Adaptation for Mobile Data Access

(DM1 & PL1)

Yerang Hur

Mar. 24, 1998

Outline

- Motivation
- Models of Adaptation
- Application-transparent Adaptation
  - Design and implementation
  - Evaluation
- Application-aware Adaptation
  - Design and implementation
  - Evaluation
- Conclusion and Future Work

Motivation

- Constraints of mobility
  - Lack of local resources and the physical security $\longrightarrow$ reliance on server
  - Variable network connectivity in bandwidth, latency, reliability, and cost $\longrightarrow$ reliance on client
- Mobility needs *adaptive* system to meet the intrinsic constraints.

Models of Adaptation

- Who is responsible for adaptation?
  - Individual applications (laissez-faire adaptation)
  - The system (application-transparent adaptation)
    - system provides resource arbitration.
    - existing applications continue to work even when mobile.
  - Both (application-aware adaptation)
    - application specific information is used while the system controls resources.

Application-aware

(ex. Odyssey)

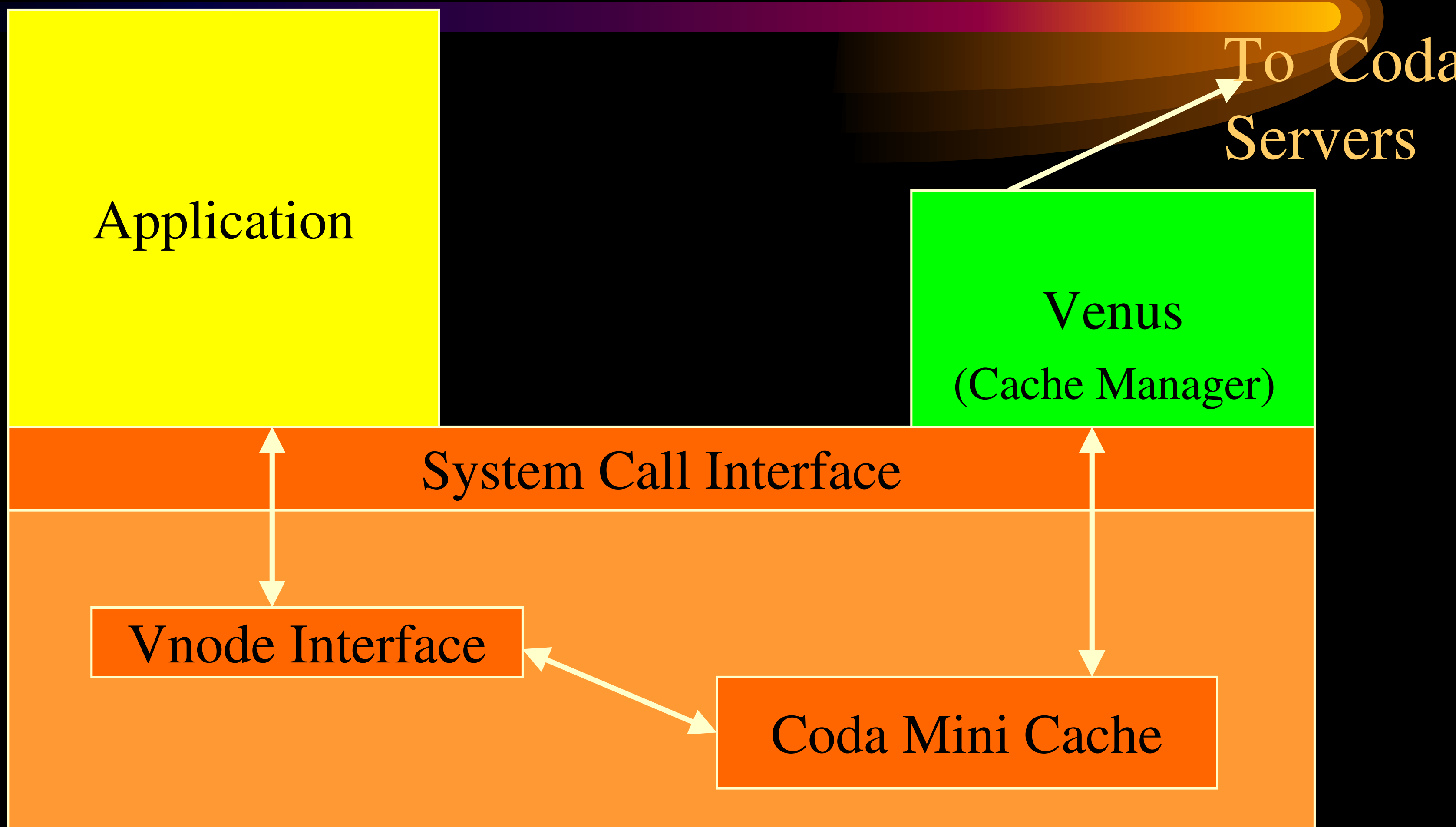
Laissez-faire

(ex. Eudora)

Application-transparent

(ex. Coda)

Application-transparent Adaptation (Coda)



