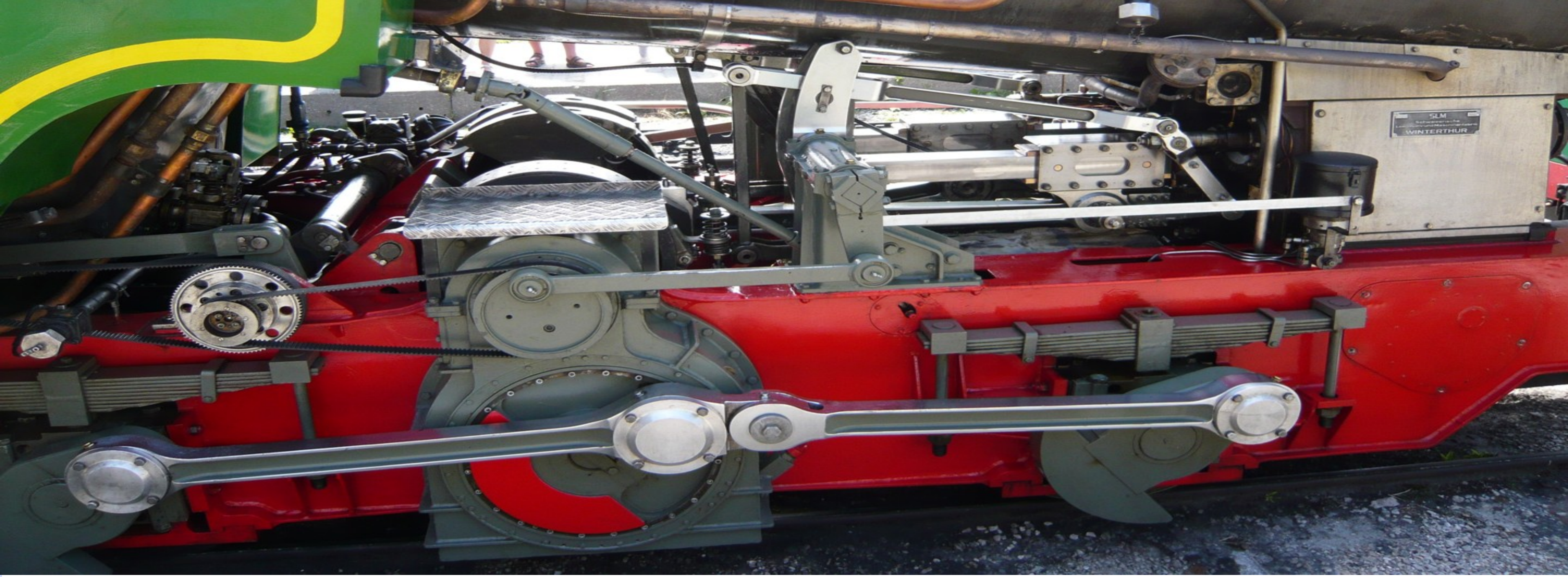


SMC 2.0

Fabrizio Manfredi Furuholmen
Giuseppe Guarino

Beolink.org

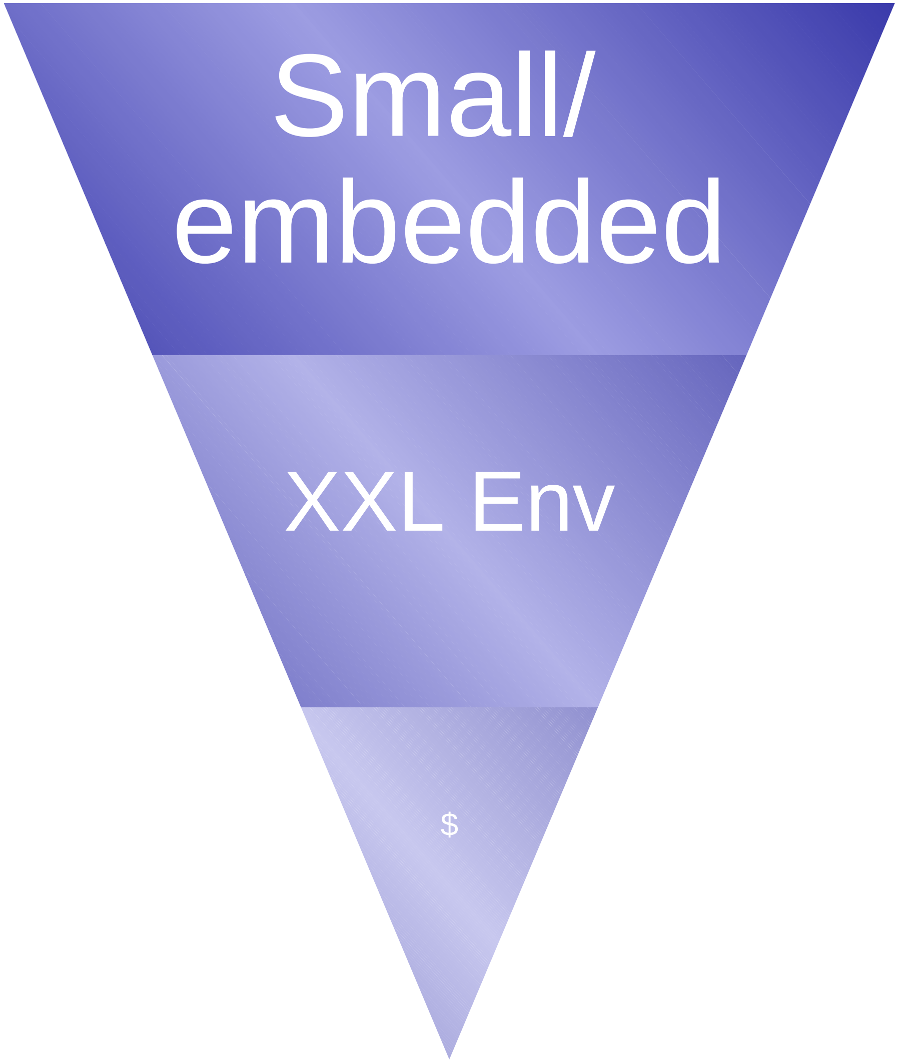
Agenda

- Introduction

- SMC
  - Goals
  - Architecture
  - Internals
  - Configuration and Deploy

- CloudVFS SubProject
  - Overview
  - internals

Introduction: Samba usage



Small/ embedded

XXL Env

