

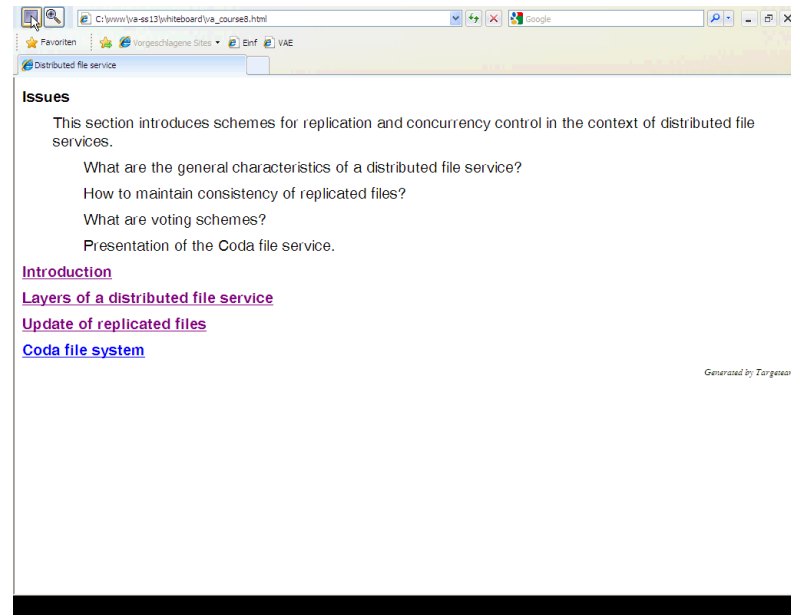
Script generated by TTT

Title: Distributed_Applications (15.07.2013)