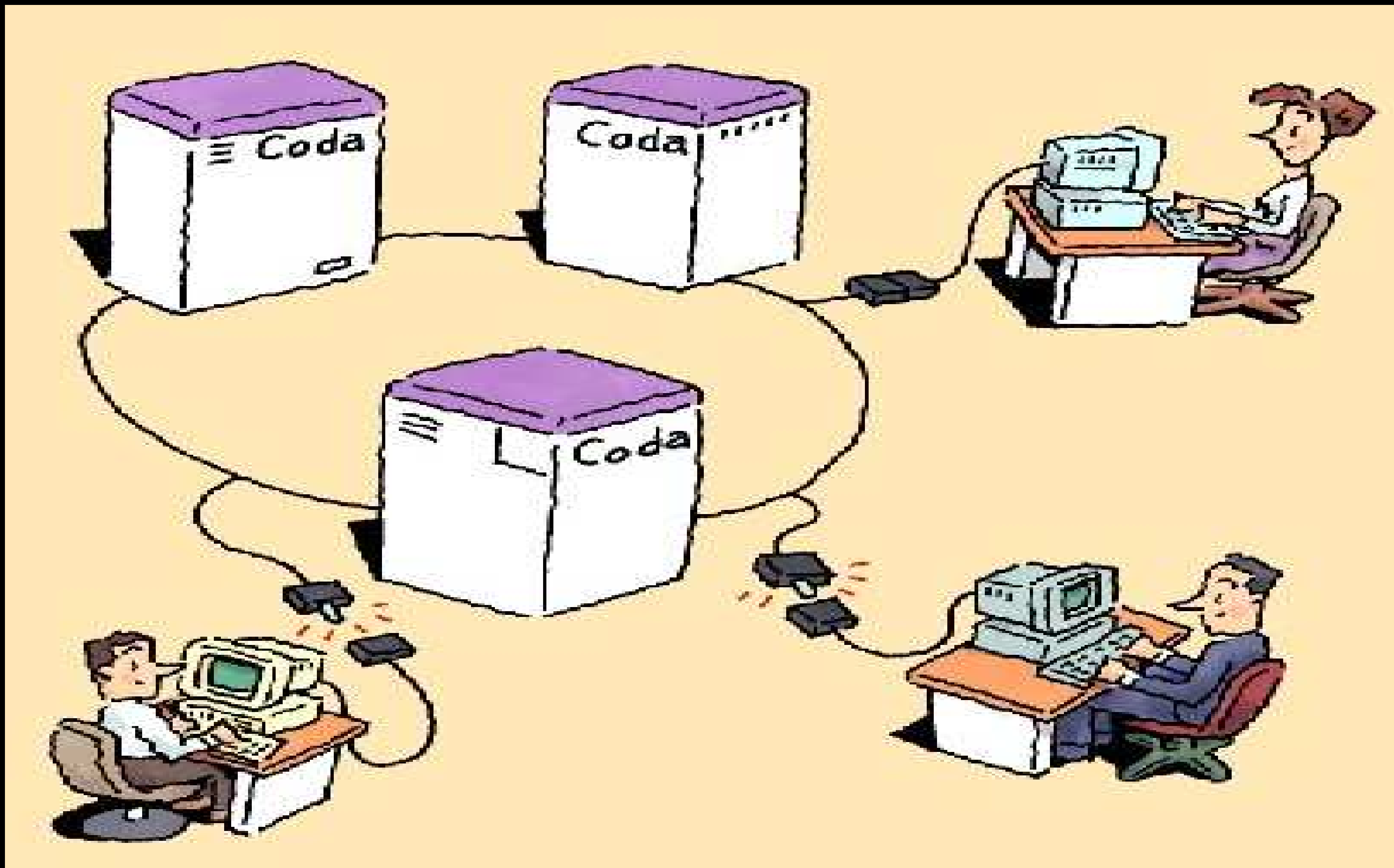


Illustration by Gaich Muramatsu

Design and Implementation

- Goal: high availability
  - Disconnected operation
    - while disconnected, Venus serves file system requests.
    - when disconnection ends, Venus propagates modifications.
  - Server replication
    - allows *volumes* to have replicas at more than one server.

Design rationale

- Scalability
  - callback-based cache coherence
    - servers notifies clients when their cached copies are no longer valid.
  - whole-file caching
    - when Venus fetches a file, it fetches the entire file from the servers.
    - cache misses can occur only when files are open.