$

Domestic Storage NAS
All in one Appliance

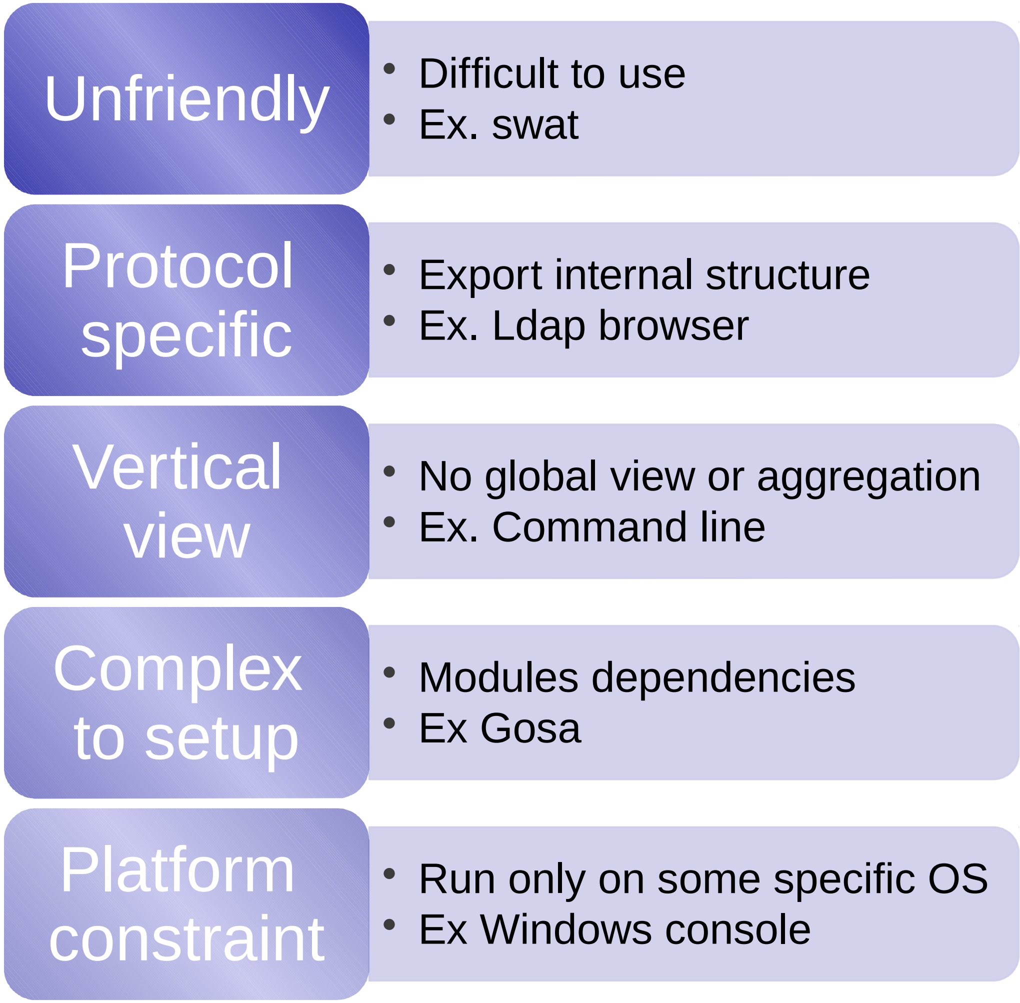
No i386 hw
Heterogeneous env
High performance

Fanatic
No money..

Few small business
Few installation for office automation

Introduction: Management

"A major advantage of GUIs is that they make computer operation more intuitive, and thus easier to learn and use.."



