MOBILE COMPUTING

CSE-4225

Fall-2019

Mobile File System (MFS) - Motivation

- **Goal**
  - efficient and transparent access to shared files within a mobile environment while maintaining data consistency
- **Problems**
  - limited resources of mobile computers (memory, CPU, ...)
  - low bandwidth, variable bandwidth, temporary disconnection
  - high heterogeneity of hardware and software components (no standard PC architecture)
  - wireless network resources and mobile computer are not very reliable
  - standard file systems (e.g. NFS) are very inefficient, almost unusable

MFS - Motivation

- **Solutions**
  - replication of data (copying, cloning, caching)
  - data collection in advance (hoarding, pre-fetching)

MFS - consistency problems

- A central problem of distributed, loosely coupled systems
  - are all views on data the same?
  - how and when should changes be propagated to what users?
- Strong consistency
  - many algorithms offering strong consistency like in database systems (via atomic updates) cannot be used in mobile environments
  - invalidation of data located in caches through a server is very problematic if the mobile computer is currently not connected to the network

File systems - consistency problems

- Weak consistency
  - occasional inconsistencies have to be tolerated, but conflict resolution strategies must be applied afterwards to reach consistency again
- Conflict detection
  - content independent: version numbering, time-stamps
  - content dependent: dependency graphs

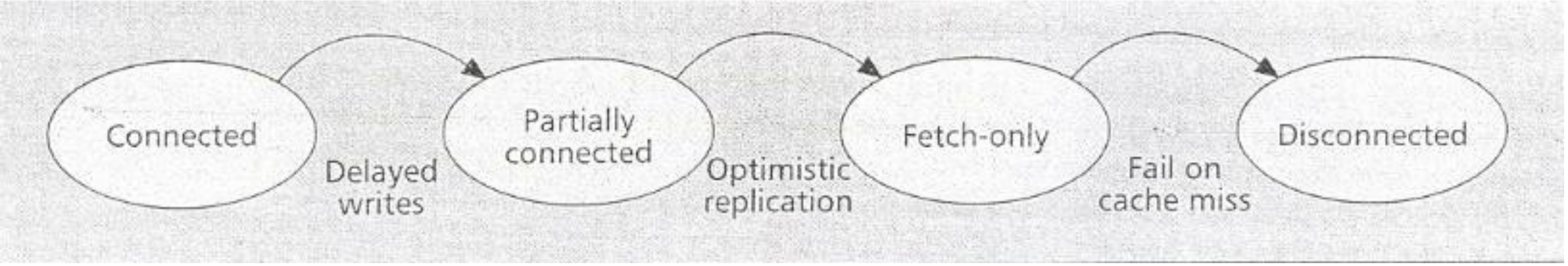
MFS variables

- Client/Server or Peer-to-Peer relations
- Support in the fixed network and/or mobile computers
- One file system (or namespace) or several file systems
- Transparency
  - hide the mobility support, applications on mobile computers should not notice the mobility
  - user should not notice additional mechanisms needed
- Optimistic or pessimistic consistency model
- Caching and Pre-fetching
  - bytes, paragraphs, single files, directories, subtrees, partitions, ...
  - permanent or only at certain points in time
- Data management
- Conflict solving

MFS Characteristics

- Provide location transparency
- Provide replication (optimistic/pessimistic)
- Provide cache consistency
- Provide connected and disconnected operational modes
- Provide scalability

