# Device-Transparent Personal Storage

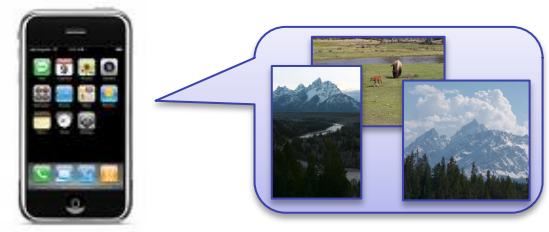
Jacob Strauss, Justin Mazzola Paluska, Chris Lesniewski-Laas
Bryan Ford, Robert Morris, Frans Kaashoek

Quanta Research Cambridge MIT Yale

June 17, 2011

# Personal Data Management: Point-to-point Synchronization

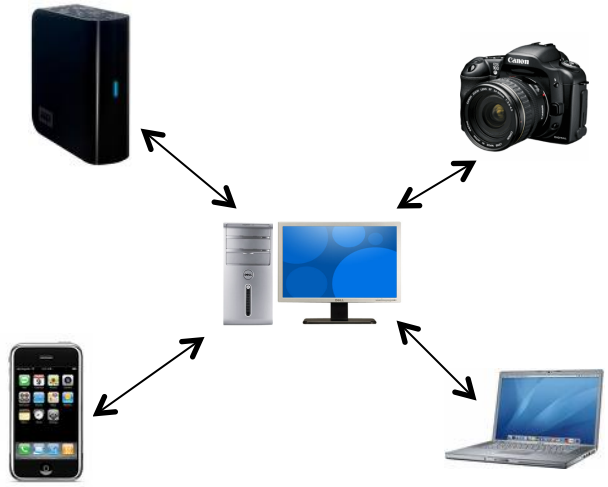
1. Take photos



2. Go home, sync new photos to desktop

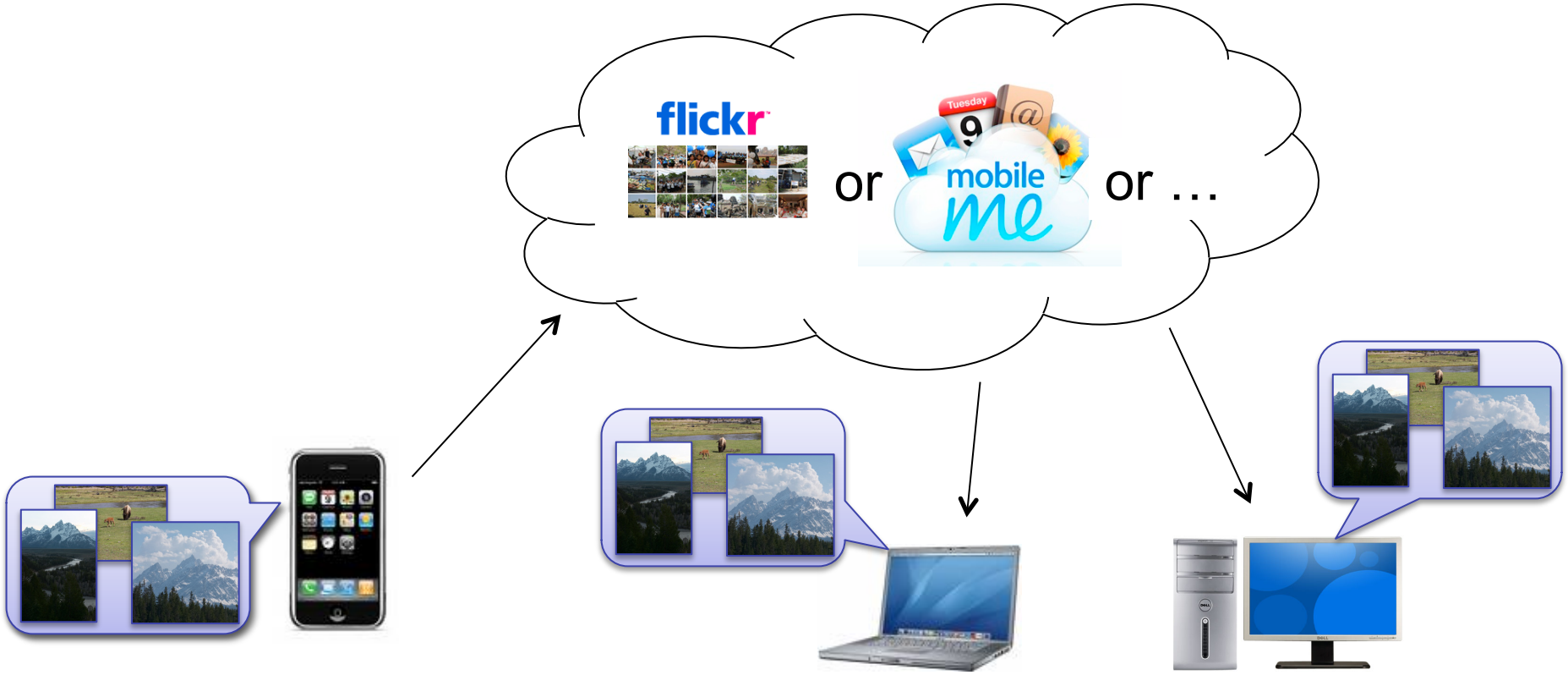
- Good Properties:
  - Local connection: fast & inexpensive
  - Simple to use