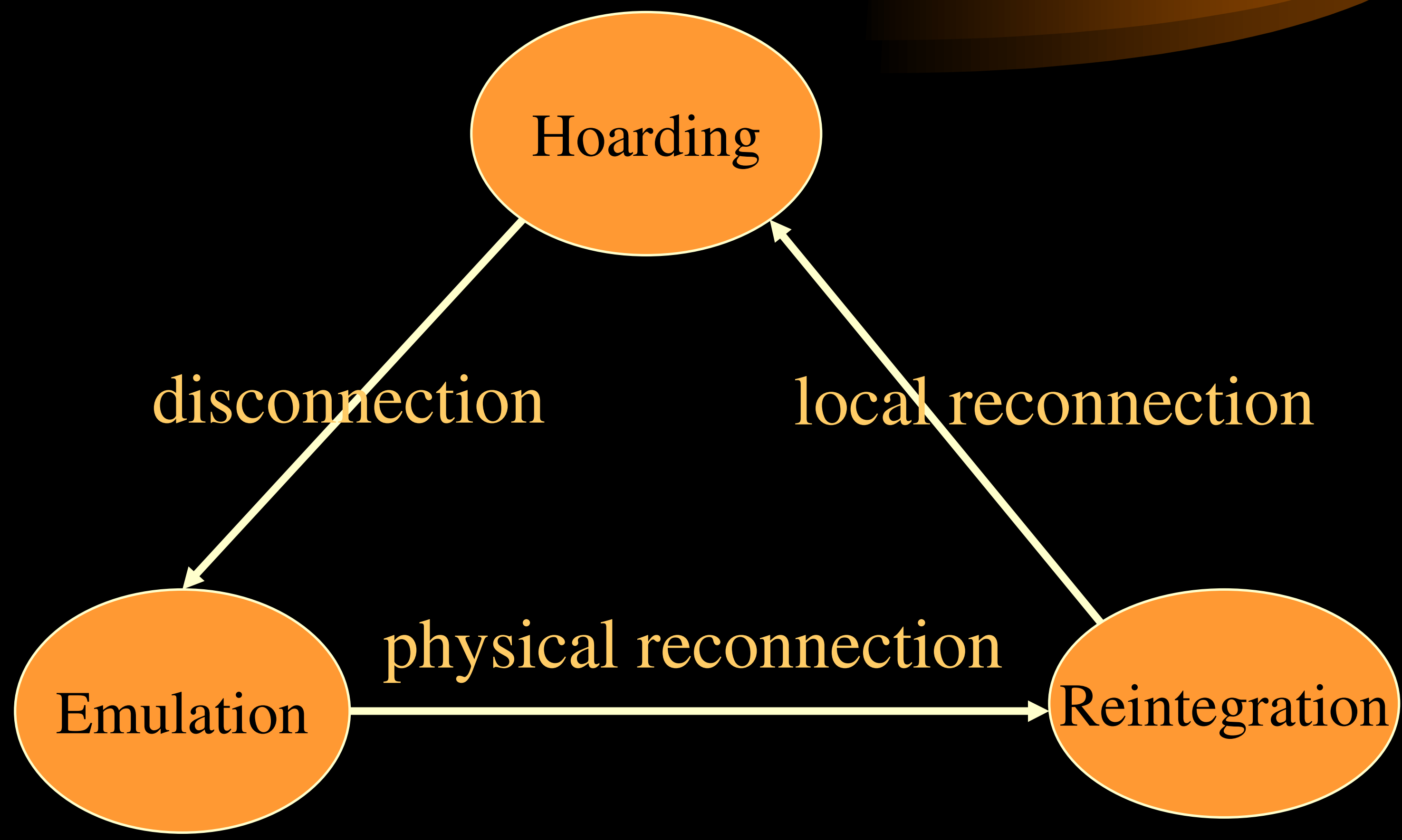
Design rationale

- Advent of portable workstation
  - disconnection
- Optimistic replica control
  - when a client is disconnected, the system permits reads and writes everywhere.
  - treats conflicts after their occurrence.
  - provides higher availability.

Ex. cat /coda/usr/jjk/foo

- open call is forwarded by the Vnode interface.
- When it is realized that the request is for a file in the /coda file system, it is handed to the Coda MiniCache in the kernel.
- When the MiniCache does not have usr/jjk/foo, the request is passed to Venus and Venus checks the client disk cache for usr/jjk/foo and in case of a cache miss, it contacts the servers to ask usr/jjk/foo.
- Venus enters a disconnected mode, when there is no network connection to any server.

Venus states and transitions



Hoarding

- Why?
  - Venus cannot serve a cache miss during a disconnection. important files should be kept in the cache up to date.
- Prioritized cache management
  - Users may give information on priority of files (HDB).
  - Recent reference history and HDB are used for hoarding.
- Hoard walking
  - updates the hoarded files.

Hoarding

- Hoard profiles

  # Personal files
  a /coda/usr/jjk d+
  a /coda/usr/jjk/papers 100:d+
  a /coda/usr/jjk/papers/sosp 1000:d+

  # System files
  a /usr/bin 100:d+
  a /usr/etc 100:d+
  a /usr/lib 100:d+

Hoard walking

- Goal: to meet *equilibrium.*
  - Cache is in *equilibrium* when no uncached file has a higher priority than a cached file.
  - Every 10 minutes or by users request.
- Phase 1
  - Name bindings of HDB entries are re-evaluated.
- Phase 2
  - Priorities in the cache and HDB are re-evaluated.

Emulation

- Venus takes the role of pseudo-server.
- Logging
  - records information for reintegration in a *replay log*.
    - logs a *store* record rather than logging the every *open, close*, and intervening *write* operation.
    - Discards a previous *store* record when a new one for the same file is appended to the log.
- Persistence
  - cached directory, replay logs, and the HDB is stored in nonvolatile storage.

Reintegration

- Venus propagates changes made during emulation and updates its cache. All activity is suspended till completion of reintegration.
- Replay algorithm
  - Clients
    - Venus obtains permanent file ids for new files.
    - Venus transfers the replay log to the servers.
  - Servers
    - parses the log and all files referenced in the log are locked. Transaction begins.
    - validates each object in the log and executes it.
    - transfers data.
    - commits the transaction and releases all locks.

Reintegration

- Conflict handling
  - During phase two of replay, a server compares the unique *storid* associated with that in a log entry.
  - If there is a conflict the entire reintegration is aborted.
    - Ex:
      - conflict in the case of a store of file
      - creating a new directory: conflicts occur if the name collides with an exiting name.
      - modification of attributes
  - Venus stores all replay information in its local disk when reintegration fails, and users can selectively replay it manually.

