CS514: Intermediate Course in Computer Systems

Lecture 27: Nov. 26, 2003

"Challenges of Mobility"

Mobility is a huge topic

CS514

- Breaks existing applications
  - Anything bandwidth intensive or synchronous
- Opportunities for new applications
  - Location-specific (nearest Starbucks)
  - Ubiquity (short messaging)
  - Ad hoc networks
- Can't possibly give it justice in one lecture

Focus on a couple systems-level attempts

CS514

- What can the system do to support applications in mobile contexts, and how how effective is it?
- Coda (mobile file system)
  - CMU (AFS-based)
  - Application awareness
- Rover (mobility systems toolkit)
  - MIT
  - Application transparent

Characteristics of mobility

CS514

Disconnection (long or short, predictable or sudden)	Caching, hoarding, prefetching, DB/file inconsistencies
Variable and asymmetric bandwidth	Above, plus compression, prioritization, clever use of downlink
Expensive ($$$) BW	Above, plus user control
Battery, battery, and battery	Minimize transmissions (and also processing)
Weakened security (physical and radio)	User auth, encryption

Rover goals

CS514

- Philosophy is that applications know best how to deal with mobility
  - But there are general mechanisms that all applications can benefit from
- Provide a toolkit to applications
- Make it easier to write applications that deal with mobility issues
  - Reduce client/server communications requirements
  - Allow the user to effectively work offline

(Some slides care of Michael Ferguson)

Rover project

CS514

- Build Rover toolkit
- Build range of applications using Rover toolkit
  - Email
  - Calendar
  - Browser
- Evaluate effectiveness of Rover for supporting these applications

Conclusion: Success!

CS514

- Though Rover itself has not taken off…
- Applications easy to port
  - They say…
- Performs well
  - Compared to what?
- In my mind, they never really answer the question whether a toolkit/OS approach is better
  - This would be a hard question to answer…

Two basic mechanisms

CS514

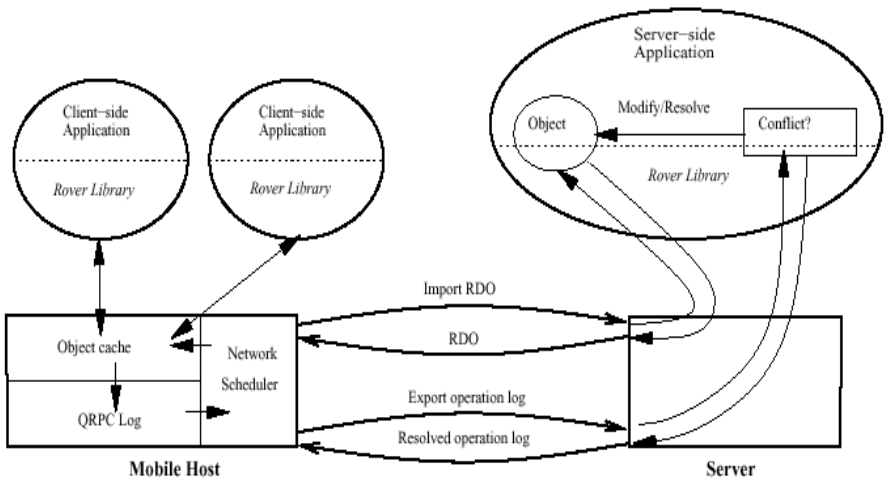
- Relocatable dynamic objects (RDO)
  - Code/data shipping, like simple agents or process migration
  - Allows dynamic control over processing versus communications tradeoff
- Queued remote procedure calls (QRPC)
  - Asynchronous RPCs
  - Allows offline operations without blocking

Rover operations

CS514

- *Import* objects onto client machine
  - RDO: contains data and operations on the data
- *Invoke* methods on those objects
- *Export* logs of method invocations to servers
  - Can also export RDOs
- Reconcile client data with server data

