client
Local Database
Name
Attributes
foo
I'
Dedup
Rsync
overwrite D1 &D2
File System
foo was modified
Inotify
I'
D1'
D2'
crash AFTER database is changed
Disk
I
D1
D2
...
inconsistent data on disk
foo: inod

Case 1
Reboot

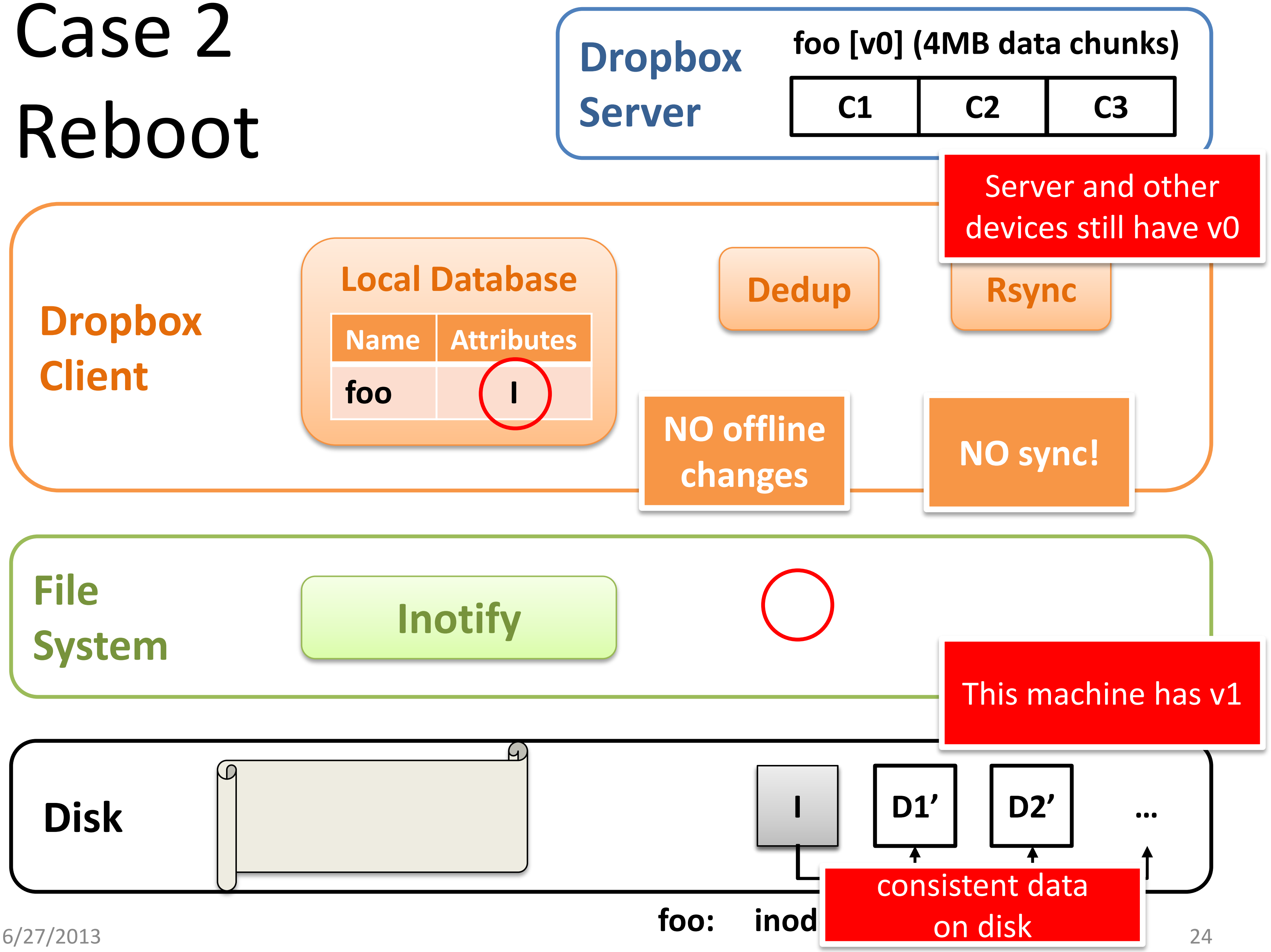


Case 2 Crash



Dropbox Server
foo [v0] (4MB data chunks)
C1
C2
C3
Dropbox Client
Local Database
Name
Attributes
foo
I
Dedup
Rsync
overwrite D1 &D2 (O_SYNC)
File System
Inotify
I'
D1'
D2'
crash **BEFORE** database is changed
Disk
I
D1
D2
...
foo:
inod
consistent data on disk

Case 2
Reboot



Recover using Data on Cloud?

- Data on server does not always reflect a consistent state on disk
  - Dropbox uploads data asynchronously
  - Dropbox reorders file uploading
  - Actively modified files may get delayed

- When crash occurs, files on server could be inconsistent with respect to disk

Summary

- Inconsistent content gets propagated
- “Out-of-sync” files may exist
  - Different client/devices see different versions of the same file
- Need in-depth communication between local FS and cloud
  - Cloud has very weak sense of actual FS state

Outline

- Introduction
- Data Corruption
- Crash Consistency
- **Current Status**
- Conclusion

Current Status of *-Box

- Finished
  - Data checksumming in ext4
  - Dropbox-aided corruption recovery in ext4
  - Fine-grained inotify in Linux
    - Add a ranged file-update notification
    - Adapt ownCloud (an open-source sync service) to use it
- In-progress
  - Use in-memory snapshot to facilitate crash recovery in ext4 ordered mode

Outline

- Introduction
- Data Corruption
- Crash Consistency
- Possible Solutions
- **Conclusion**

Conclusion

- Many copies do NOT always make your data safe
  - Propagation of corrupt data and inconsistent state
  - Synchronized files are out-of-sync
- Propose *-Box project
  - Solve problems by reducing the semantic gap between existing local FS and cloud storage
- Ultimately may need a cohesive system that provides capabilities unachievable in isolation

Thank you!

Questions?

Advanced Systems Lab (ADSL)

University of Wisconsin-Madison

http://www.cs.wisc.edu/adsl

Wisconsin Institute on Software-defined Datacenters in Madison

http://wisdom.cs.wisc.edu/