# Synchronization Among Multiple Devices



- Single server to hold & organize entire collection
- Requires hub be reachable

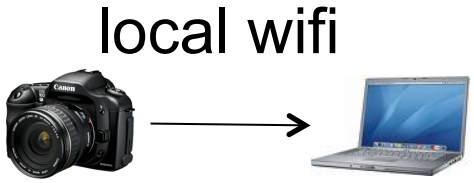
# Store & Fetch from Cloud




- More flexibility than a single hub
- Not always reachable, can be slow

# Ad-hoc Manual Management

- Push manually to nearby device for more storage




- Upload to cloud later when connected


- Problem: user must track where objects are

# Ideal: Device Transparent Storage


Same global view of data collection from each device

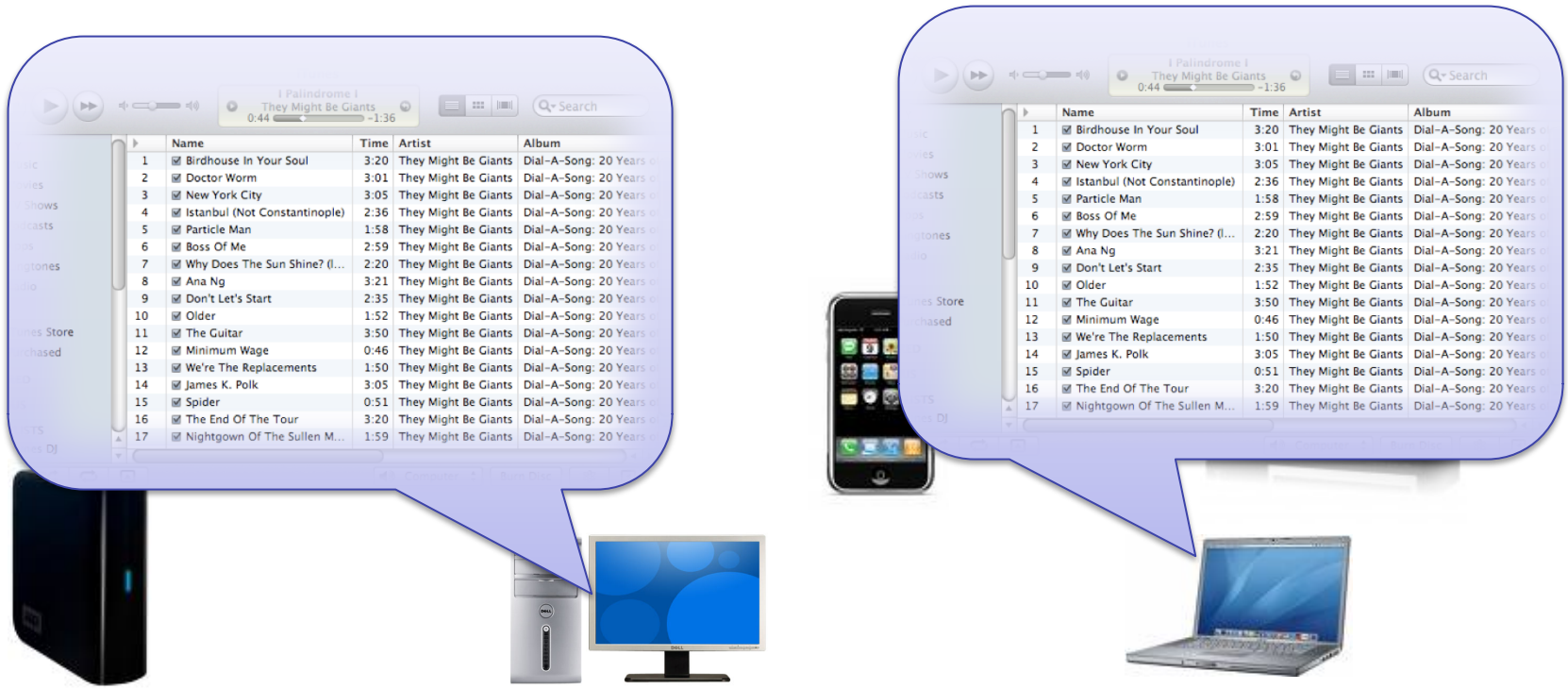
# Device-Transparency: Impossible?

- Limited Storage Capacity
  - Can't put everything everywhere
- Devices might be disconnected
  - Can't use files stored on unreachable devices

# Approach: Split Metadata from Content

- Fully replicate all metadata
  - Small: fits everywhere
- Partially replicate all content
  - Not small: place where needed

# Metadata is Useful Alone

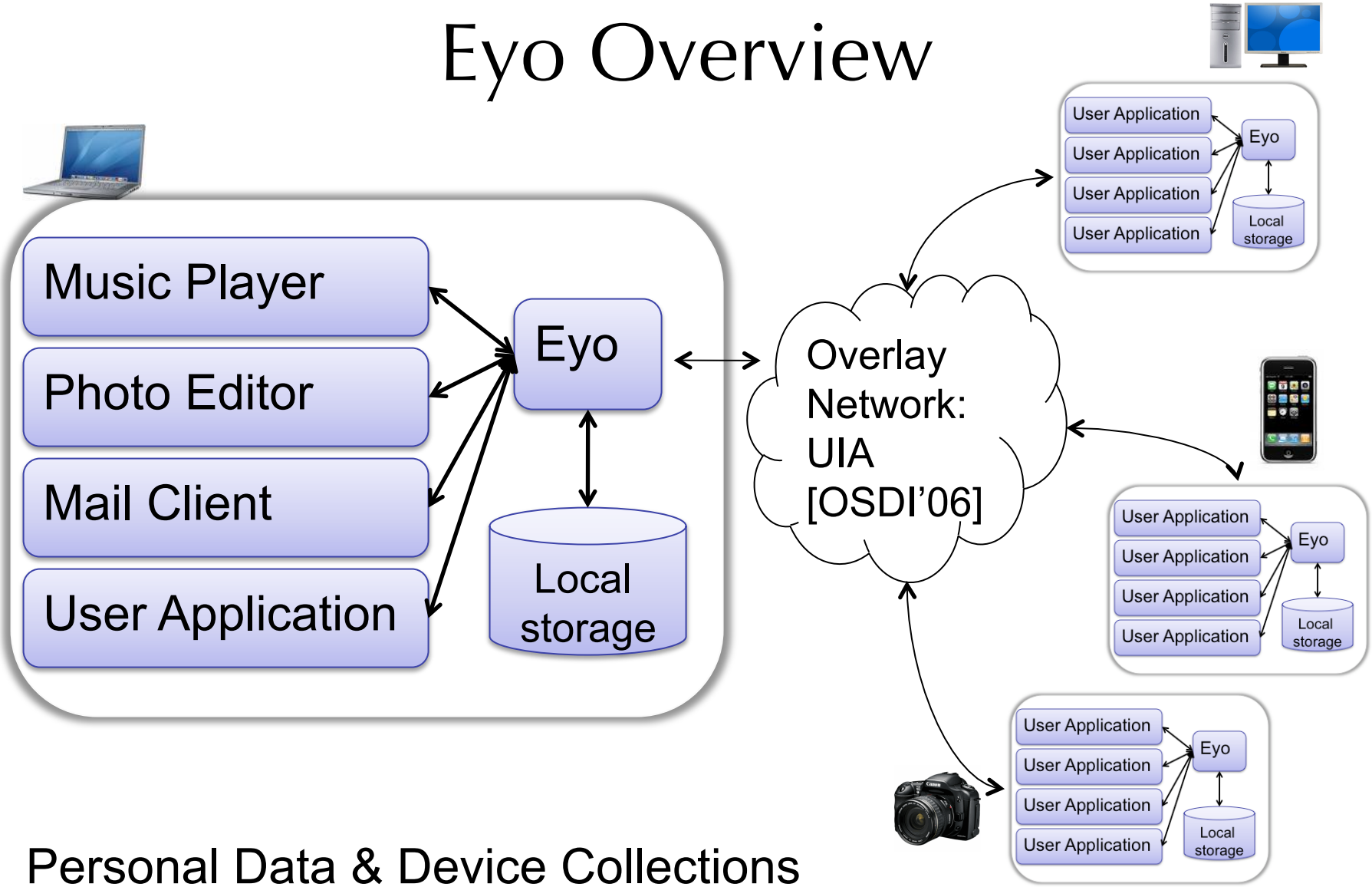



- When disconnected & without content:
  - View complete collections of objects
  - Move objects between collections
  - Identify devices that do hold the content

# Device-Transparent Storage Approach

- Separate Metadata from Content
  - Global Metadata Replication
  - Partial Content Replication
- Peer-to-peer Continuous Synchronization
  - Approximate global store as connectivity permits
- Automate Conflict Resolution
  - Eventually consistent metadata collection