Rover operation

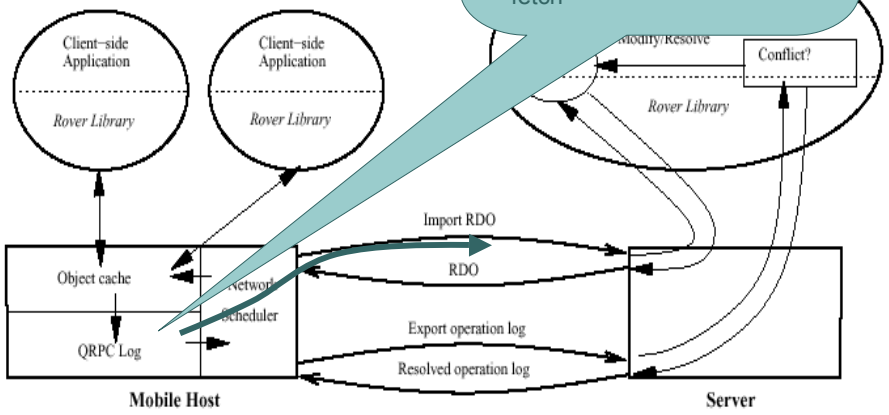


Import

Import call provides:
- Object ID (URN)
- Session ID
- callback routing
- arguments

Import call returns a “promise”

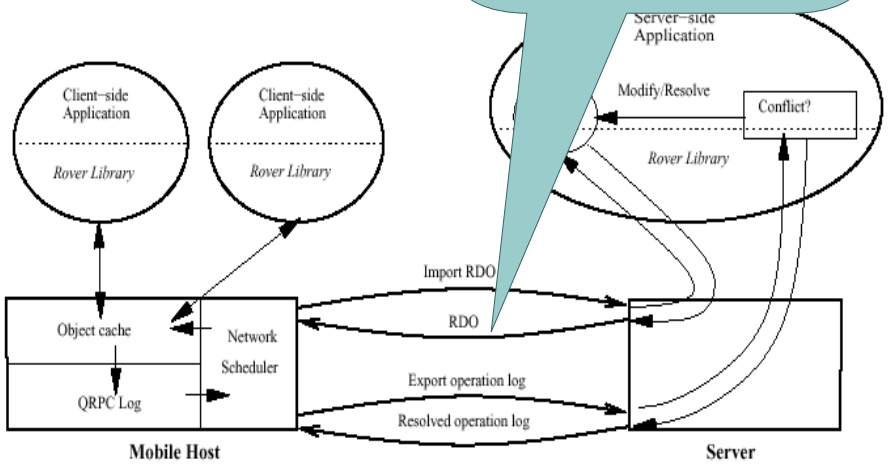
Call is queued (QRPC) for lazy fetch



Import

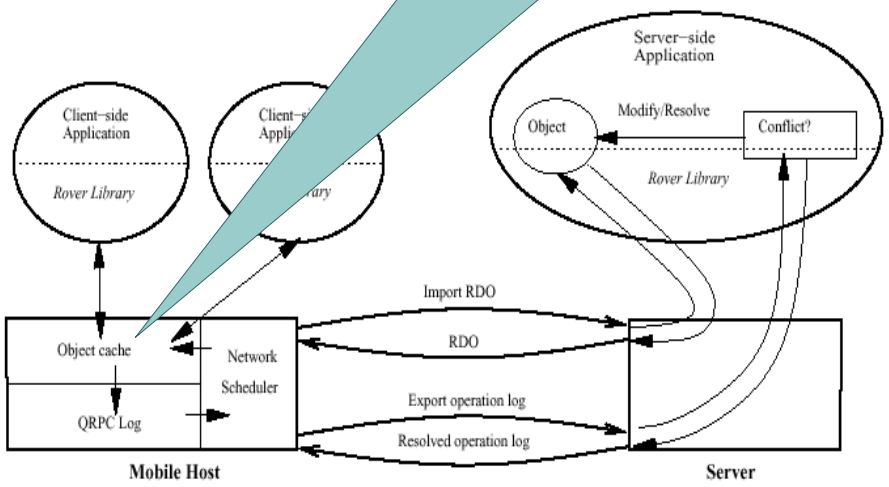
When imported object arrives:
- Callback routine is called
- Object is put into the cache
- RDO may invoke a thread