Unfriendly

- Difficult to use
- Ex. swat

Protocol specific

- Export internal structure
- Ex. Ldap browser

Vertical view

- No global view or aggregation
- Ex. Command line

Complex to setup

- Modules dependencies
- Ex Gosa

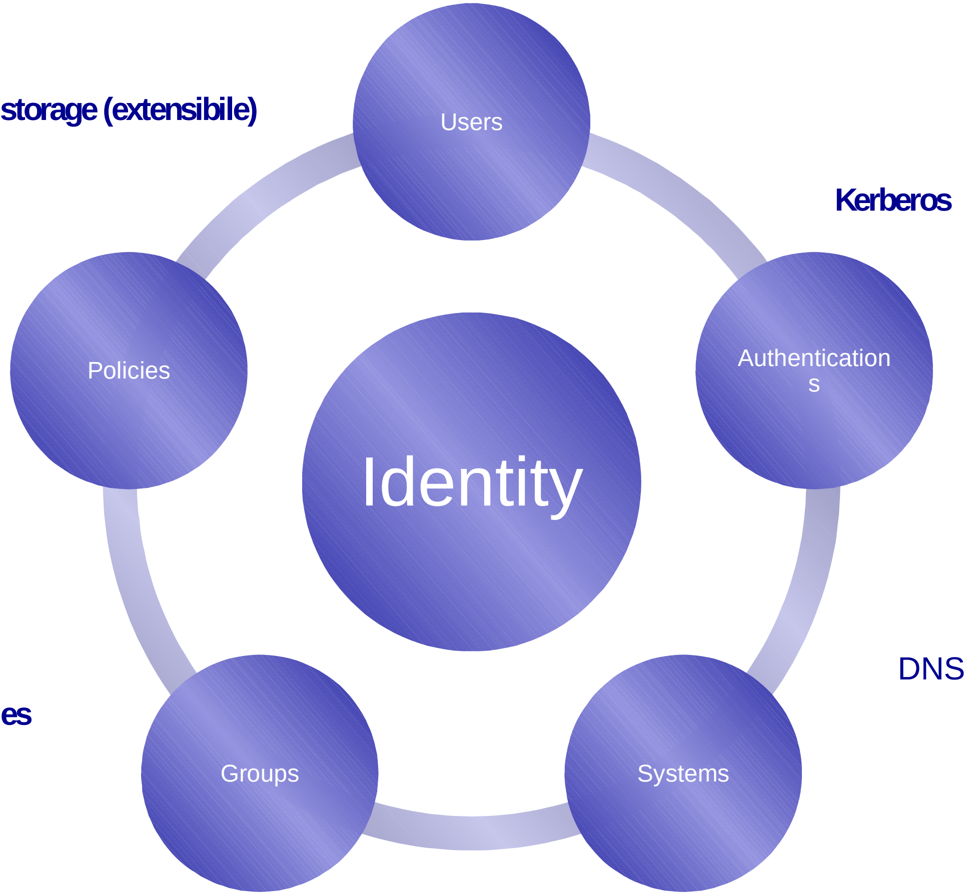
Platform constraint

- Run only on some specific OS
- Ex Windows console

Introduction: Samba4 AD

LDAP storage (extensibile)

Kerberos



DNS (update)

Group Policies

Powerful Identity Management (for Free)

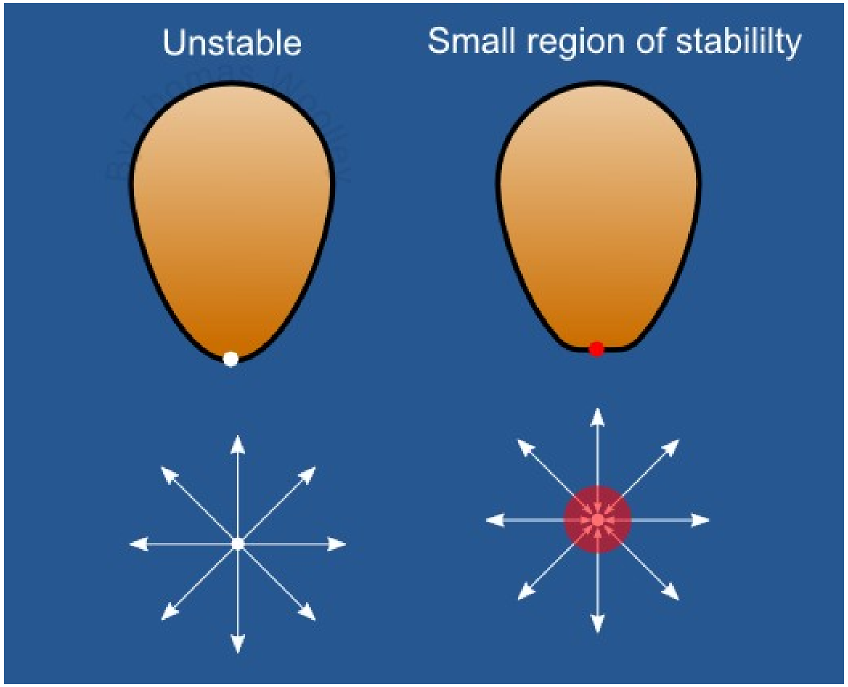
Introduction: Idea

Samba Management Console 1.0

Introduction: Idea

The result…

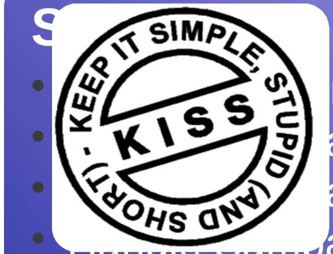
Introduction: Idea



Try again…

Introduction: Project Goals

S



- r interface
- a configuration
- a Status (Process, session management
- Handle Samba Users/Groups
- Setup

