Eyo

Device-Transparent Personal Storage

Based on the article „Eyo: Device Transparent Personal Storage” by Jacob Strauss, Justin Mazzola Paluska, Chris Lesniewski-Laas, Bryan Ford, Robert Morris, Frans Kaashoek

Outline

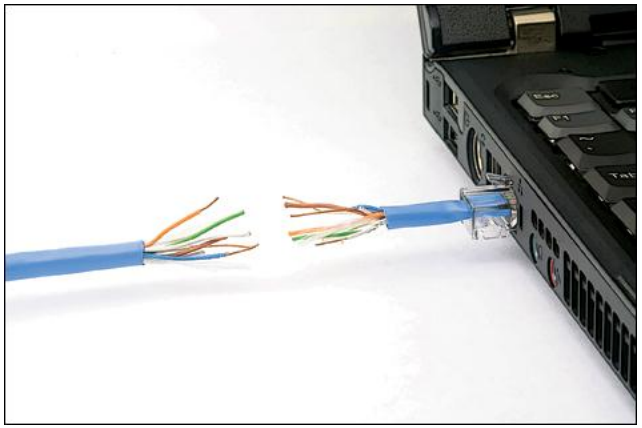
- Eyo's assumptions and API
- Main features
  - Placement rules
  - Version histories
  - Conflicts management
- Continuous synchronization
- Implementation
- Evaluation of the system

What is a device-transparent storage?

- Single view of entire data collection from every user's (personal) device
- A user can think in terms of „file X" rather than **"file X on device Y"**
- More than *location transparency* in traditional distributed systems (why?)

Challenges

- Mobile devices can't store entire data collection
- Concurrent updates from **disconnected** devices result in conflicts



Eyo

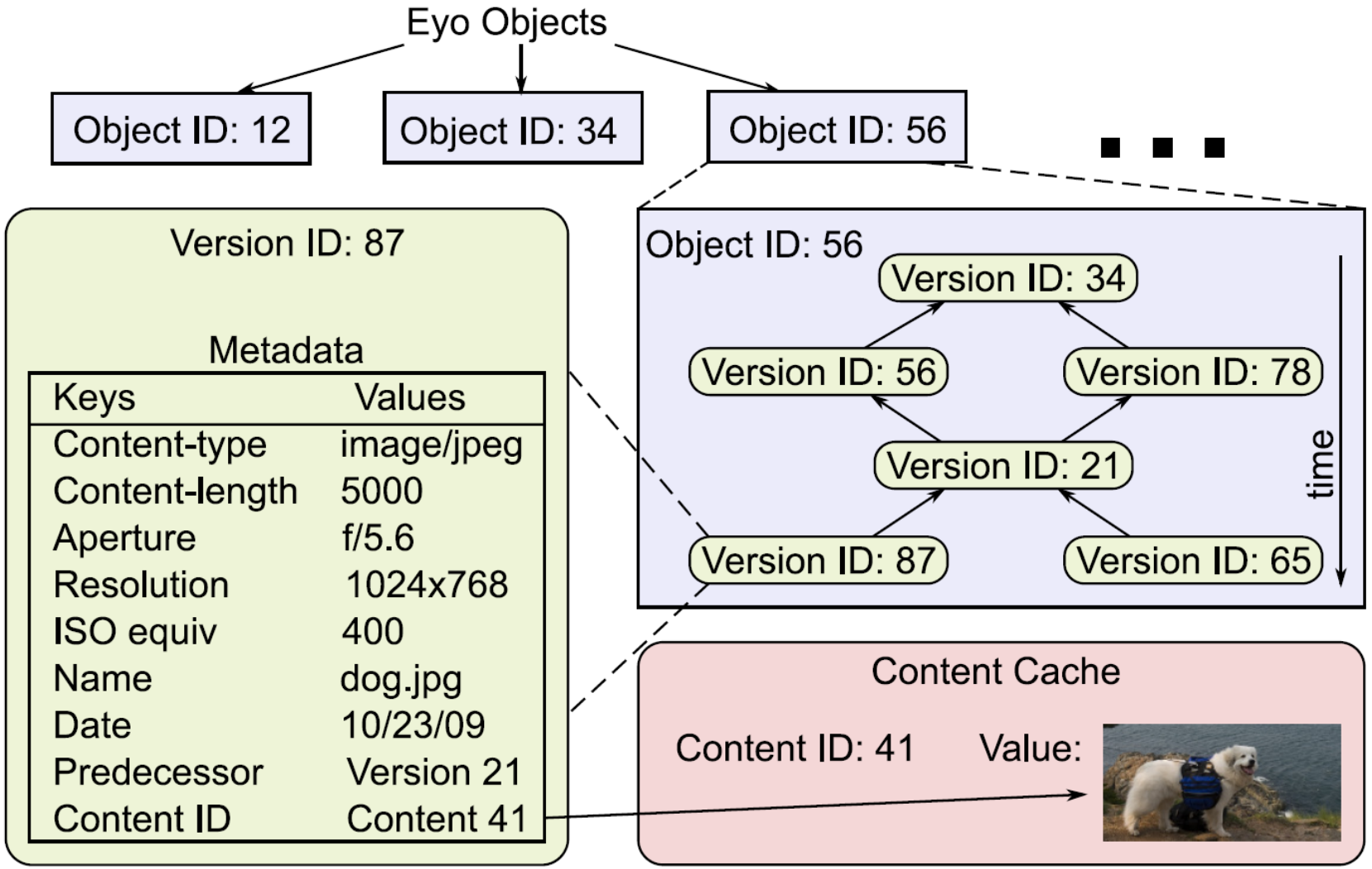
- Created by researchers from:
  - MIT
  - Yale University
  - Quanta Research Cambridge
- New personal storage system
  - Device-transparent
  - Design and storage model motivated by disconnected devices
  - Supports limited-storage devices
  - Peer-to-peer communication