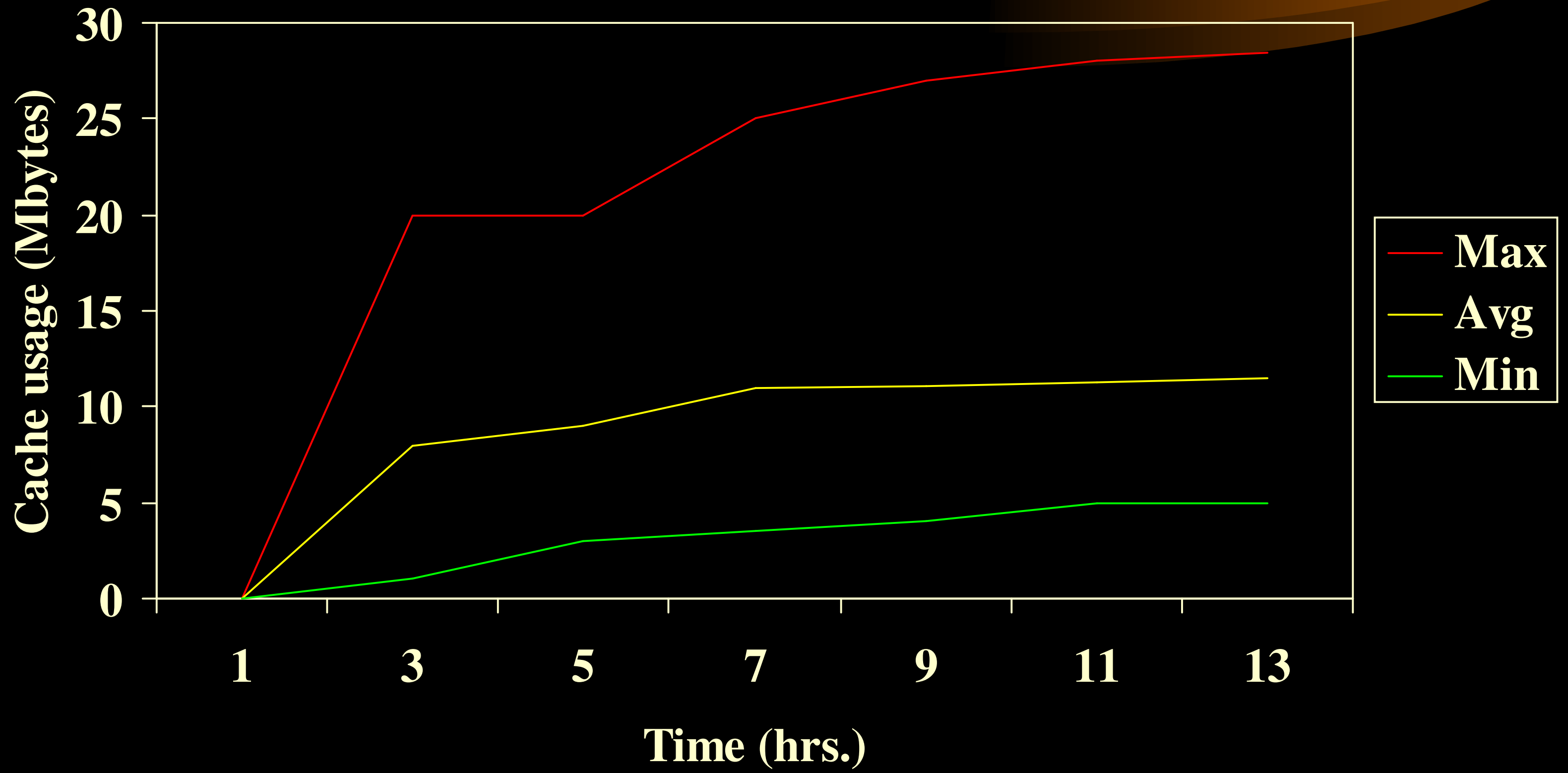
Evaluation

- Duration of reintegration
  - time to allocate permanent fids +
    time for the replay at the servers +
    time for server replication
- Cache size
- Likelihood of conflicts

Time for reintegration

	Elapsed time (sec.)	Reintegration time (sec.)				# of records in the replay log
		Total	Alloc Fid	Replay	Update	
Andrew benchmark	**288**	**43**	**4**	**29**	**10**	**223**
Venus make	**3,271**	**52**	**1**	**40**	**10**	**193**