Object may be locked at server



Invoke

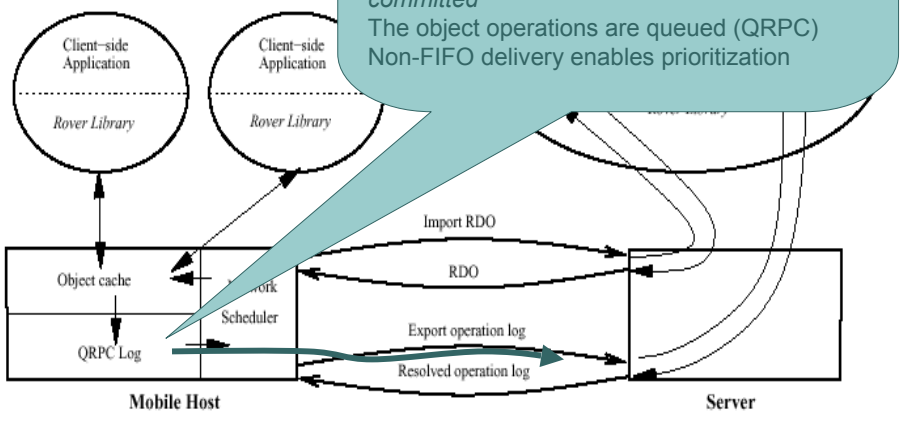
Application invokes methods on received object
- Each operation is stored
- Changes in object indicated by version vectors



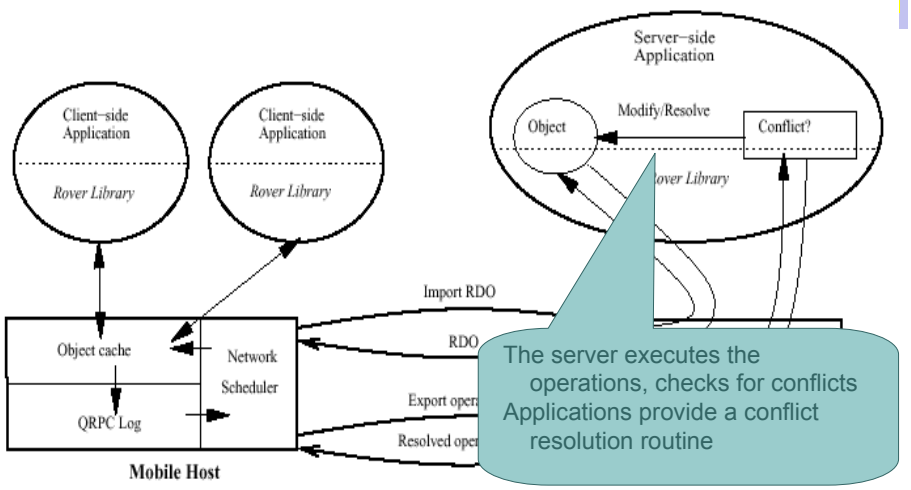
Export

Export call provides:
- Object ID (URN)
- Session ID
- callback routing
- arguments

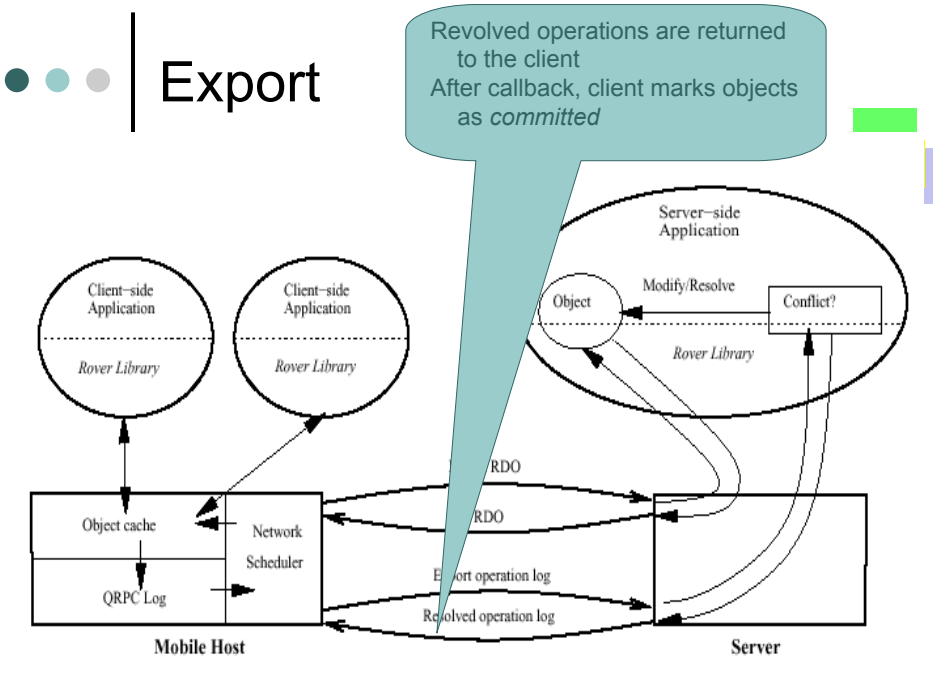
Export call returns a “promise”
Updates to object are labeled *tentatively committed*
The object operations are queued (QRPC)
Non-FIFO delivery enables prioritization