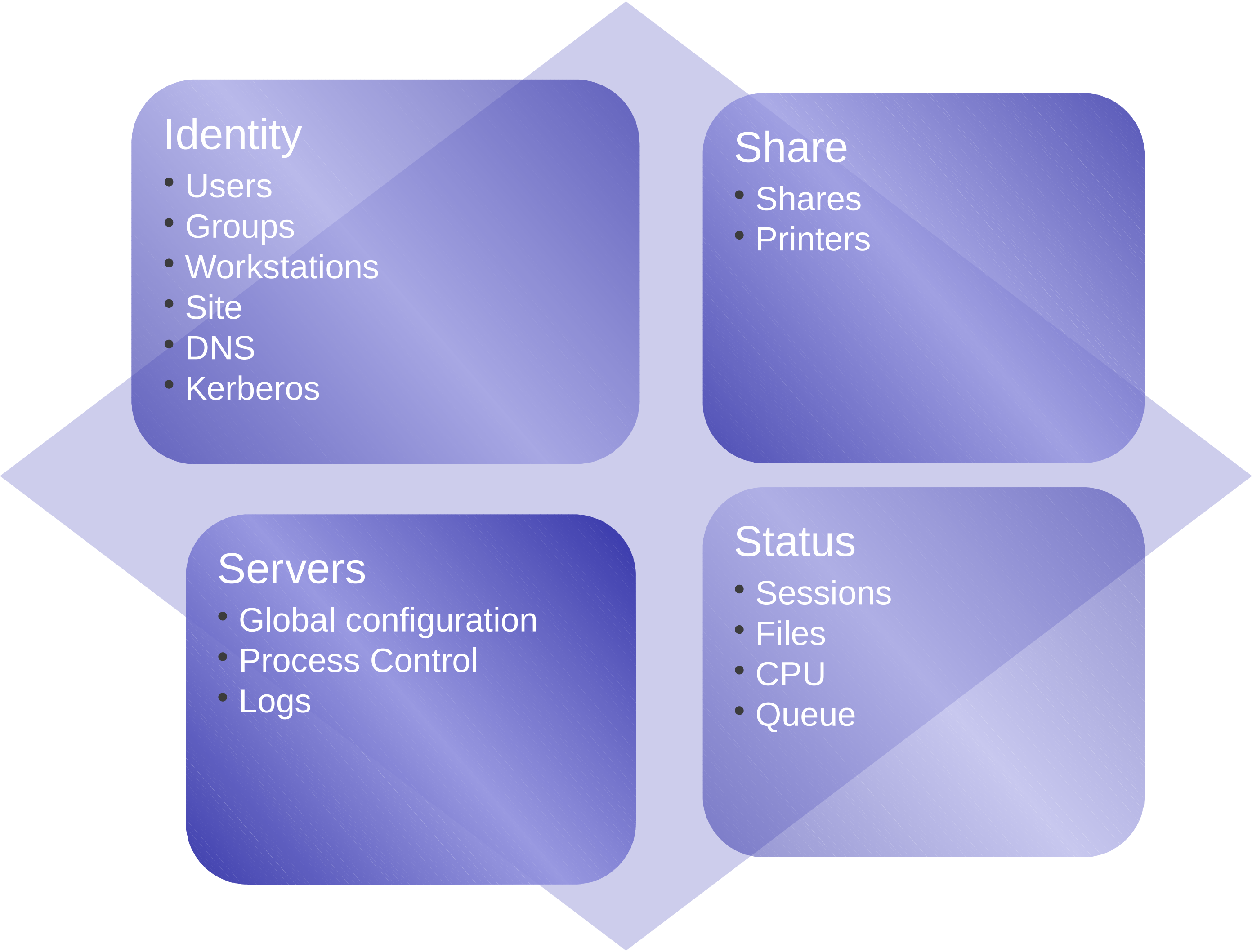
C

- ration/Automation with other systems
- nba PDC/AD
- dows AD

C

- vers from a central point
- s of servers as one
- ation from different servers

Architecture : functions



Identity

- Users
- Groups
- Workstations
- Site
- DNS
- Kerberos

Share

- Shares
- Printers

Servers

- Global configuration
- Process Control
- Logs

Status

- Sessions
- Files
- CPU
- Queue

Architecture: Design 1.0

Beolink.org