Cache size



Cache usage (Mbytes)
30
25
20
15
10
5
0
1
3
5
7
9
11
13
Time (hrs.)
Max
Avg
Min

Likelihood of conflicts

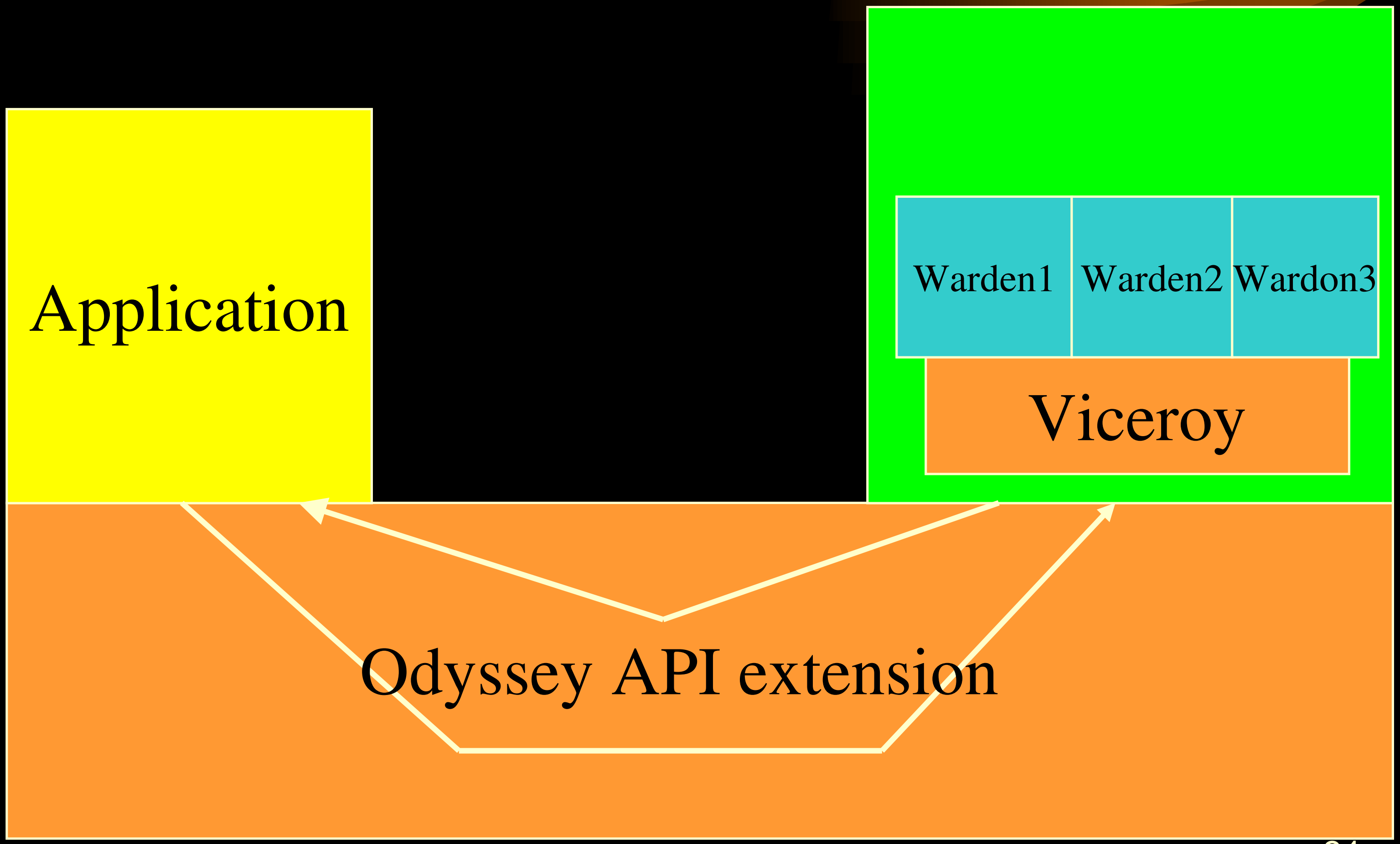
Type of volume	Type of object	Same user	Different user
User	Files	99.87%	0.13%
	Directories	99.80%	0.20%
Project	Files	99.66%	0.34%
	Directories	99.63%	0.37%
System	Files	99.17%	0.83%
	Directories	99.54%	0.46%

Summary

- Coda can support disconnect operation.
  - Ex: 100MB local disk, 50MB cache, one or two day disconnection

    → about 1 minute reintegration

Application-aware Adaptation (Odyssey)



Design and implementation

- Goal: Collaborative partnership between the operating system and applications
  - Adaptation is the key to mobility.
    - unpredictable variation in network quality
    - disparity in the availability of remote services
    - limitations on local resources
  - Odyssey monitors resources, interacts with each application, and applications decide the best adaptation when notified.

Design rationale

- Fidelity
  - Degree to which data presented at a client matches the reference copy at the server.
    - video data: frame rate and image quality
    - telemetry data: sampling rate and timeliness
- Concurrency
  - Ability to execute multiple independent applications concurrently on a mobile client.
- Agility
  - Speed and accuracy with which it detects and responds to changes in resource availability
    - The larger change is, the more important the agility is.

Viceroy and wardens

- Viceroy
  - Centralized resource management
    - monitoring the availability of resources
    - notifying applications of changes
- Wardens
  - Type-specific operations to change the fidelity
  - Responsible for communicating servers and caching data

Expressing resource expectation

- Generic resources in Odyssey
  - network bandwidth, network latency, disk cache space, CPU, battery power
- Resource negotiation operations
  - **request (in path, in resource-descriptor, out request-id)**
  - **cancel (in request-id)**
  - resource descriptor
    - resource-id, lower bound, upper bound, name of upcall handler
- Upcall handler
  - **handler (in request-id, in resource-id, in resource-level)**
- TSO (Type-Specific Operations)
  - **tsop (in path, in opcode, in insize, in inbuf, inout outsize, out outbuf)**