Export

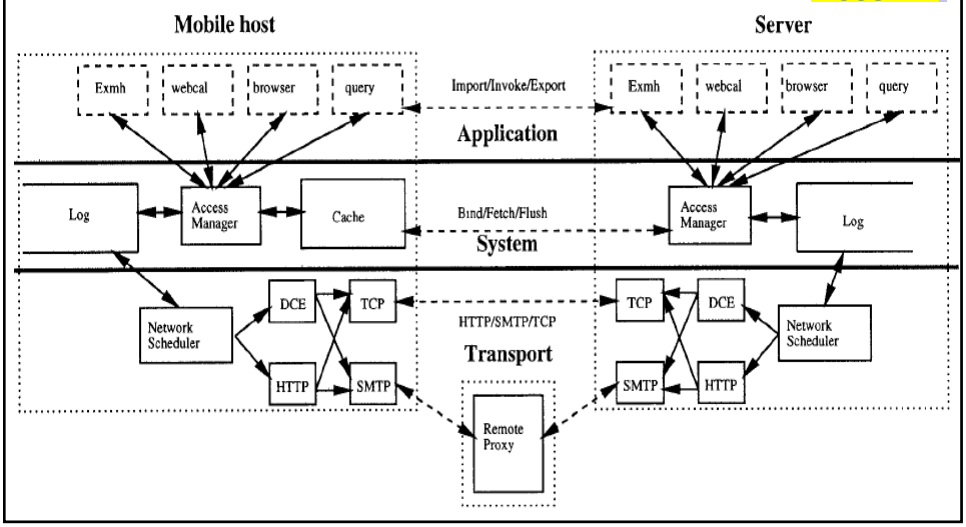


Export



Rover Architecture

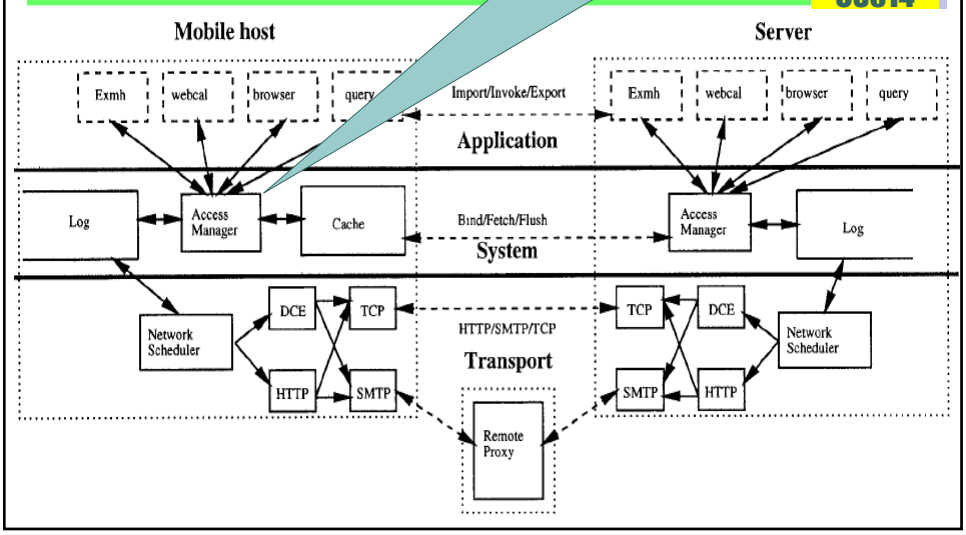
CS514



Rover Arc

All applications run over a single Rover process
Communicate via Local RPC
Allows prioritization across applications