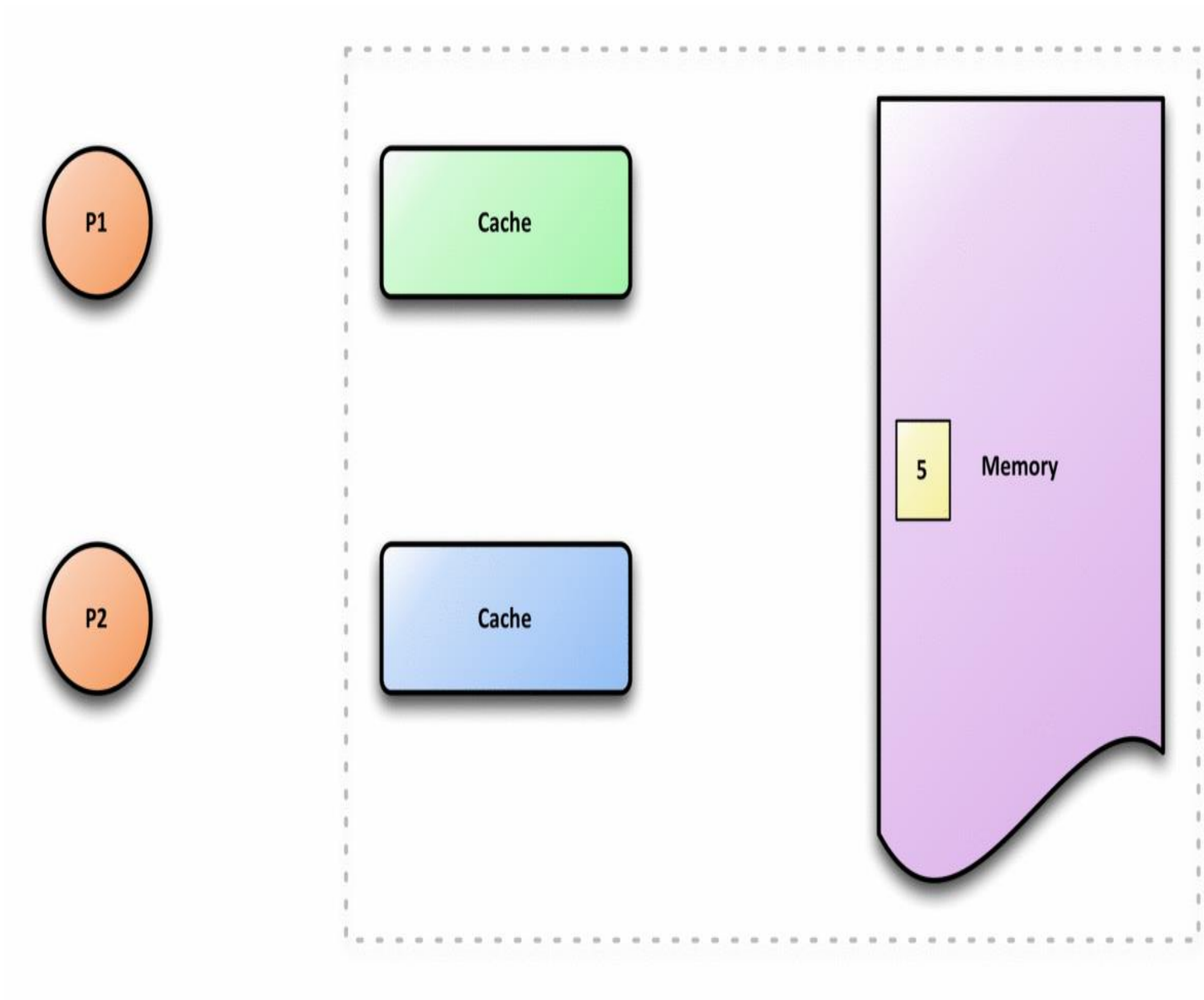
MFS Modes of Operation



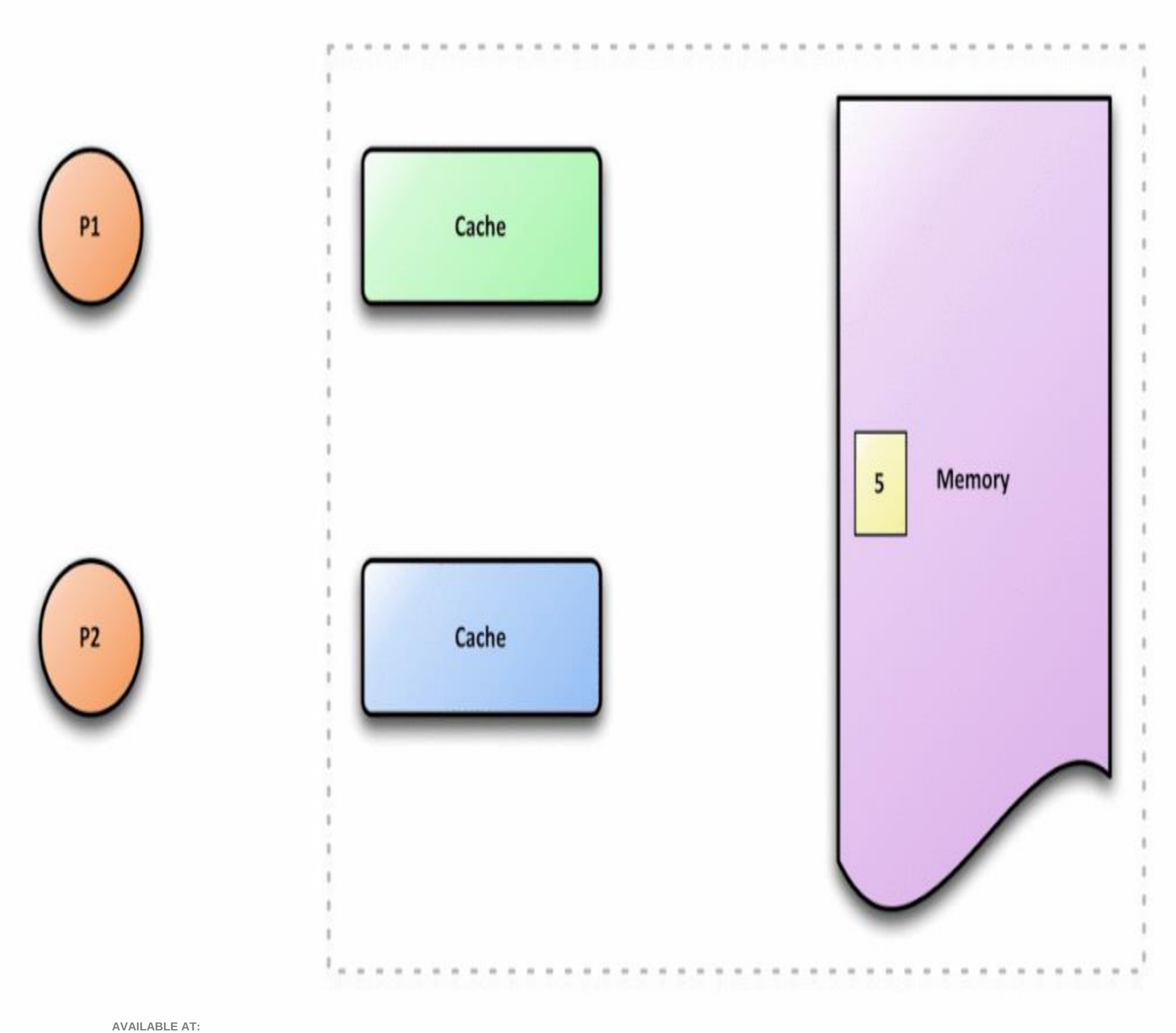
Coda Distributed File System

- Files grouped into volumes replicated on Coda servers
- Version stamping to resolve conflict
- Client caching reduces network dependence and provides scalability
- Uses optimistic approach
- Callbacks to maintain cache coherency