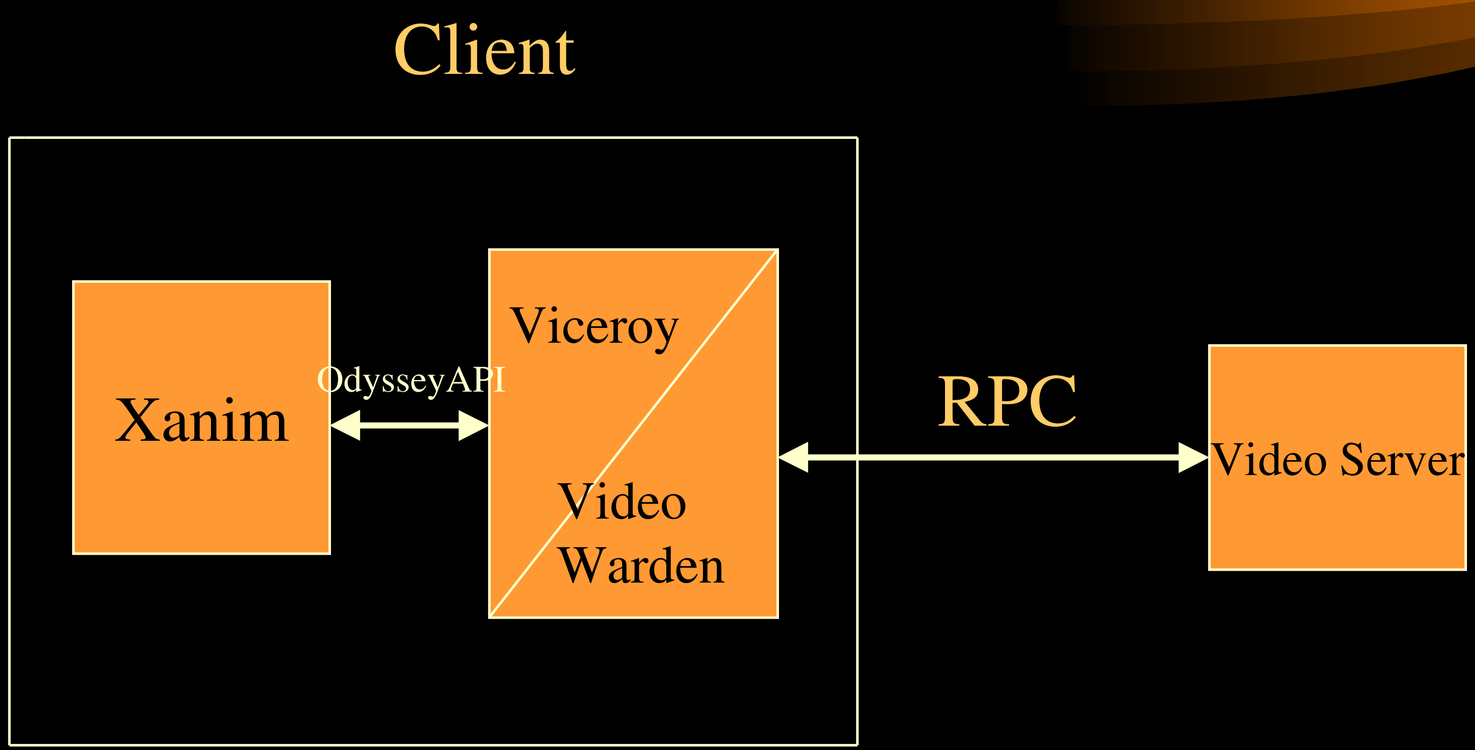
Notifying applications

- Viceroy generates an **upcall** to the corresponding application
- Application adjusts its fidelity according to its policy with the **resource-level**.