Rich Client

Node Collector

Scheduler

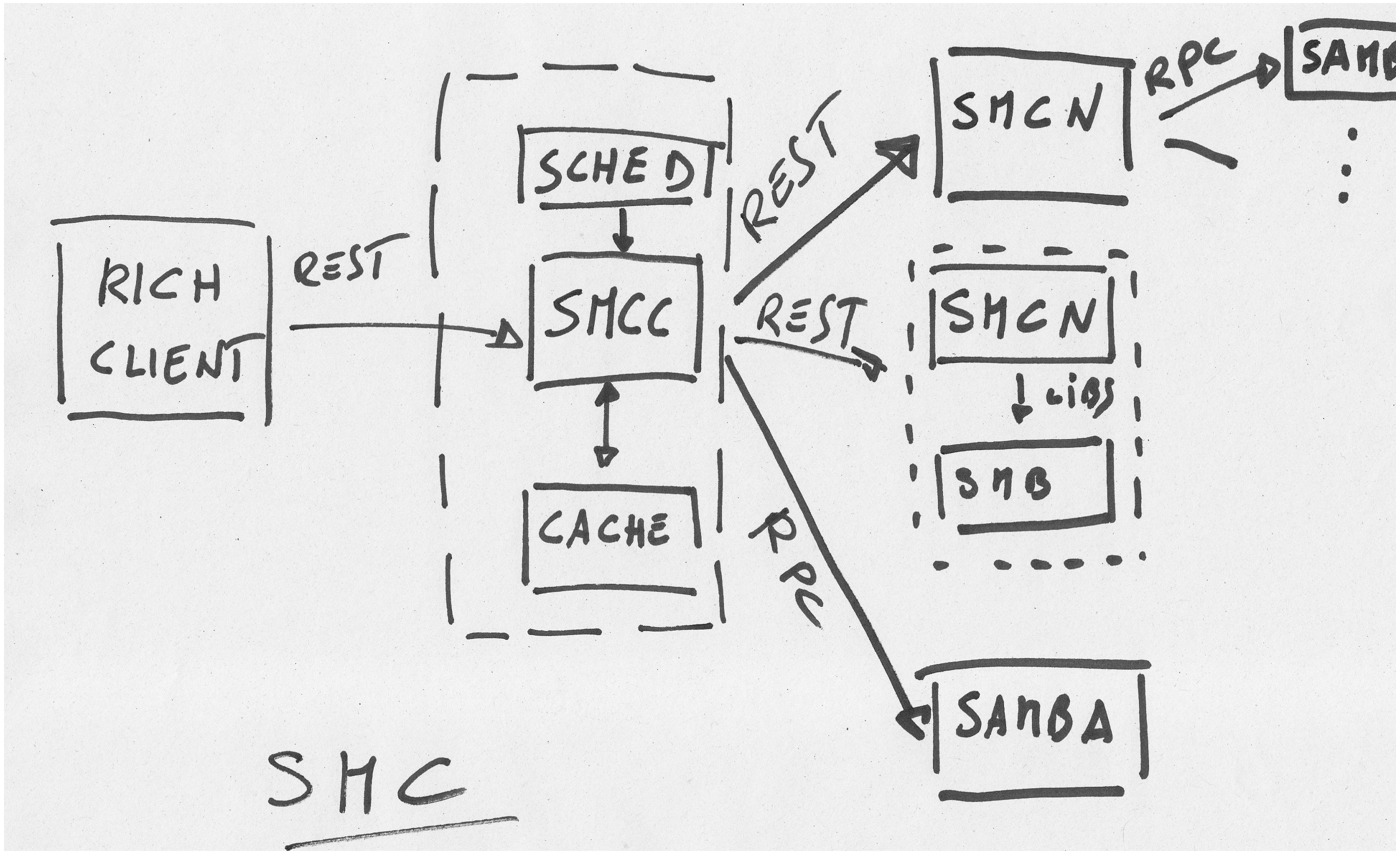
Web Int

Cache

Node Controller

Proxy

Local



Architecture: Design 2.0

Rich Client

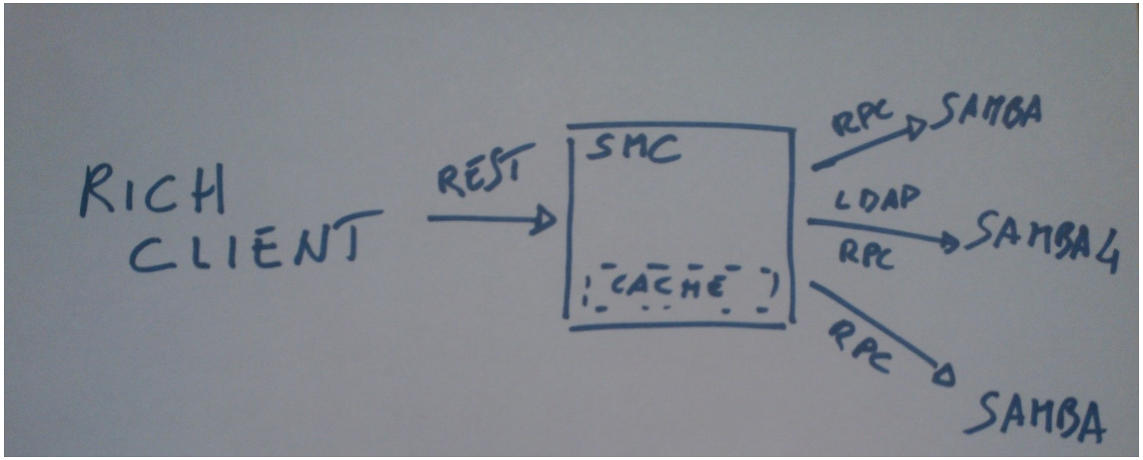
SMC Controller

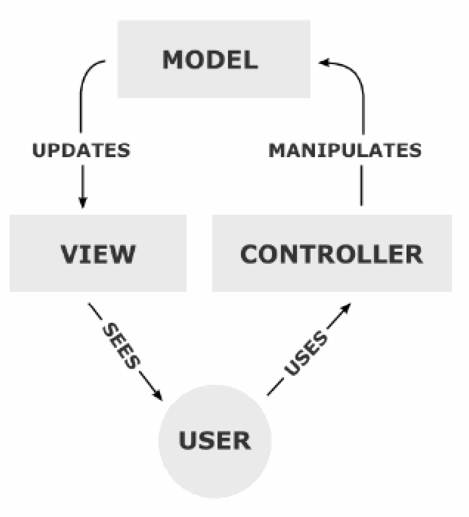
Scheduler

Web Int

Cache

AD/RPC base



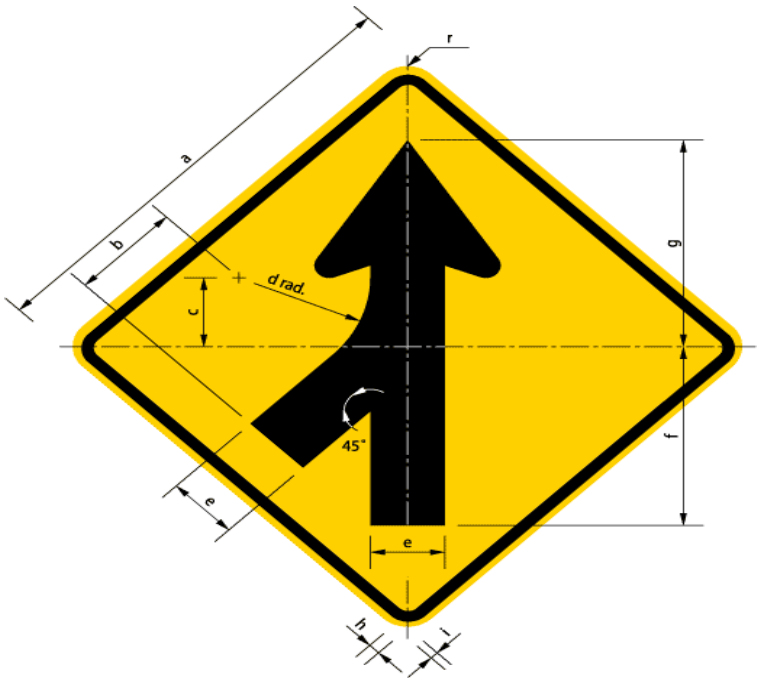


Architecture: Rest Interface

Beolink.org

URL /rest/type/resource/id

- **Type**
  Servers, Identity, Status, Share
- **Resources**
  User, groups…
- **Id (optional)**
  Resource Identification
- **Operations**
  - Get: list elements/attributes in resource
  - Post: create new resource
  - Put: update resource
  - Delete: remove resource



Architecture: Operations

Area	Resource	Protocol
Identity	Users Groups Sites DNS DHCP	Ldap
Server	Process Global config	RPC
Status	Session Stats on share Logs	RPC/External
Shares	Share Printing	RPC(Registry)