Incoherent caches



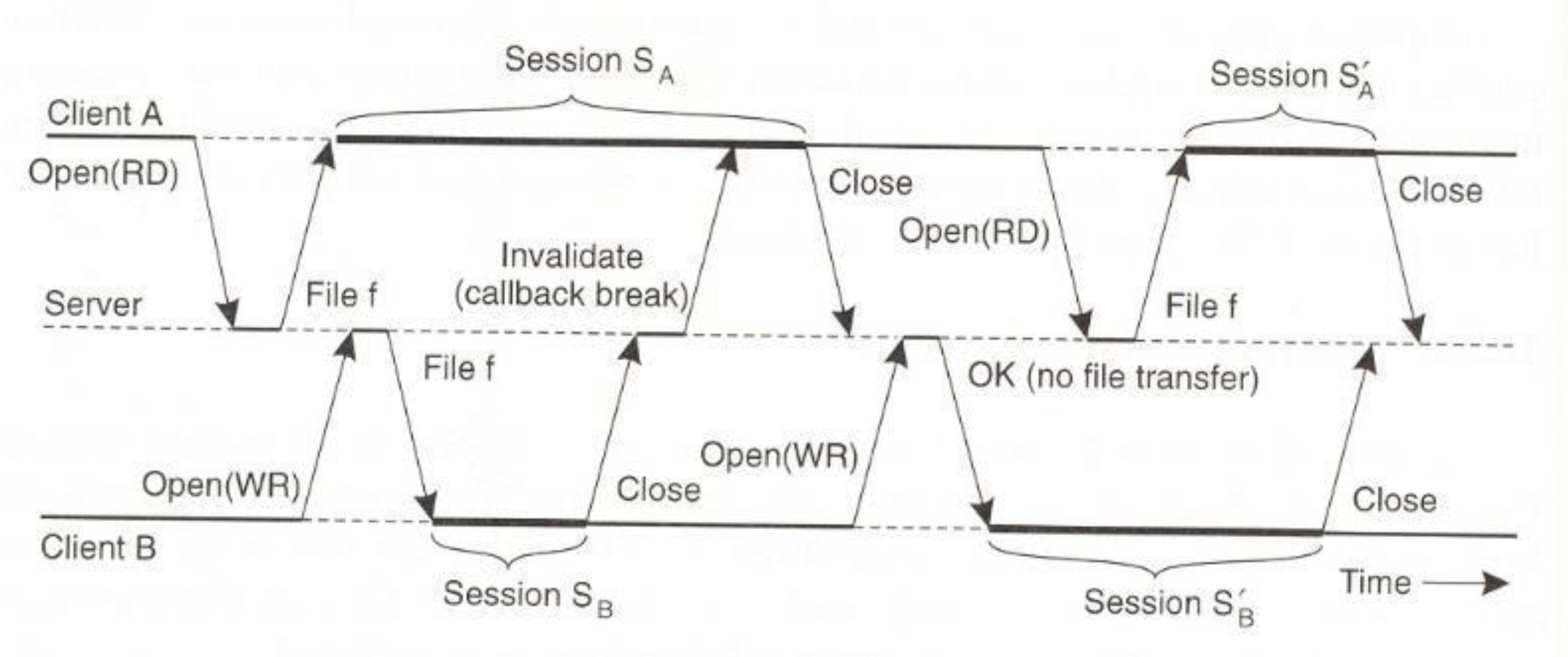
Coherent caches



Coda features

- Disconnected operation for mobile computing
- Is freely available under the GPL
- High performance through client side persistent caching
- Server replication
- Security model for authentication, encryption and access control
- Continued operation during partial network failures in server network
- Network bandwidth adaptation
- Good scalability
- Well defined semantics of sharing, even in the presence of network failure

Coda Client File Session



Sharing Files in Coda

- **When a file is opened, the entire file is transferred to the client**
  - If a client has opened file *f* for writing, another client cannot open *f*
  - If a client has opened file *f* for reading, another client can open *f* for reading or writing

- **When a file is closed, the file is transferred back to the server if it has been modified**

- **A session is treated as a transaction**
  - In the figure, session S_A is considered to have been scheduled before session S_B. Thus, client A can proceed to read its local copy despite the fact that the copy is outdated