Assumptions

- **Separate metadata from content**
- **Metadata is small enough to replicate on every device**
- Updates of objects content are rare
- Updates of metadata, inserting and deleting objects are common
- Metadata for common data types are specified
  - e.g. ID3, EXIF, email headers

Objects, metadata and content

- User's data stored as a flat set of versioned objects
- Each object is a directed acyclic graph of its versions
- Each version has assigned metadata, content and list with IDs of its ancestors


Eyo Objects
Object ID: 12
Object ID: 34
Object ID: 56
Version ID: 87
Metadata
Keys
Values
Content-type
image/jpeg
Content-length
5000
Aperture
f/5.6
Resolution
1024x768
ISO equiv
400
Name
dog.jpg
Date
10/23/09
Predecessor
Version 21
Content ID
Content 41
Object ID: 56
Version ID: 34
Version ID: 56
Version ID: 78
Version ID: 21
Version ID: 87
Version ID: 65
time
Content Cache
Content ID: 41
Value:

Objects, metadata and content

create(*ID hint*) → (*objectID*, *versionID*)

lookup(*query*) → list<(*objectID*, *versionID*)>

getVersions(*objectID*) → list<*versionID*>

getMetadata(*objectID*, *versionID*) → list<(*key*,*value*)>

open(*objectID*, *versionID*) → *contentID*

read(*contentID*, *offset*, *length*) → *contents*

newVersion(*objectID*, list<*versionID*>,
metadata, *contents*) →*versionID*

deleteObject(*objectID*) → *versionID*

Queries

- Application retrieves objects via queries on metadata:
  - lookup() – single search query
  - addWatch() – persistent query (conflicts resolution)
- Eyo uses subset of SQL
  - Efficient to execute
  - Limited in expressivness
  - Unsupported query: 10 most viewed pictures
  - Supported: pictures viewed more than 100 times

Placement rules

- Determines which object has highest priority for storage
- Important on devices with limited storage
- Eyo stores rules as objects without content
  - Rules are synchronized like other metadata
- Synchronization ensures that at least one copy of content is stored even if no placement rule matches

Placement rules

addRule(*name*, *query*, *devices*, *priority*) $\rightarrow$ *ruleID*

getRule(*name*) $\rightarrow$ (*ruleID*, *query*, *devices*, *priority*)

getAllRules() $\rightarrow$ list$<$(*ruleID*, *query*, *devices*, *priority*)$>$

removeRule(*ruleID*)

Conflicts management

- Application is responsible for handling concurrent updates using Eyo's API
- Version graphs
  - indicate most recent common ancestor
- Event notifications

addWatch(*query*, *watchFlags*, *callback*) → *watchID*

removeWatch(*watchID*)

callback(*watchID*, *event*)