Example applications

- Video player
- Web browser
- Speech recognizer

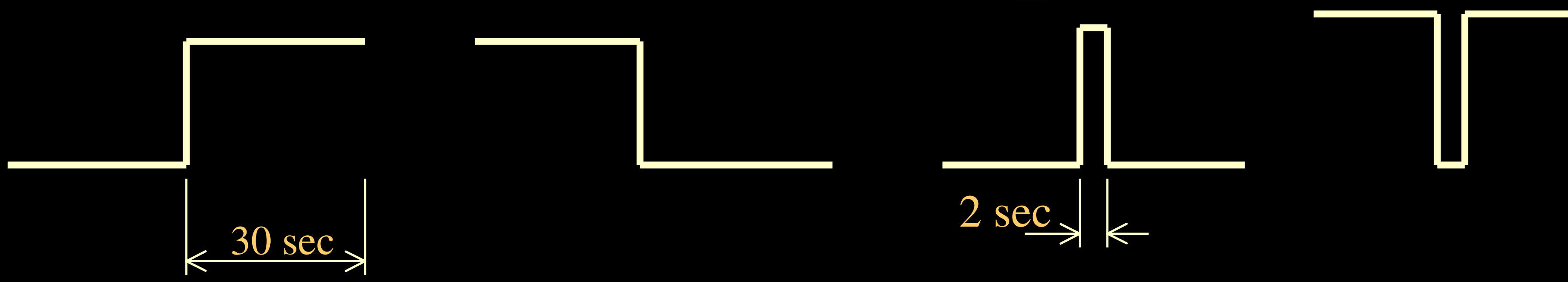
Video player



Three fidelity levels: color frames at quality 99 and 50, and b/w frames

Evaluation

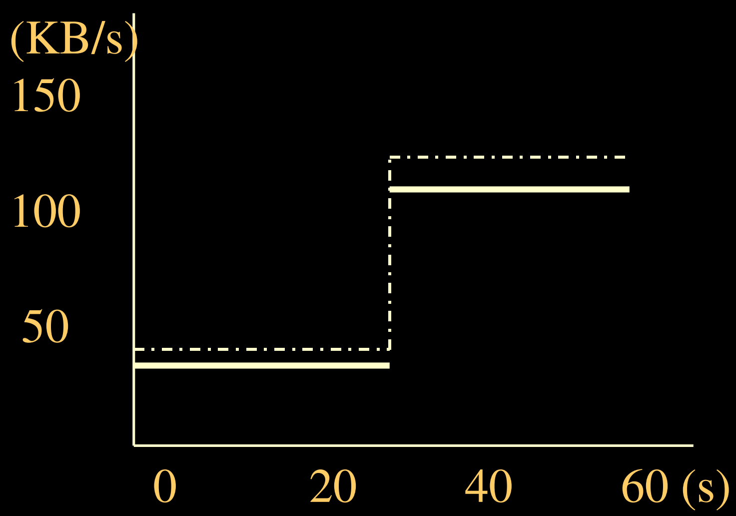
- Reference waveforms for agility experiments

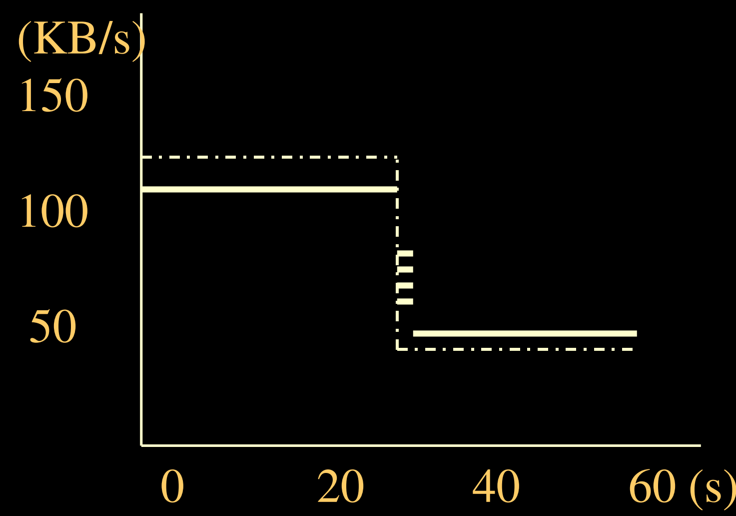


- 90 MHz Pentium client and 200MHz Pentium Pro servers
- Customized NetBSD 1.2

Results

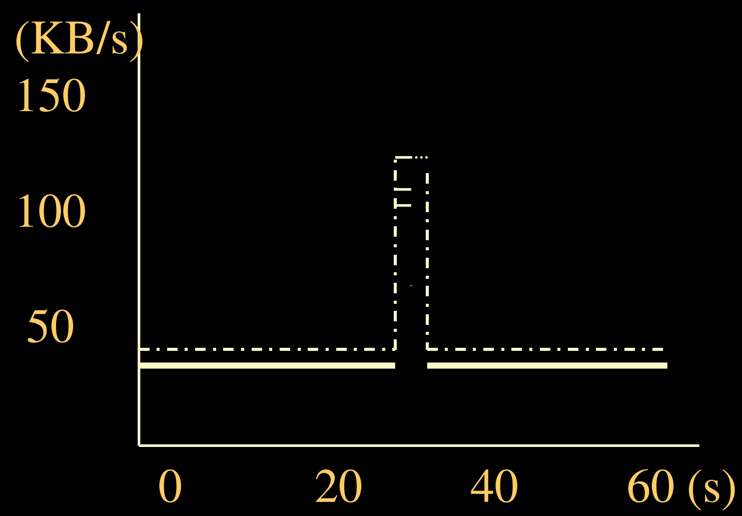
Agility (supply estimation agility)