# Eyo Overview

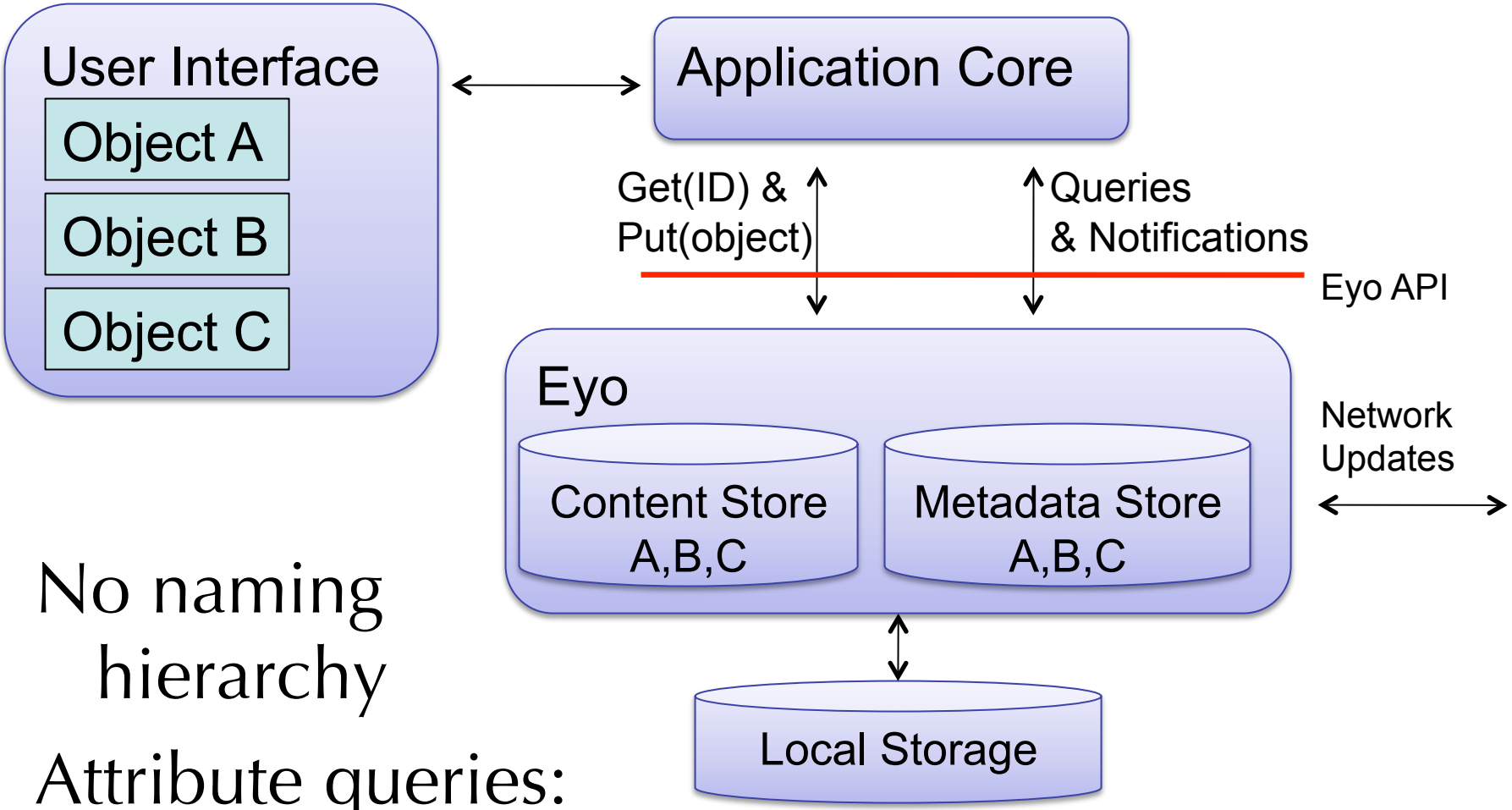



Personal Data & Device Collections

# Eyo API Design

- Challenge: Automated Conflict Resolution
- API Properties
  - First-class version history
  - Explicit metadata and content split
  - Placement policy
- Borrows mechanisms from existing work
  - Distributed File Systems, Optimistic Replication, Version Control Systems

# Using the Eyo Storage API

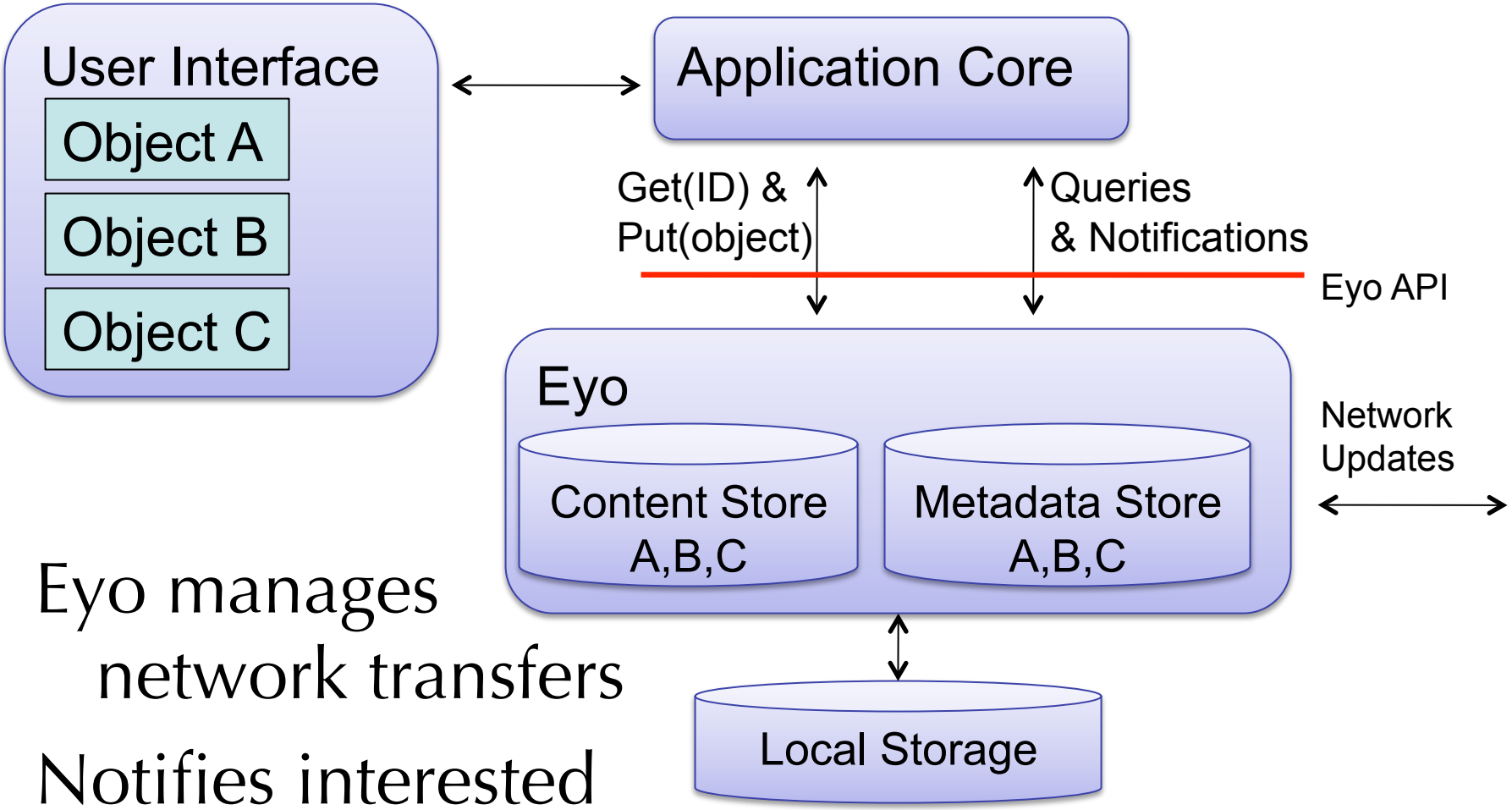



No naming hierarchy

Attribute queries:

List of objects ← lookup(has key ‘content-type’ with value ‘image’)

# Using the Eyo Storage API




Eyo manages network transfers

Notifies interested applications when updates arrive

# Content Store

- Content block per object
  - Immutable after writing
- Device holds subset of all content
  - Guided by placement rules [Cimbiosys, Perspective]
  - Application specified query mapping objects to set of devices
    - Ex: songs with tag "top-rated" → ipod

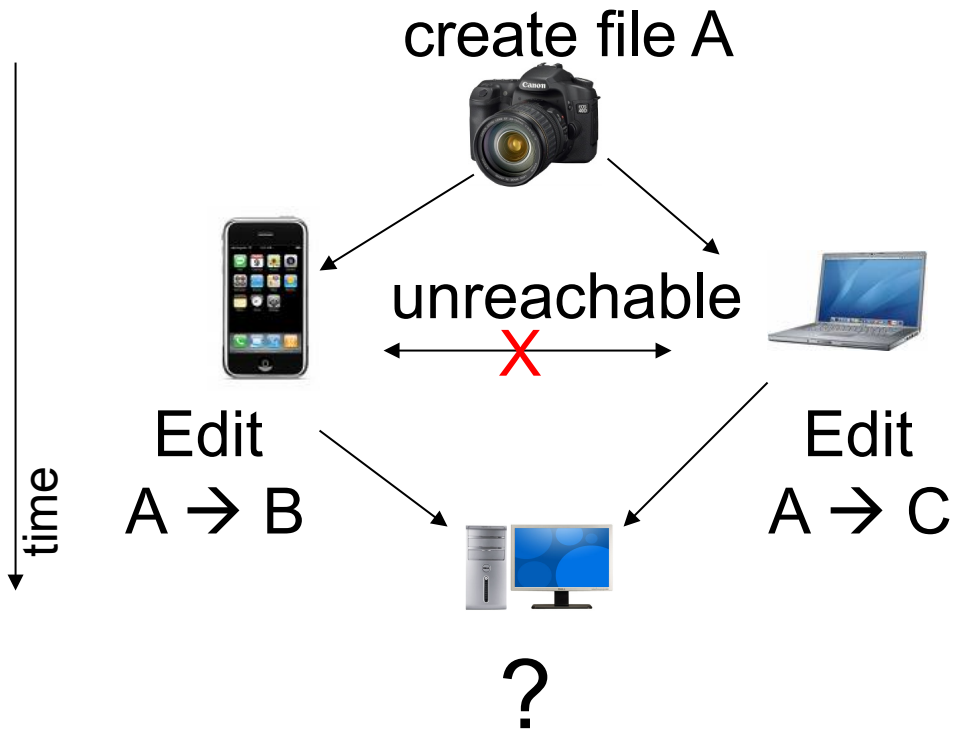
# Metadata Store

- Metadata includes:
  - everything users need to name and find objects
  - album, song name, artist, location, etc.

Content-type: audio
Size: 1234
Artist: U2
Album: The Joshua Tree
Playlists: 80's
Rating: 4/5

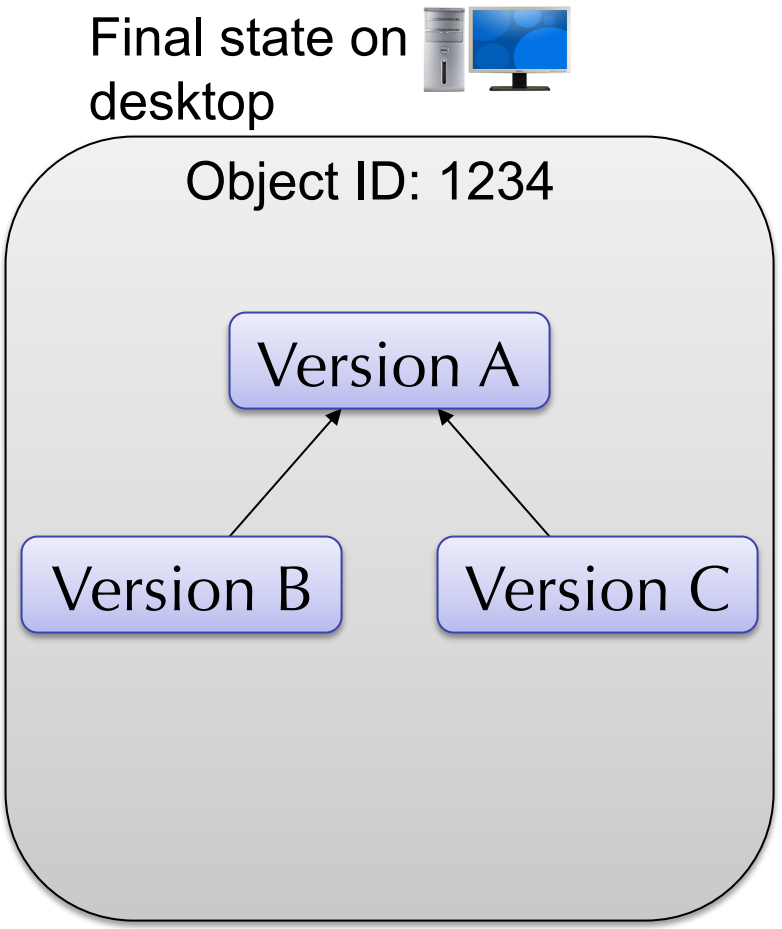
- Eyo replicates metadata store to all of user's devices
  - Each device knows about all objects
- Small enough to store everywhere
  - Small updates: quick, frequent transfers

# Concurrent Updates to Metadata



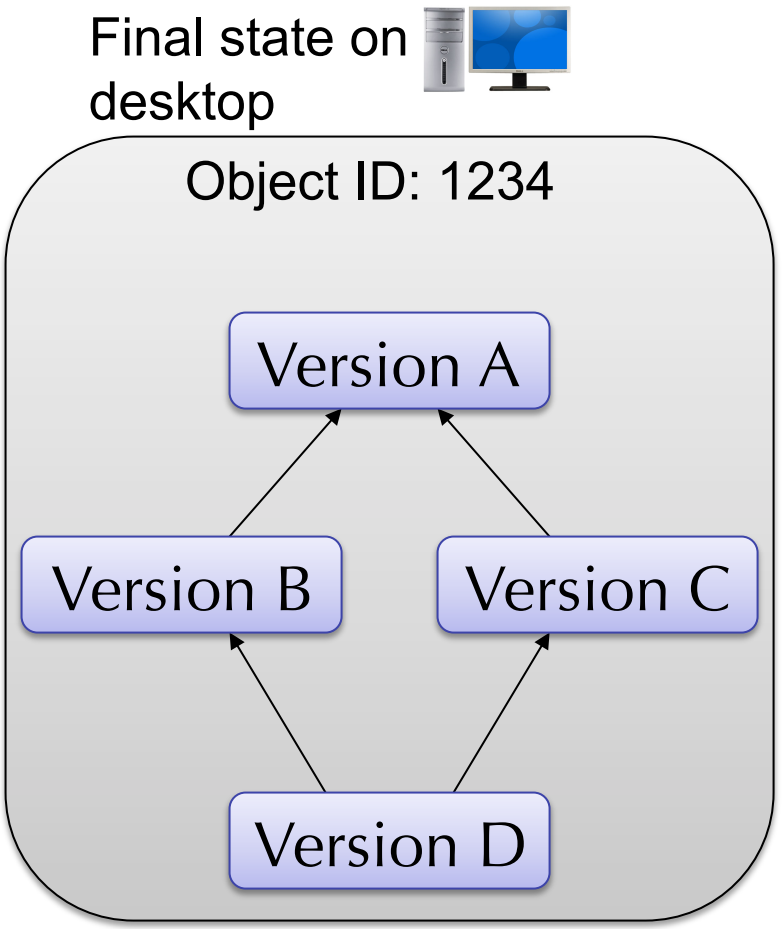

- Disconnected changes lead to conflicts
- When and where should these be resolved?