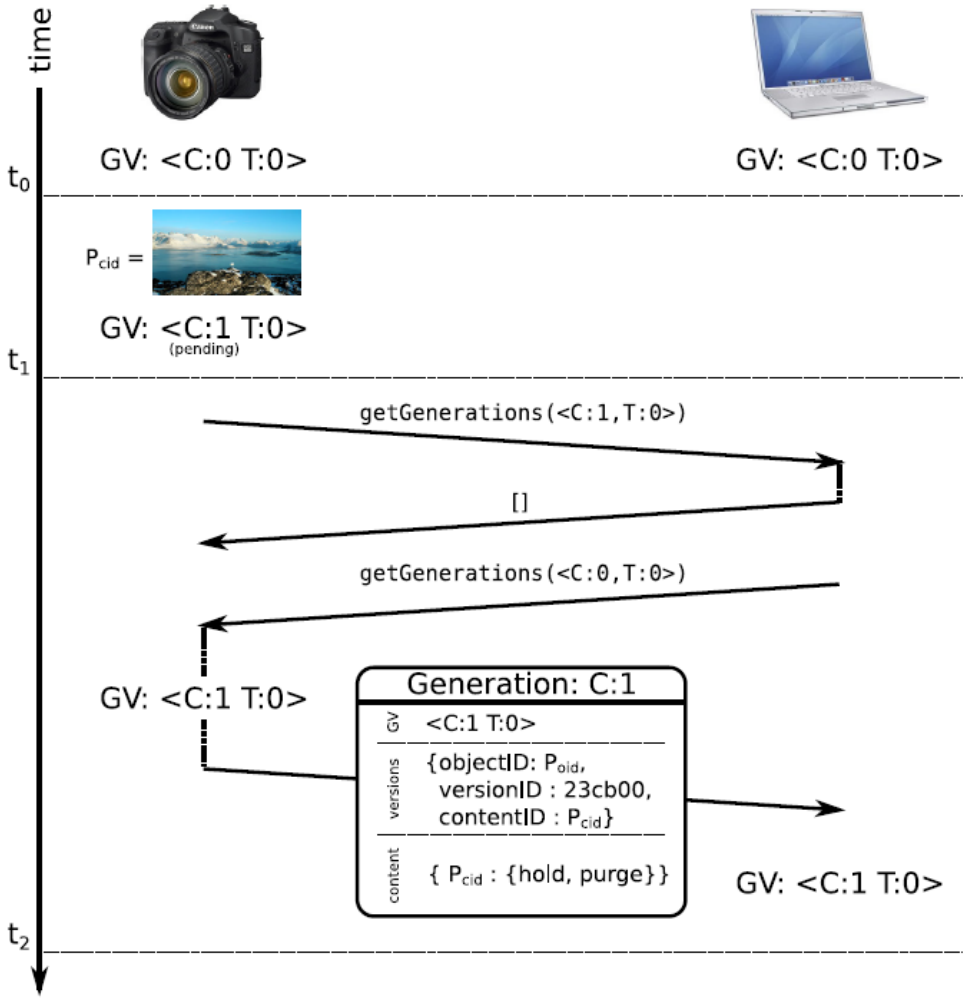
Continuous synchronization

- Reduces update conflicts by propagating changes as fast as network allows
- Synchronization start right after update
  - pending updates structure
- **Metadata synchronization**
  - device-transparency effect
- **Content synchronization**

Metadata synchronization

- Metadata is usually small and updates must be passed as fast as possible
- Eyo groups recent updates into generation, identified by device ID and generation counter on that device
- In the personal group of n devices, each device stores vector of n elements with information about latest generations received from other devices

Metadata synchronization (2)