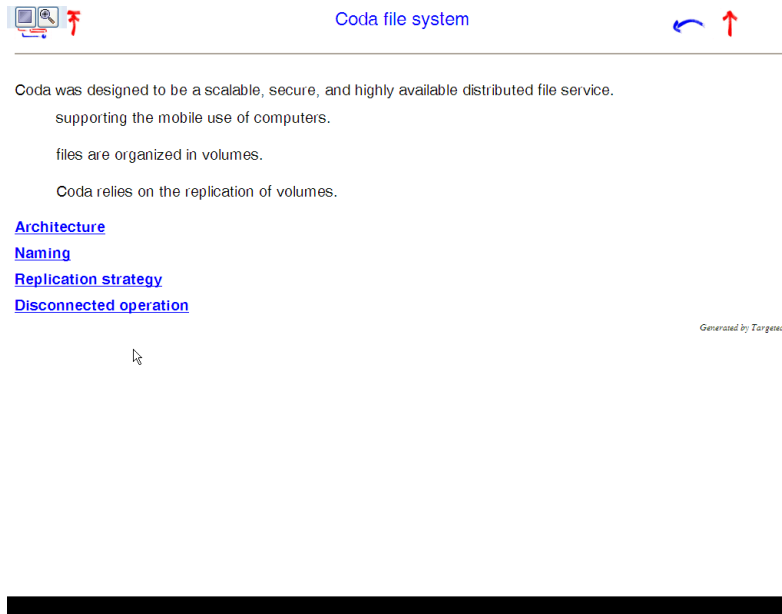
Date: Mon Jul 15 09:11:34 CEST 2013

Duration: 45:07 min

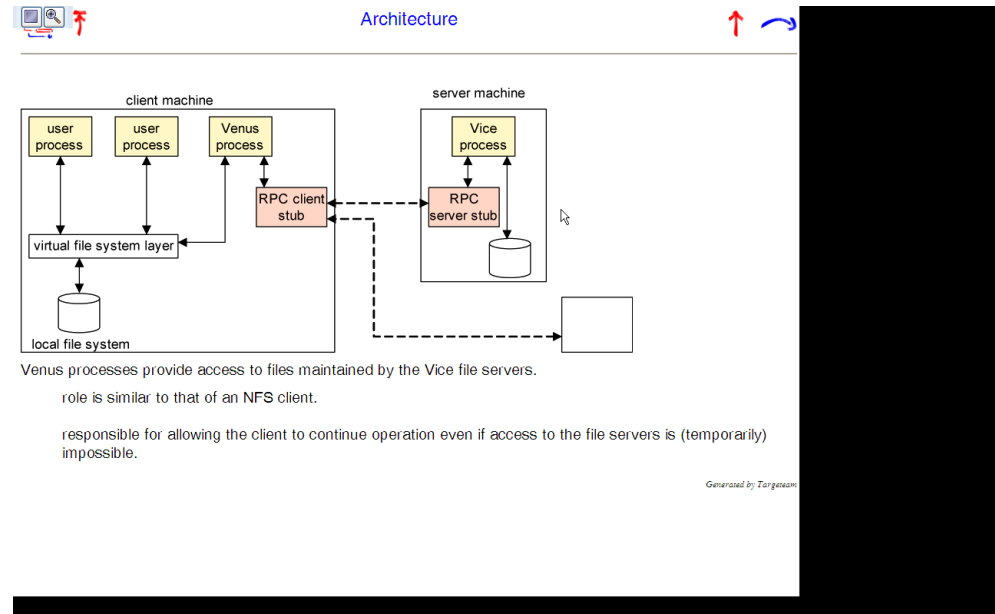
Pages: 9



The screenshot shows a web browser window with the address bar containing 'C:\www\ya-st-13\whiteboard\ya_course8.html'. The page title is 'Distributed file service'. The content includes a section titled 'Issues' with the following text: 'This section introduces schemes for replication and concurrency control in the context of distributed file services.' followed by three bullet points: 'What are the general characteristics of a distributed file service?', 'How to maintain consistency of replicated files?', and 'What are voting schemes?'. Below these is the text 'Presentation of the Coda file service.' and three links: 'Introduction', 'Layers of a distributed file service', and 'Update of replicated files'. A link for 'Coda file system' is also present. A small 'Generated by Targem' watermark is visible in the bottom right corner.



The screenshot shows a web browser window with the address bar containing 'C:\www\ya-st-13\whiteboard\ya_course8.html'. The page title is 'Coda file system'. The content includes the following text: 'Coda was designed to be a scalable, secure, and highly available distributed file service. supporting the mobile use of computers. files are organized in volumes. Coda relies on the replication of volumes.' Below this are four links: 'Architecture', 'Naming', 'Replication strategy', and 'Disconnected operation'. A small 'Generated by Targem' watermark is visible in the bottom right corner.



The screenshot shows a web browser window with the address bar containing 'C:\www\ya-st-13\whiteboard\ya_course8.html'. The page title is 'Architecture'. The content includes a diagram illustrating the architecture of the Coda file system. The diagram shows a 'client machine' and a 'server machine'. The client machine contains a 'local file system' (cylinder icon) connected to a 'virtual file system layer' (rectangle). Above this layer are three boxes: 'user process', 'user process', and 'Venus process'. The 'Venus process' is connected to an 'RPC client stub' (orange box). The server machine contains a 'Vice process' (yellow box) connected to an 'RPC server stub' (orange box). The 'RPC server stub' is connected to a 'Vice process' and a 'local file system' (cylinder icon). Dashed arrows indicate communication between the 'RPC client stub' and the 'RPC server stub'. A small 'Generated by Targem' watermark is visible in the bottom right corner.



Coda file system



Coda was designed to be a scalable, secure, and highly available distributed file service, supporting the mobile use of computers.

files are organized in volumes.

Coda relies on the replication of volumes.

[Architecture](#)

[Naming](#)

[Replication strategy](#)

[Disconnected operation](#)

Generated by Targem



Naming



Each file is contained in exactly one volume. Distinction between physical volumes.

logical volume (represents all replicas of a volume).

RVID (Replicated Volume Identifier): identifier of a logical volume.

VID (Volume Identifier): identifier of a physical volume.

[File identifier](#)

Generated by Targem



Replication strategy



Coda relies on replication to achieve high availability. It distinguishes between two types of replication.

[Client caching](#)

[Server replication](#)

Generated by Targem



Server replication



Coda allows file server to be replicated; the unit of replication is a volume.

Volume Storage Group (VSG): collection of servers that have a copy of a volume.

client's Accessible Volume Storage Group (AVSG): list of those servers in the volume's VSG that the client can contact.

AVSG = {}; client is disconnected.

Coda uses a variant of the "read-one, write-all" update protocol.

[Coda version vector](#)

Generated by Targem



Message passing model

variables have to be marshalled from one process, transmitted and unmarshalled into other variables at the receiving process.

Distributed shared memory

the involved processes access the shared variables directly; no marshalling necessary.
processes may communicate via DSM even if they have non-overlapping lifetimes.

Implementation approaches

in hardware

shared memory multiprocessor architectures, e.g. NUMA architecture.

in middleware

language support such as Linda tuple space or JavaSpaces.

Generated by Targem



Issues of the section

implicit communication via shared memory

what is the Linda tuple space?

JavaSpaces as modern tuple space

[Introduction](#)

[Programming model](#)

[Consistency model](#)

[Tuple space](#)

[Object Space](#)

Generated by Targem