# Handling Conflicts



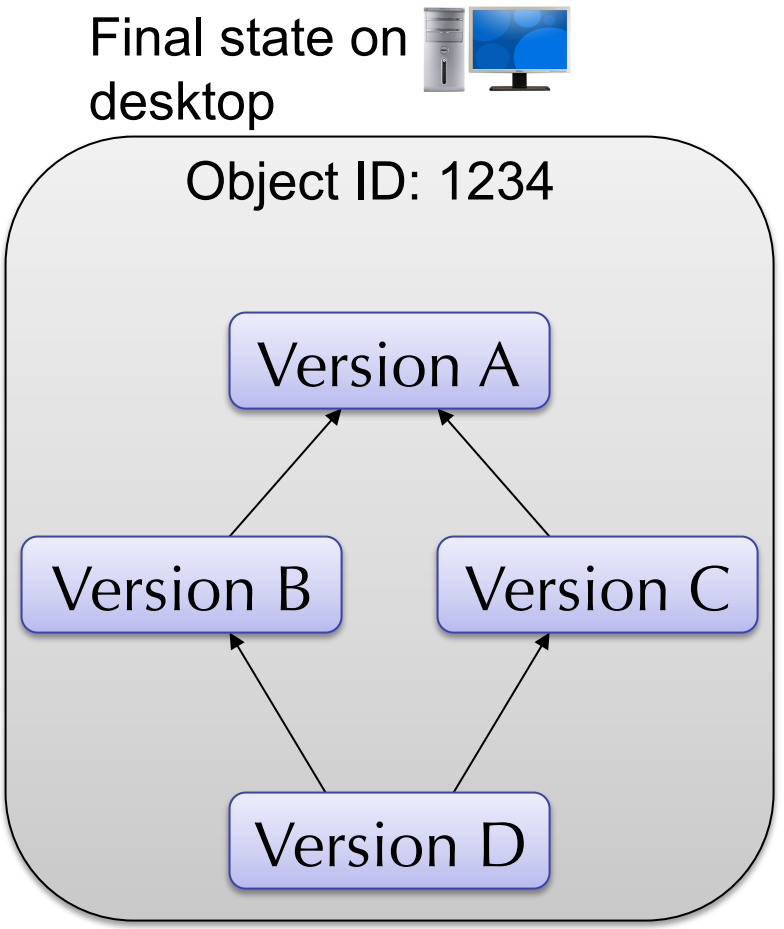

- Track common ancestor
- Eyo provides version history to applications
- Applications specify predecessors when replacing old versions
- Compare to branches in version control systems
- Permits many concurrency strategies

# Handling Conflicts




- Do nothing: fork history
- Pick arbitrarily (based on timestamp)
- Let the user pick a version
  - Media player: song title: A→B and A→C
  - Write a new version that replaces both B and C

# Handling Conflicts Automatically




- Use application-specific knowledge
  - Media Player:
    - play song in two places,
    - increment playcount on each
    - Merge to total sum
  - Photo Editor:
    - Tag photos concurrently; merge to include both
  - User never aware a conflict occurred

# Storage API Summary

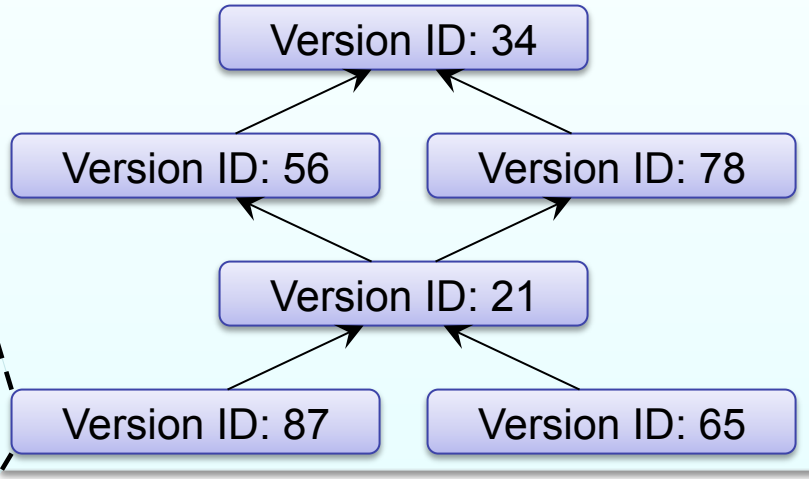
Eyo Objects

Object ID: 12 | Object ID: 34 | Object ID: 56 | . . .

Version 87
Metadata

| Keys | Values |
|---|---|
| Content-type | Image/jpg |
| Content-length | 5000 |
| Aperture | f/5.6 |
| Resolution | 1024x768 |
| ISO equiv | 400 |
| Name | dog.jpg |
| Date | 4/27/10 |
| Predecessor | Version 21 |
| Content ID | Content 41 |

Object ID: 56




Content Store

ID: 41 Value:

# API Implementation Challenges

- Device to Device Connectivity
  - Which devices?
  - Where are they?
  - Secure communication

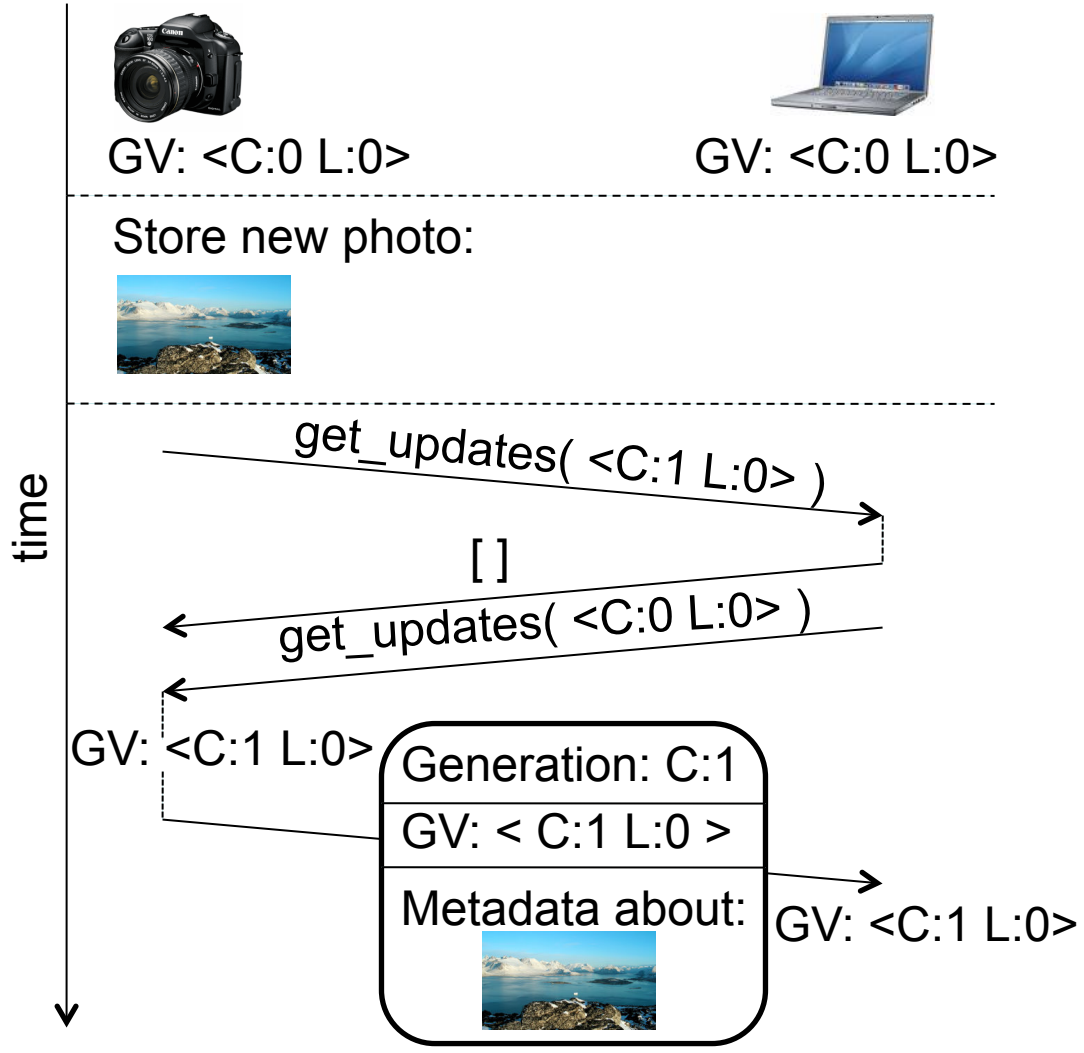
  } Provided by UIA [OSDI'06]
- Continuous Synchronization
  - Approximates device transparency
  - Send updates between all reachable peers
  - How to do so efficiently?