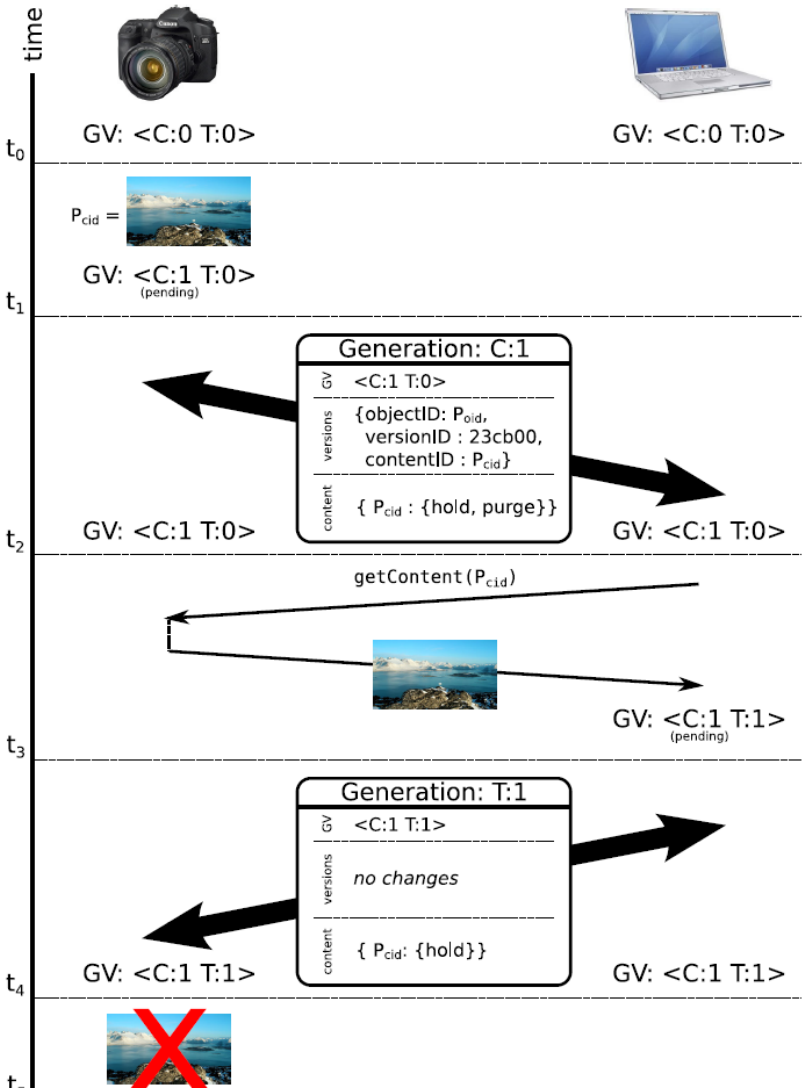
Metadata synchronization (3)

- **Versions graph truncation**
  - Eyo knows when generation objects have been seen by all devices (checking generation vectors)
  - Eyo can combine contents of those generation objects into single archive and truncate the version history
- **Adding and removing devices**
  - new device sends getGenerations() request with missing elements and gets complete copy of all generations

Content synchronization

- Device keeps a sorted list of content object to fetch
- Eyo uses the global distribution of metadata to track where is the content
- Metadata flags:
  - hold – device wants to store the content
  - purge – device wants to delete the content

Content synchronization (2)



Implementation

- **Eyo daemon**
  - written in Python
  - libraries for C applications
  - runs on each participating device
  - handles external communication
  - runs on Linux and Mac OS X
  - connects to other devices via UIA
- **SQLite (metadata storage)**
- **Local file systems (content storage)**
- **XML-RPC**
- **UIA** (Unmanaged Internet Architecture) – global connectivity architecture for mobile devices

Evaluation

Adapting existing apps to use Eyo

Size (lines)	Rawstudio	Rhythmbox	QuodLibet	gPodder	Email
total project	59,767	102,000	16,089	8,168	3,476
module size	6,426	9,467	428	426	312
lines added	1,851	2,102	76	295	778
lines deleted	1,596	14	2	2	N/A
language	C	C	python	python	python
content	←	individual files		→	N/A
metadata	central DB, sidecar files	←	central DB	→	N/A

Adapting existing apps to use Eyo (2) – conflicts resolution

Application	Type	User-Visible Conflicts Possible?	Why?
IMAP	Email Gateway	No	Boolean flag changes only
gPodder	Podcast Manager	No	User cannot edit metadata directly
Rhythmbox	Media Player	Yes	Edit Song title directly
QuodLibet	Media Player	Yes	Edit Song title directly
Rawstudio	Photo Editor	Yes	Edit settings: contrast, exposure...