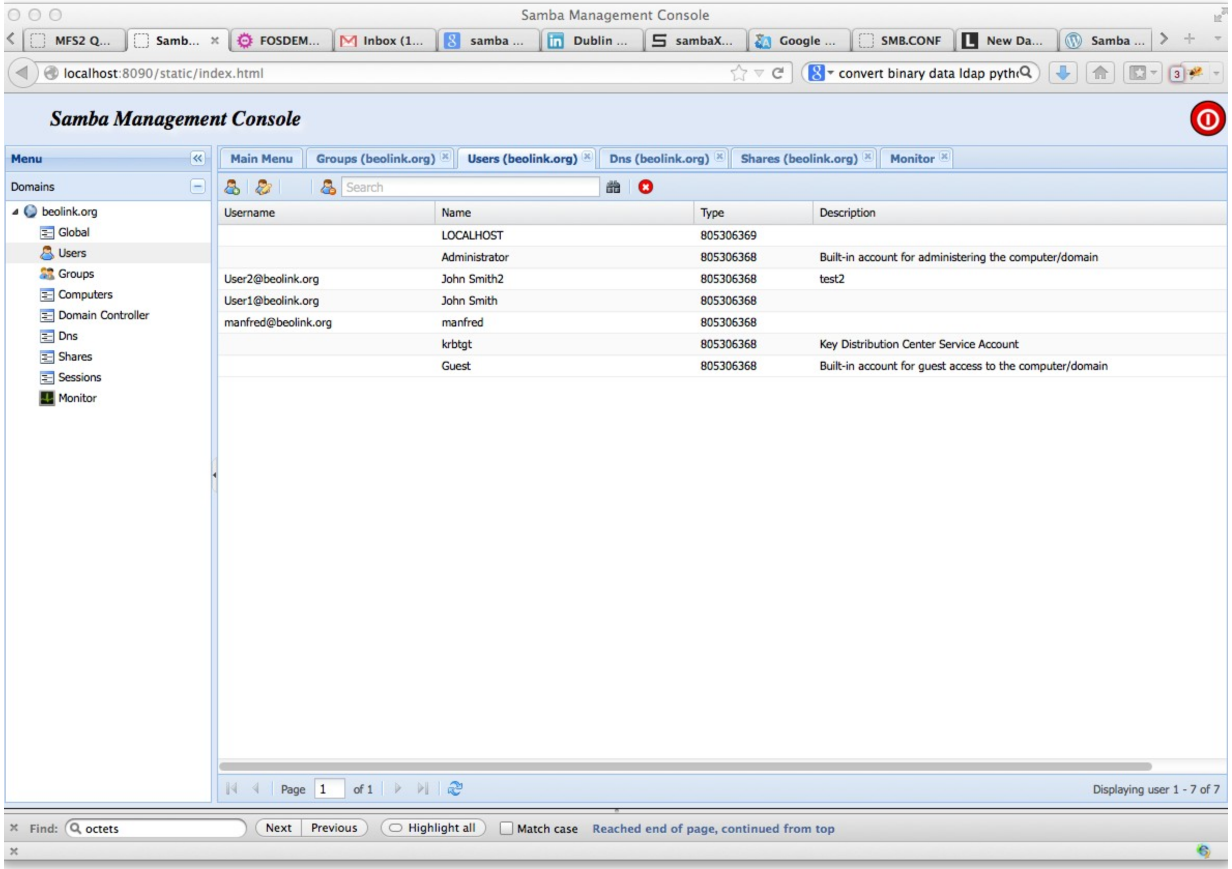
Internals: automation

The Rest interface gives all the functions needed for automation

- ❑ Retrieve information with simple url
- ❑ Interconnect to monitoring system
- ❑ User/group provisioning
- ❑ User Administration (password, lock..)

Architecture: Web Interface

Ajax base



Samba Management Console

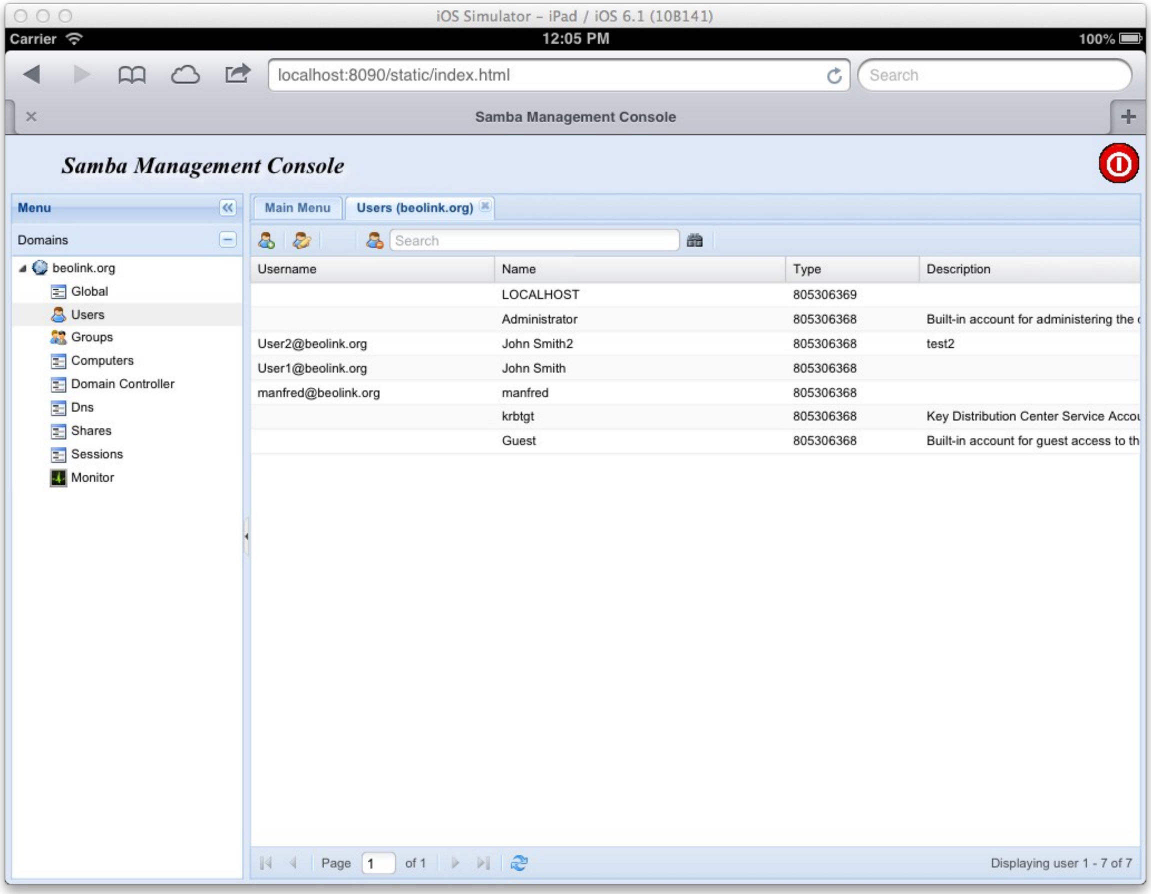
Username	Name	Type	Description
	LOCALHOST	805306369	
	Administrator	805306368	Built-in account for administering the computer/domain
User2@beolink.org	John Smith2	805306368	test2
User1@beolink.org	John Smith	805306368	
manfred@beolink.org	manfred	805306368	
	krbtgt	805306368	Key Distribution Center Service Account
	Guest	805306368	Built-in account for guest access to the computer/domain