# Separate Synchronization Protocols

- Metadata
  - Fast, frequent, small changes
  - Result in identical collections
  - Use metadata to track content
- Content
  - Can be big, slow to move
  - Place objects where they belong

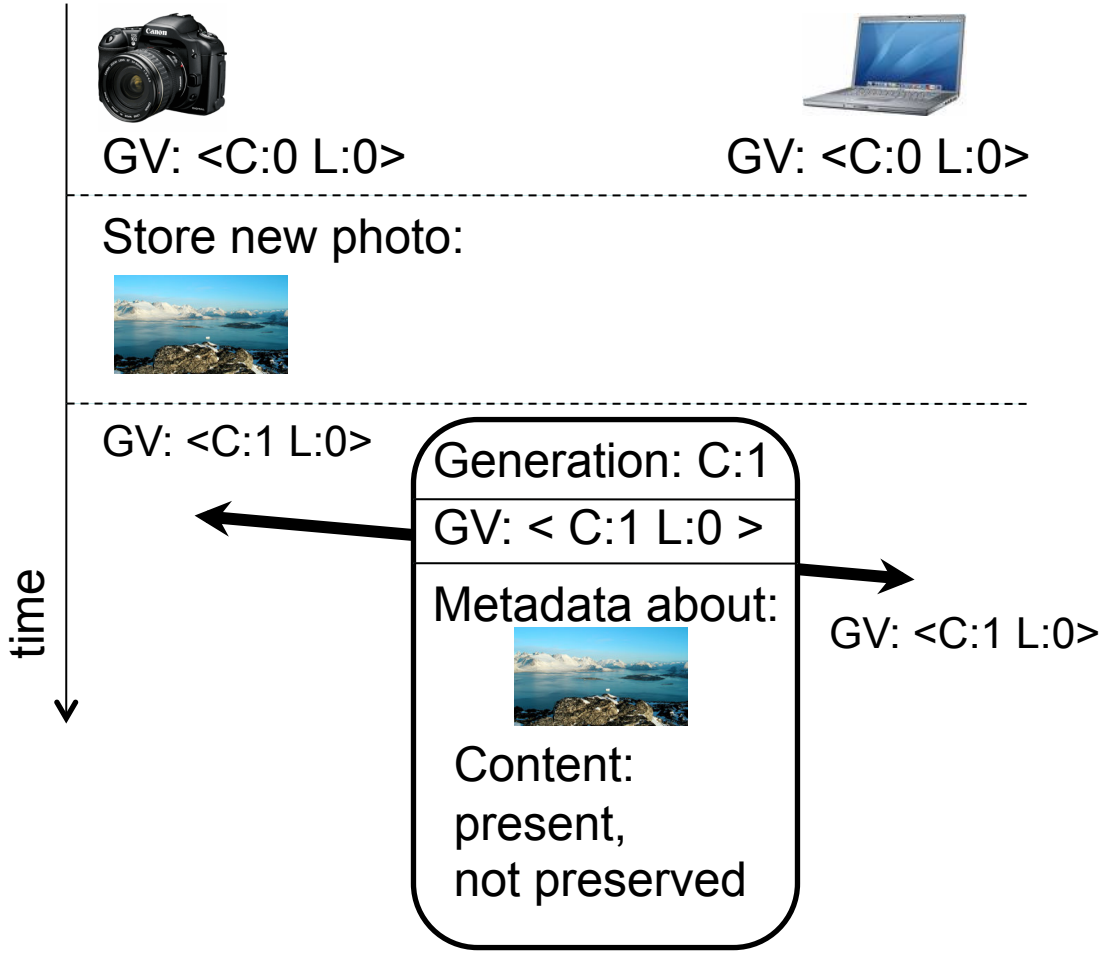
# Metadata Synchronization

- Find and send only changed objects from large set of unchanged objects
- Group updates into an immutable *Generation*
- Single *Generation Vector* describes set of updates each device has seen
- Single lookup identifies state to send



