Code File System Review (by Saurabh Talbar)

Just at the name of the paper suggests, 'disconnected' operation is the key motivation for this research project @ CMU. Coda is a refined version of Andrew file system and inherits most of the characteristics from it. Coda was designed with the aim of providing similar service as Andrew to the 'mobile' users. When a user is continuously connecting/disconnecting from the server, authors were motivated to design some service for such users even if they are disconnected. Providing good availability of resources is the fundamental contribution by Coda and it does this by caching of data at local (client) side. Most important contribution of this research is to thrust the development in mobile computing by exploiting user side characteristics. Coda has implemented its 'venus' client on various platforms and even today they maintain an active community.

Coda has some distinct features keeping in mind the need of the mobile clients. Though, the paper suggests Coda does not perform good for concurrent applications.

Every Coda client has a Venus (Cache Manager) service running & it makes sure that required data is made available to application / user. Coda provides high availability of data by implementing below two techniques:

Replication of servers: this mechanism allows multiple read-write operations simultaneously. Venus uses has a access to Available Volume Storage Group (AVSG) which contains an active list of servers.

Disconnected Operation: It as an autonomous service taken care by Venus. When an AVSG is empty, response to client requests are solely done on the local cache.

From the perspective of a programmer, we are interested in the design of how venus is deployed in the client and authors have implemented venus on user level. Venus can be present in one of three states at any moment namely:

Hoarding: This state is where venus is connected to the server and its manages the cache such that there is balance between the connected and disconnected modes. It uses prioritized caching to ensure this.

Emulation: venus enters this state when it disconnects from the server and plays the role of pseudo-server.

Reintegration: When client reconnects to the server, to maintain the persistence of data, venus synchronizes the server side state in its cache and vice versa.

Evaluation:

The paper provides few benchmarks of the performance of Coda, like it

take 52 seconds to totally reintegrate with server compared to 43 sec required by Andrew.
Though, these results are not convincing as to why Coda should perform better than other distributed file systems. Plus, Coda has few weaknesses such as:

Coda does not consider all types of conflicts occur from user end

There is not much data to evaluate how Coda will work under critical circumstances e.g when network connectivity is very bad. Neither is their any discussion about the security model. While the authors suggest that Coda can be scaled to large number of clients, there are no convincing results for the same.

Class Discussion:
Q. How would coda support for later portable devices?
-> It is feasible since originally coda was developed on UNIX platforms. Though, the performance of coda should be considered and also the memory it demands from local disks.

Q. What about wide area support?
-> synchronization and consistency are major issues to deal with

Q. How do you compare coda with AFS?
-> w.r.t. reliability, coda is more reliable due to server side replication ;
performance wise, it should be equal or less than AFS &
scalability wise, it can be compared with AFS

Q. Venus @ user level, why?
-> Portability is the major issue we are trying to solve in coda, so user level implementation is better.