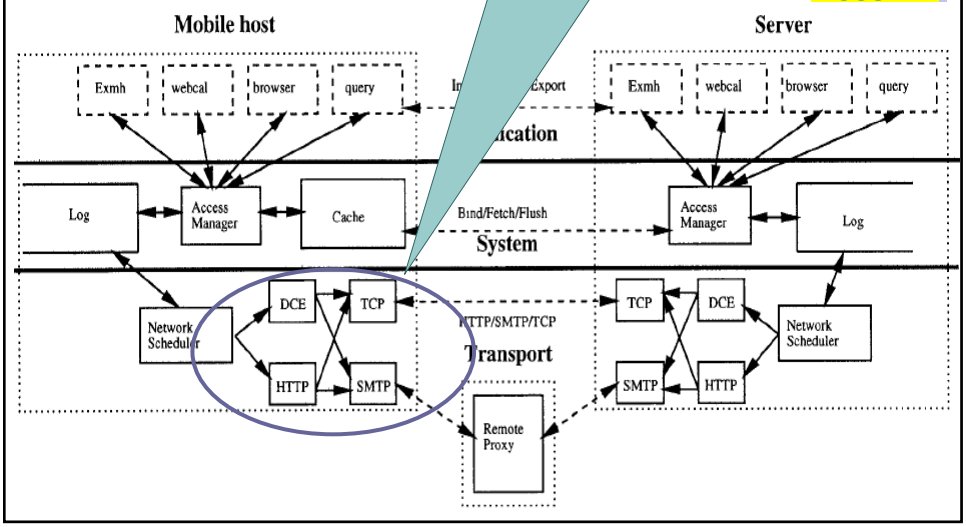
CS514



Rover Arc

Network scheduler can select TCP or SMTP (mail) for transport
Latter for batching lower priority requests
(Interim solution)

CS514



Implementation

CS514

- RDOs are Tcl/tk objects
- Transported in HTTP
  - Using CERN's Web Common Library
  - Server uses CGI scripts

Applications

CS514

- Mail reader (based on Exmh Tcl/Tk)
- Calendar (based on Ical Tcl/Tk calendar)
- Web browser proxy (new application)

Mail Reader

CS514

- Parts of GUI and messages sent as RDOs
- RDOs used for prefetching and application-specific consistency (inconsistent folder changes)

Calendar

CS514

- Each calendar item (appointment or notice) is an RDO
- Item imported, changed tentatively, and exported and committed
- Routines for conflict resolution
  - Error notice, or give some users priority

Web browser proxy

CS514

- Implements “click ahead”
  - During disconnection, clicks are queued for later download
  - User has access to list of queued clicks
    - List is an RDO
- Does prefetching

Some thoughts on Rover

CS514

- Rover is a nice proof-of-concept for how to deal with mobility
- But Rover itself of limited value
  - Tcl/Tk based RDOs probably overtaken by Java
  - Use of SMTP a bad choice (they know this)
  - Probably hard to automatically prioritize among disparate applications
    - User would prefer to control this based on immediate circumstances
    - Not clear there is much value to running Rover as a single, system service

Some thoughts on Rover

CS514

- Email not the best proof-of-concept application
  - Already fundamentally asynchronous, so not much different with Rover
- Click-ahead sounds like a bad idea to me
  - I'd rather control when clicks happen…
- Calendar is a decent proof-of-concept application

Surprising conclusion

CS514

- From the Rover paper:
  - "*The largest, most important, drawback of the Rover approach is that application designers must think carefully about how application functions should be divided between a client and a server*"
- Funny…this struck me as probably the main advantage of Rover!!!
  - Provides a nice model for how to think about disconnection, asynchrony, and consistency