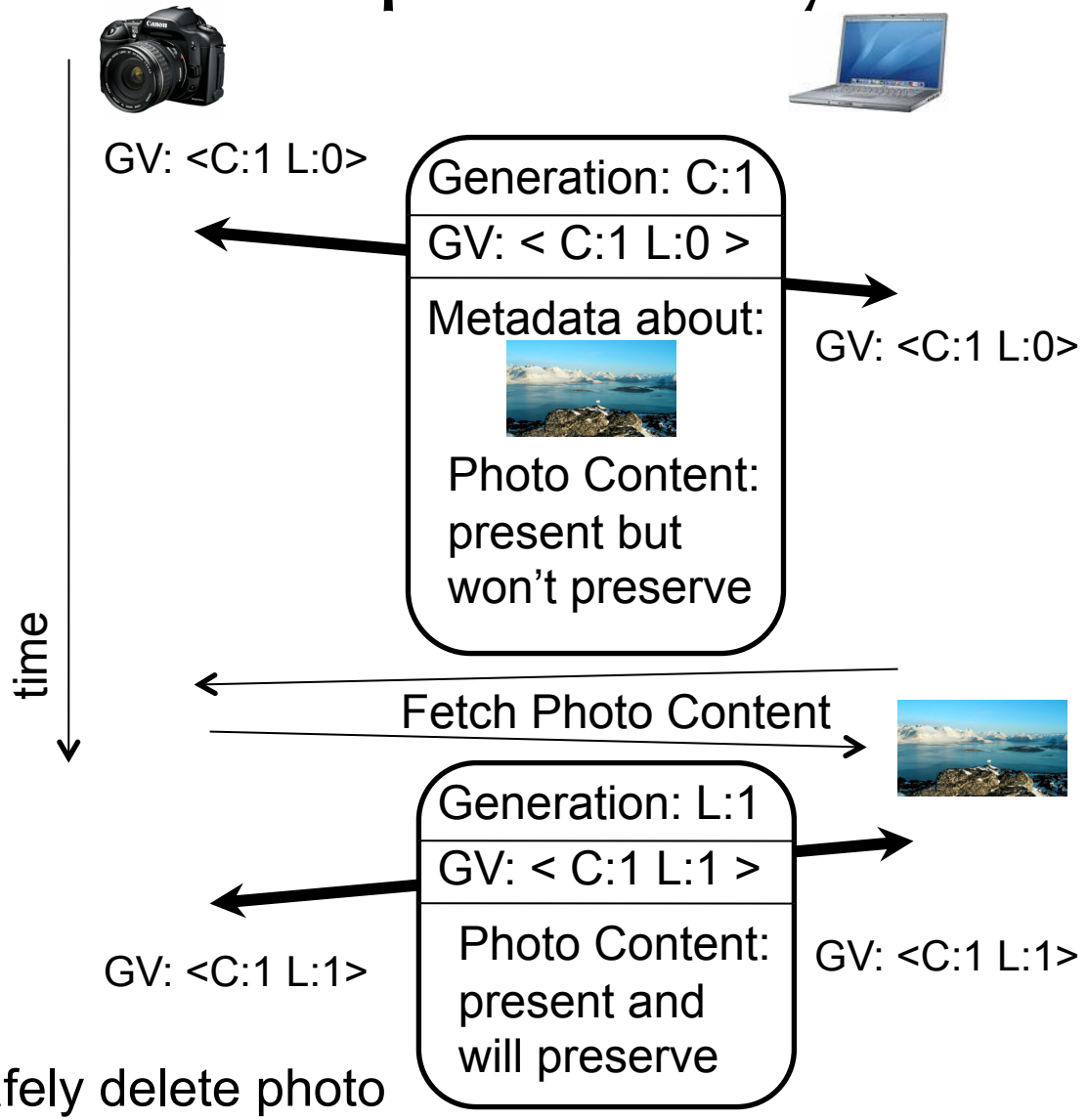
# Passing Content Responsibility

- Exchange responsibility for storing objects
- Does not rely on correct placement rules
- Guarantees at least one live copy
  - Assuming no lost or failed devices



# Passing Content Responsibility

- Exchange responsibility for storing objects
- Does not rely on correct placement rules
- Guarantees at least one live copy
  - Assuming no lost or failed devices




Camera can safely delete photo

# Eyo: Implementation

- Python per-device daemon
  - RPC for metadata sync
  - http for fetching content (no swarming yet)
- Python and C client libraries
  - Sqlite for metadata storage & queries
  - D-bus for event notifications
- UIA for group identity and communication
  - Users create a group of “my” devices
  - Tracks current locations, builds overlay network
  - Authenticates & Encrypts communication

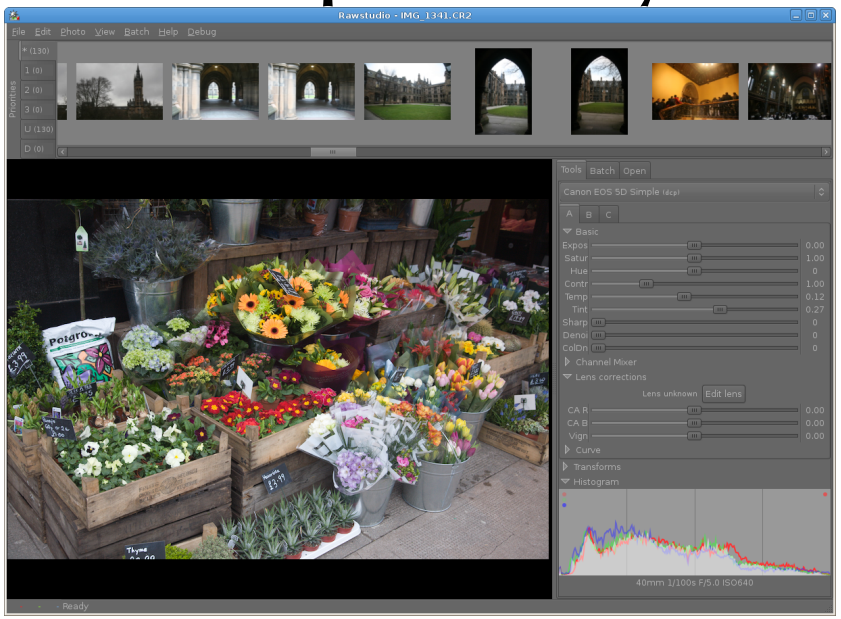
# Evaluation Questions

- What can we do with Eyo that we couldn't do otherwise?
- Is Eyo's API a good fit for real applications?
  - How difficult is adapting applications?
  - Usability of explicit version histories?
- Is the metadata-everywhere model feasible?
  - Storage costs?
  - Bandwidth overhead?

# Evaluation Approach

- Modify applications to use Eyo
  - Rhythmbox & Quodlibet media players
  - gPodder podcast manager
  - IMAP email gateway
  - Rawstudio photo editor
- Examine new features & scope and types of changes needed

# New Ability: Device Transparency




- From a disconnected device
- Browse and organize the entire collection
  - Search for tags
  - View thumbnail images
  - Add and edit tags for all images
  - Show which devices hold objects
- View and edit locally-cached full size image originals

# Few Application Changes Needed

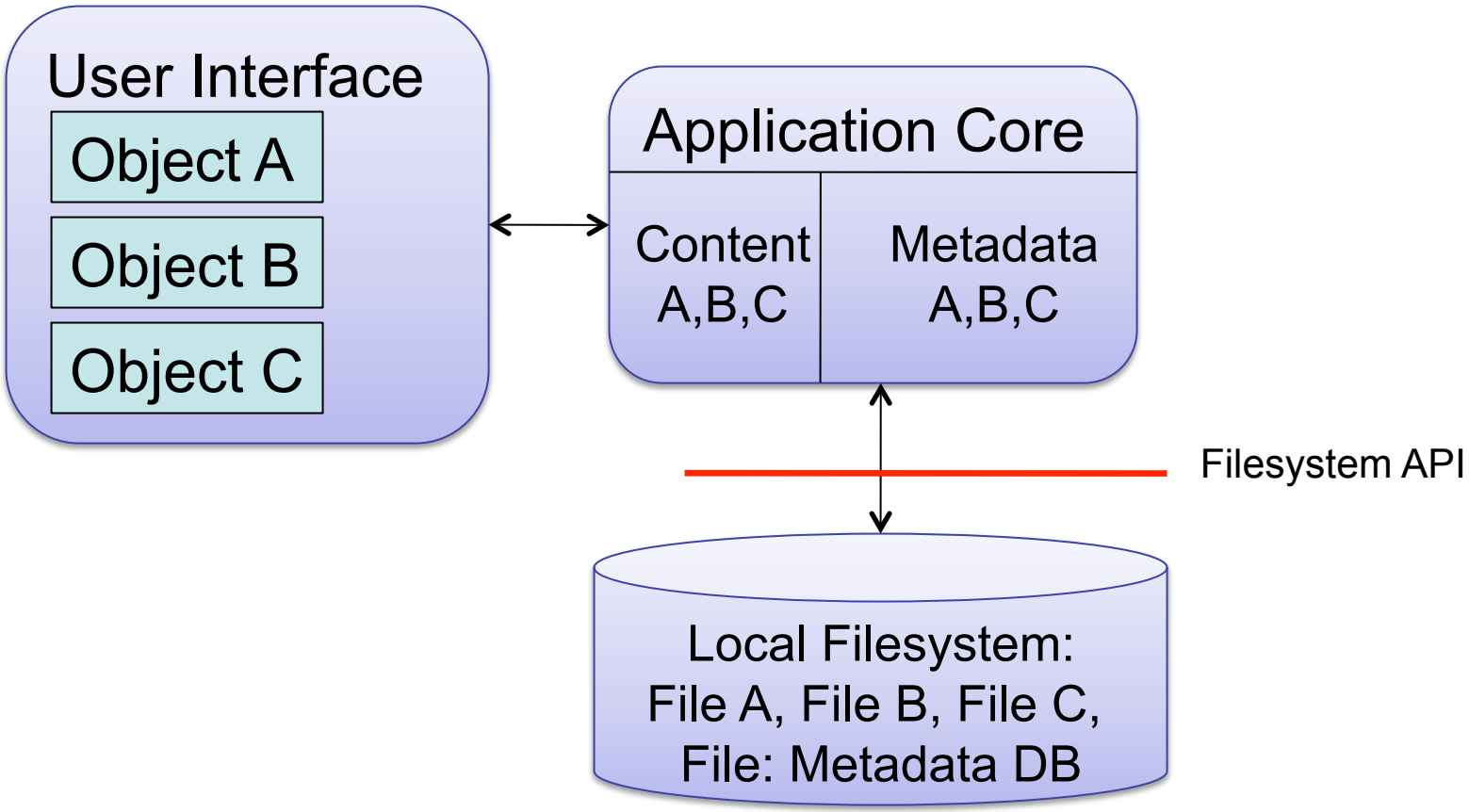
- Rawstudio photo editor (C & C++):