Metadata storage costs

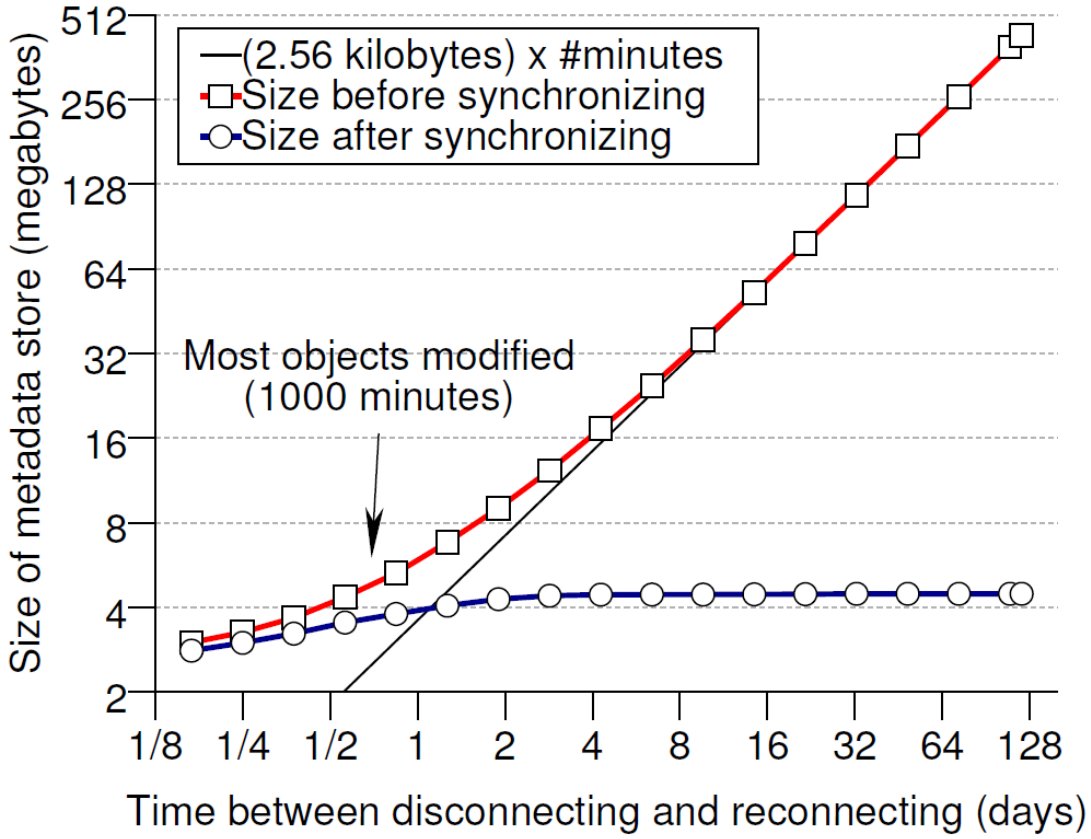
	Email	Music	Photos
number of objects	724230	5299	72380
total content size	4.3 GB	26.0 GB	122.8 GB
native metadata size	169.3 MB	2.6 MB	22.6 MB
Eyo metadata size	529.6 MB	5.8 MB	52.9 MB
metadata/content overhead	12%	0.02%	0.04%
metadata store per object	766 bytes	1153 bytes	767 bytes

Bandwidth costs

- Lack of information in the paper
- Metadata-everywhere model
- If there is no placement rules then bandwith is used only for transferring metadata and protocol overhead (XML-RPC)

Disconnected devices

- Having disconnected devices result in increasing metadata size
- Version graphs cannot be truncated



Synchronization delay

System	Description
Unison	Delays of at least 1 second for small collections. Large collections take significantly longer: 23 seconds for an existing collection of 500K objects 87 seconds for 1M objects
MobileMe	Most updates arrive after between 5 and 15 seconds. Occasionally as long as 4 minutes. Delay does not depend on collection size.
Eyo	All delays fall between 5 and 15 milliseconds. Delay does not depend on collection size.

Other device-transparent storage systems

- Cimbiosys
  - supports placement rules
- Perspective
  - supports placement rules
  - disconnected devices cannot continue to see complete collection
- Coda
  - disconnected operations
  - consistency algorithms for file systems

Summary

- Eyo separates metadata from content
- Metadata-everywhere model
- Peer-to-peer communication
- Continuous synchronization
- Supports storage-limited mobile devices
  - placement rules
- Supports disconnected devices
  - concurrent updates resolutions
- Assumes many things about applications


QUESTIONS
?