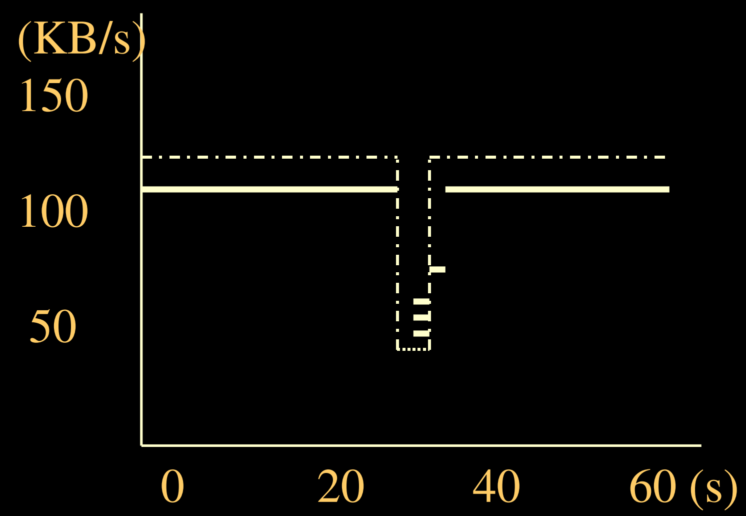




Results

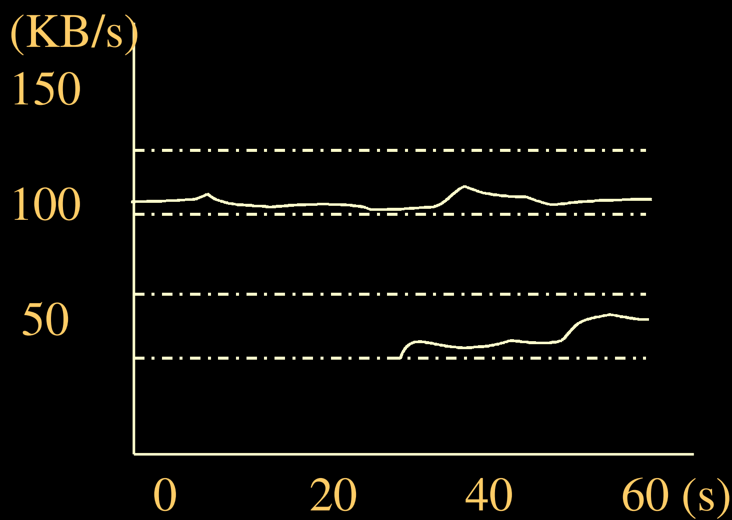
Agility (supply estimation agility)



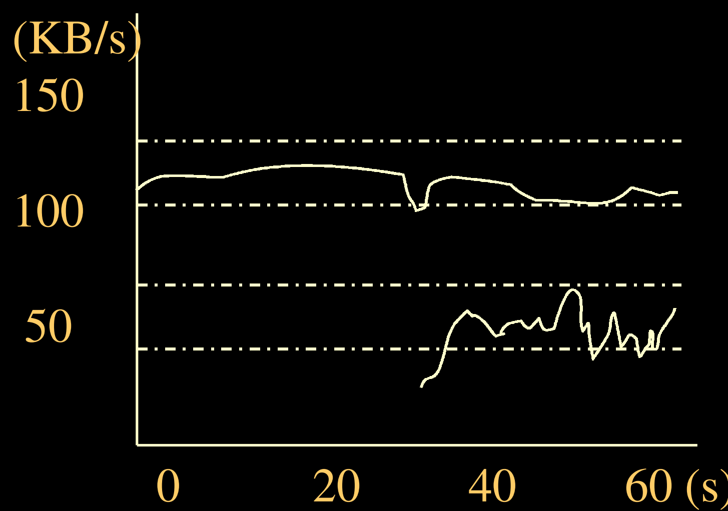


Results

Agility (demand estimation agility)

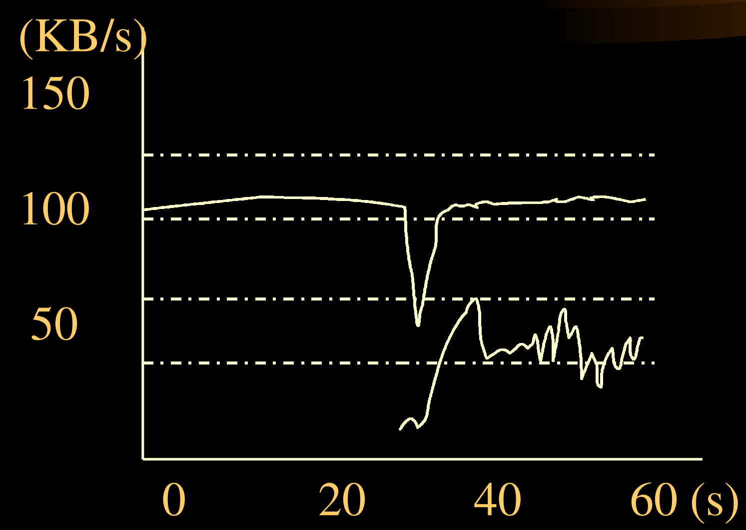


10% utilization/stream



45% utilization/stream

Results



100% utilization/stream

Results

- Effect of adaptation (performance and fidelity)

Waveform	B/W Drops	B/W Fidelity	JPEG(50) Drops	JPEG(50) Fidelity	JPEG(99) Drops	JPEG(99) Fidelity	Odyssey Drops	Odyssey Fidelity
Step-Up	0	0.01	3	0.5	7	1.0	7	0.73
Step-Down	0	0.01	5	0.5	25	1.0	25	0.76
Impulse-Up	0	0.01	3	0.5	23	1.0	23	0.50
Impulse-Down	0	0.01	0	0.5	14	1.0	14	0.98

Results

Effect of centralized resource management

	Video Drops	Video Fidelity
Odyssey	1018	0.25
Laissez-faire	2249	0.39
Blind optimism	5320	0.80

Conclusion and Future Work

- Need for adaptation in mobile information access is widely accepted $\longrightarrow$ application-aware adaptation
- They should apply resource management to other resources
- Multiple fidelity levels for other applications should be supported (ex. Speech recognizer).
- Systematic principles for adaptive mobile systems would be valuable.

References

- J. J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. ACM Transactions on Computer Systems, Vol. 10, No. 1, Feb. 1992, pp. 3-25.
- M. Satyanarayanan et al. Coda: a Highly Available File System for a Distributed Workstation Environment. IEEE Transactions on Computers. Vol. 39, No. 4, April 1990, pp. 447-459.
- B. D. Noble et al. Agile Application-Aware Adaptation for Mobility. In Proceedings of the 16th ACM Symposium on Operating System Principles. Oct. 1997.
- B. D. Noble, M. Price, and M. Satyanarayanan. A Programming Interface for Application-aware Adaptation in Mobile Computing. In Proceedings of the 1995 USENIX Symposium on Mobile and Location-Independent Computing. April 1995.