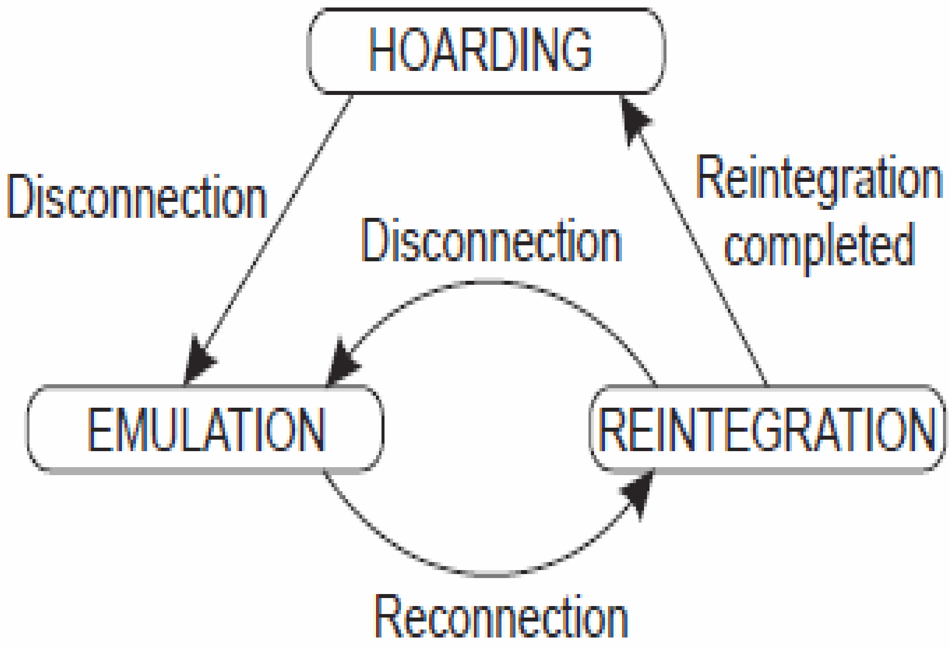
Client-Side Caching in Coda

- When a file is opened, an entire copy of the file is transferred to the client and is cached at the client
- Cache consistency is maintained by means of callbacks
  - The server keeps a callback promise for each client that has a local copy of a file
  - When a client updates its local copy of the file for the first time, it notifies the server. The server then sends an invalidation message to the other clients
    - The invalidation message is called a **callback break** because the server will then discard the **call back promise** it held for the client that received the **invalidation**

Client-Side Caching in Coda

- When a client opens a file that is still in its cache, it must check with the server if a callback promise on the file still holds. If so, the client can access the local copy of the file.

Coda Client States



Disconnected Operation in Coda

- When the client is connected to the network, it is in the HOARDING state, where it prefetches all files that it wants to access and caches them locally
- If the client becomes disconnected, it enters the EMULATION state, where all file requests are serviced locally
- When the client reconnects, it enters the REINTEGRATION state, where all updates are sent to the server
  - Conflicts are detected and resolved at the server

Conclusions

- Data consistency, data caching, scalability, location transparency, data replication, and availability
- **Adaptability to network**

References

- Satyanarayanan, M. Fundamental Challenges in Mobile Computing. Fifteenth ACM Symposium on Princ. of Distr. Comp. May 1996.
- Honeyman, P., Huston, L. B. Communications and Consistency in Mobile File Systems. *IEEE Personal Communications*, Dec. 1995, pp. 44-48.
- Braam, P. J. The Coda Distributed File System. Carnegie Mellon University. http://www.coda.cs.cmu.edu/ljpaper/lj.html.
- Schiller, J. Mobile Communications (2nd ed.). Addison-Wesley, 2003, pp. 375-381.
- Tanenbaum, A. S., van Steen, M. Distributed Systems: Principles and Paradigms. Prentice Hall, 2002, pp. 604-623.
- https://en.wikipedia.org/wiki/Coda_(file_system)