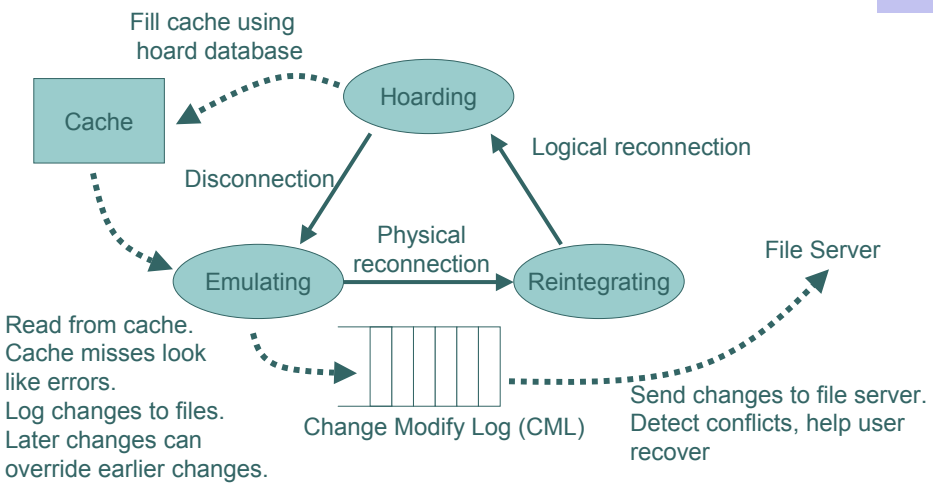
Coda File System

CS514

- Unlike Rover, makes disconnection issues transparent to the application (and, to some extent, the user)
  - Coda transparently propagates file modifications and handles conflicts

Disconnected operation states on client (Venus)

CS514



Conflict resolution

CS514

- Code resolves most directory conflicts
- For files, requires application-specific resolvers
- Unresolvable files are presented to user in a manual repair tool
  - User sees an “explosion” of inconsistent files in a directory tree
  - Use diff and grep to resolve

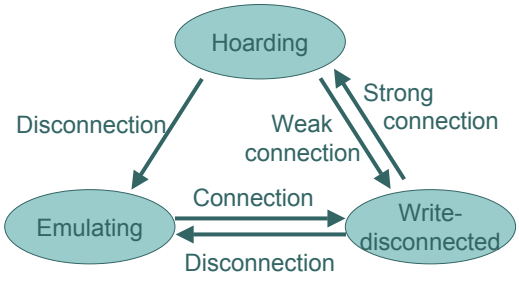
Problem with disconnected states approach

CS514

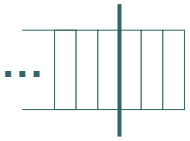
- Reintegration would consume bandwidth resources…users couldn't do anything useful immediately upon reconnect
- Solutions:
  - New states for weak connectivity
  - Rapid cache validation (version stamps for directories, not just files)
    - Cached by clients
  - "User patience threshold"…model to predict if a user would rather wait for a large file not in the cache, or be given an error

Weak connectivity operation states on Venus

CS514



Changes older than some time (10 min) cannot be over-ridden. These are "trickled" out to the file server during weak connectivity.