| Original #lines of code | Eyo Version #lines | #lines added | #lines removed |
|---|---|---|---|
| 59,767 | 59,851 | 1851 (~3%) | 1596 (~3%) |

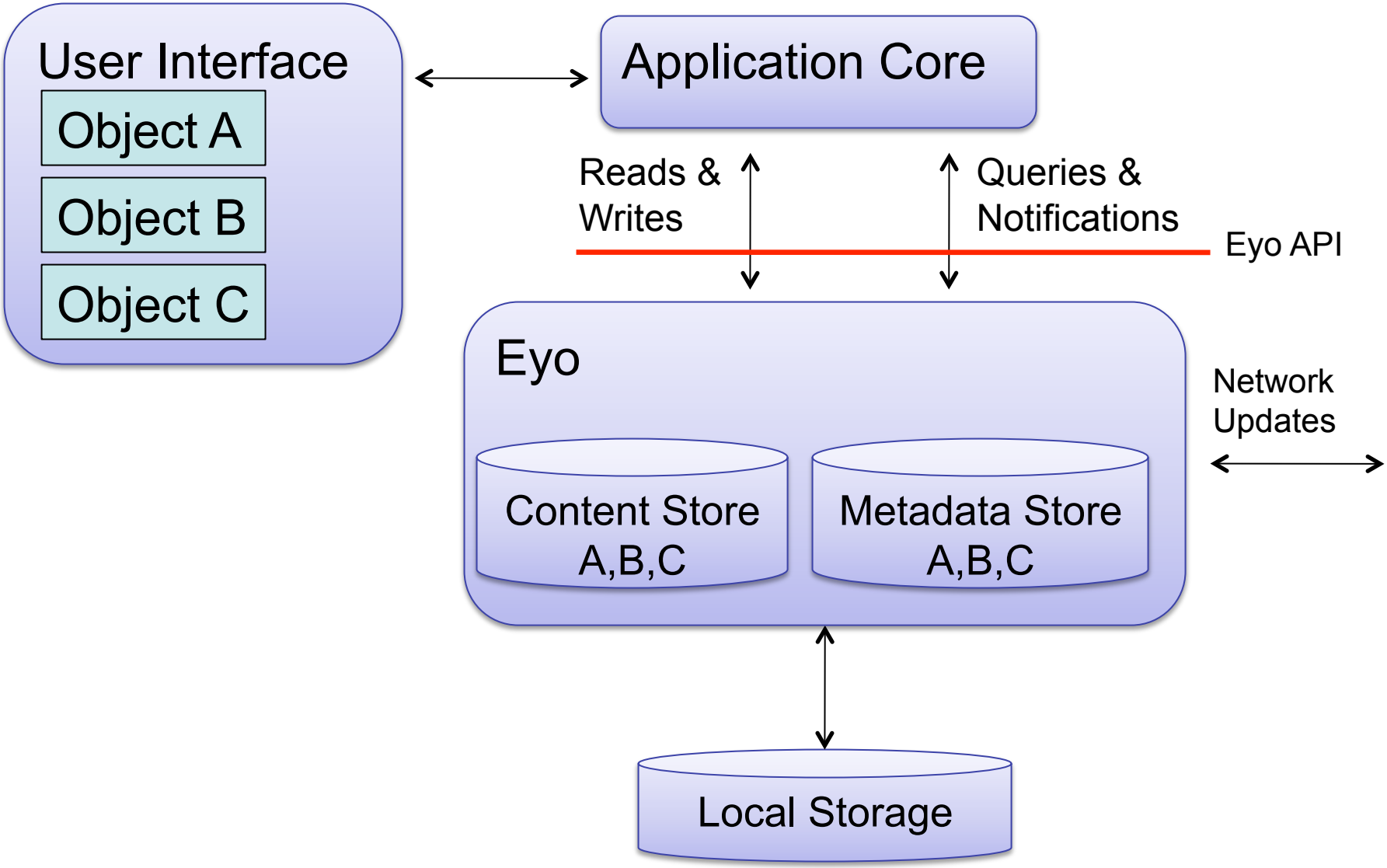
Different 'line' definitions

  - No User Interface changes in these values
- Remaining example applications:
  - Changes limited to <10% of codebase

# Applications already have Metadata split



# Eyo API Makes Split Explicit



# Metadata Storage Cost

- How much metadata?
- Look at one personal collection:

| | # objects | total size | Metadata per object |
|---|---|---|---|
| Email | 724,230 | 4.3 GB | 245B |
| Music | 5,278 | 26 GB | 511B |
| Photos | 72,380 | 122.8 GB | 328B |

Not very different

# Storage Costs: Reasonable for portable devices

- Store collections in Eyo
- Look at resulting metadata size

| | # objects | total size | Eyo metadata store size |
|---|---|---|---|
| Email | 724,230 | 4.3 GB | 529 MB |
| Music | 5,278 | 26 GB | 5.8 MB |
| Photos | 72,380 | 122.8 GB | 53 MB |

Total: <600MB, mostly from email

# Related Work

- Optimistic Replication
  - Cimbiosys, Perspective
  - Coda, Ficus, Bayou, PRACTI, EnsemBlue, Tierstore, Podbase, Ivy
- Point-to-point replication: Rsync, Unison
- Version Control Systems
  - Git, SVN
- Centralized Cloud Topologies
  - MobileMe/iCloud, Gmail/Gears, LiveMesh

# Summary

- Device Transparency
- View and mange complete collection
  - From disconnected, storage limited devices
- Eyo
  - Storage API with explicit version histories
  - Continuous peer-to-peer synchronization
  - Good fit for existing applications