Reintegration Barrier

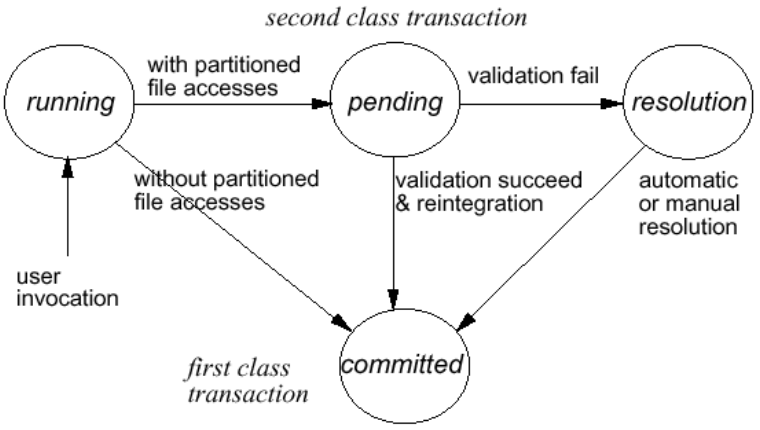
Isolation-Only Transactions

CS514

- Coda emulation of UNIX file system has benefit of backwards compatibility
- UNIX lacks notion of read-write file conflicts
  - Where an application is using a file as input, and that file is modified
    - Windows, on the other hand, locks the file
- This limitation is exacerbated by disconnected operation
- Coda deals with this by checking for possible read-write inconsistencies after reconnection

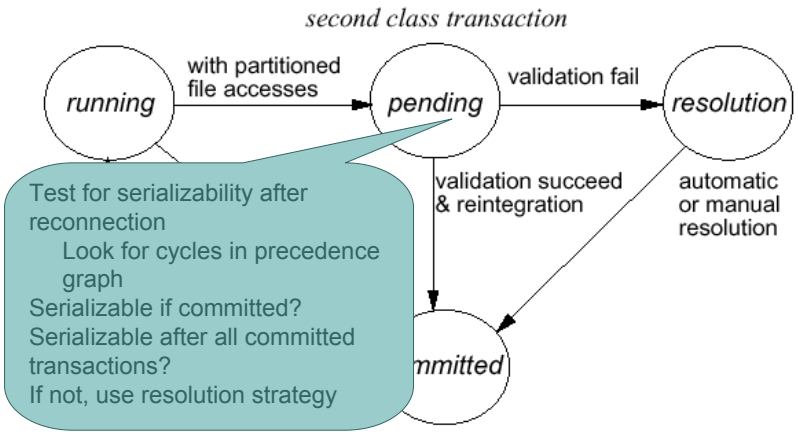
State machine for IOT

CS514



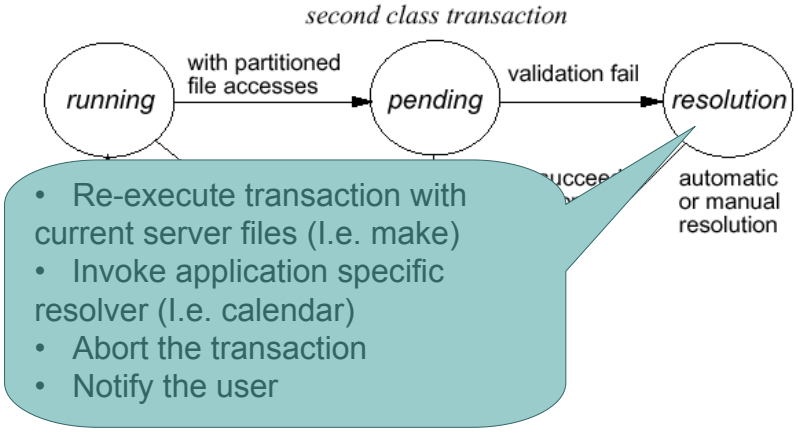
Pending validation

CS514



Resolution Strategies

CS514



Some thoughts on Coda

CS514

- File system is the wrong level of abstraction for many applications
  - Calendar, database
  - I agree with Rover on this
- As a user, I think Coda running "under the hood" would be confusing, sometimes annoying
  - If file is shared, I'd rather deal with resolution explicitly (version control, etc.)
  - If file is not shared, I'd rather control when "synchronization" takes place

Other interesting work

CS514

- Bayou (Xerox Parc)
- "Peer-to-peer" ad hoc network write conflict resolution
  - Group document editing, calendar, etc.
- Basic idea, "anti-entropy": peers do pairwise comparison of writes, try to revolve conflicts
  - Determine conflict by trying the write on neighbors version, conflict exists if result is different
  - Eventually all peers reach an agreed state

Other interesting work

CS514

- Broadcast disks
- Based on fact that radio reception much less expensive than radio transmission
  - Acharya et. al., SIGMOD95
- Continuous broadcast of “disks”, clients keep the ones they want
  - Broadcast index at set times so that clients know when to receive
- Variations to support caching, consistency
  - Broadcast version changes