Displaying user 1 - 7 of 7

Reached end of page, continued from top

Architecture: Mobile Device

Mobile devices



Username	Name	Type	Description
	LOCALHOST	805306369	
	Administrator	805306368	Built-in account for administering the
User2@beolink.org	John Smith2	805306368	test2
User1@beolink.org	John Smith	805306368	
manfred@beolink.org	manfred	805306368	
	krbtgt	805306368	Key Distribution Center Service Accou
	Guest	805306368	Built-in account for guest access to th

Displaying user 1 - 7 of 7

Architecture: User Experience

The Web interface gives all the functions and information present in the rest interface

- ❑ Asynchronous operation
- ❑ Simple to use
- ❑ Mobile device support
- ❑ Simple to extend
- ❑ Global View
- ❑ Multi REALM and SELF Service

Deploy : setup

- **Install**
  - Satisfy requirments (python/samba)
  - Uncompress the tarball

- **Configuration**
  - Define AD server
  - Define samba bin dir
  - Define http port

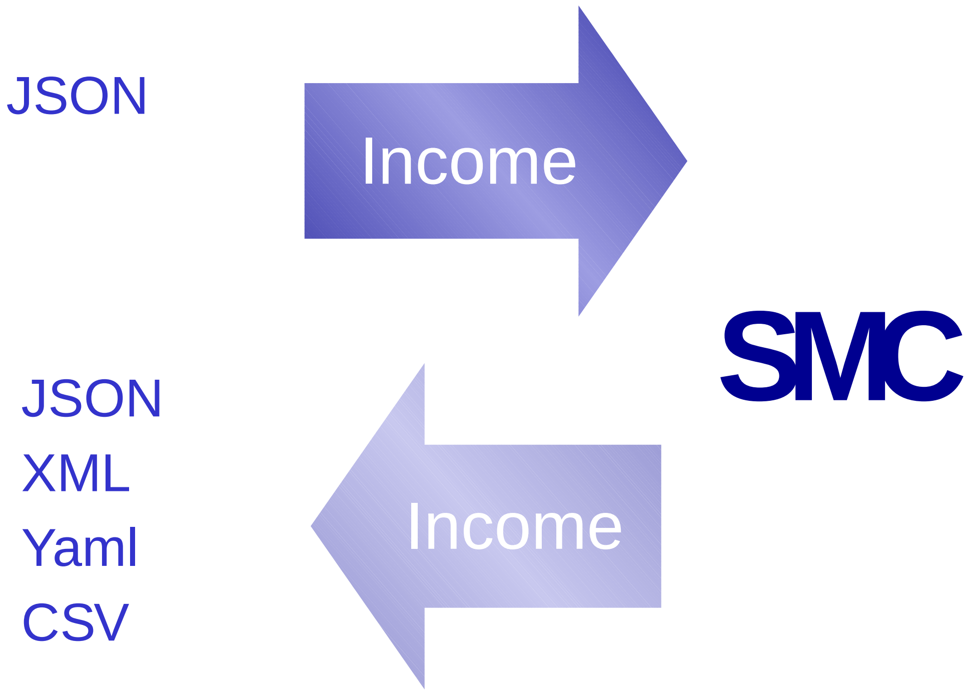
```
[global]
port=8080
smb_dir=/opt/samba
..


[servers]
servers=mysrv
```

- **Run**
  - Smcd –f configuration.file

- **Use**
  - Point your browser to the server

```
Python smcd.py –f smcd.conf -d
```

Internals: Data format



Short:

{
"description": ["test2"],
"userPrincipalName": ["User2@beolink.org"], "name": ["John Smith2"],
"sAMAccountType": ["805306368"],
"objectSid": ”[0']",
"sAMAccountName": ["User2"]
}

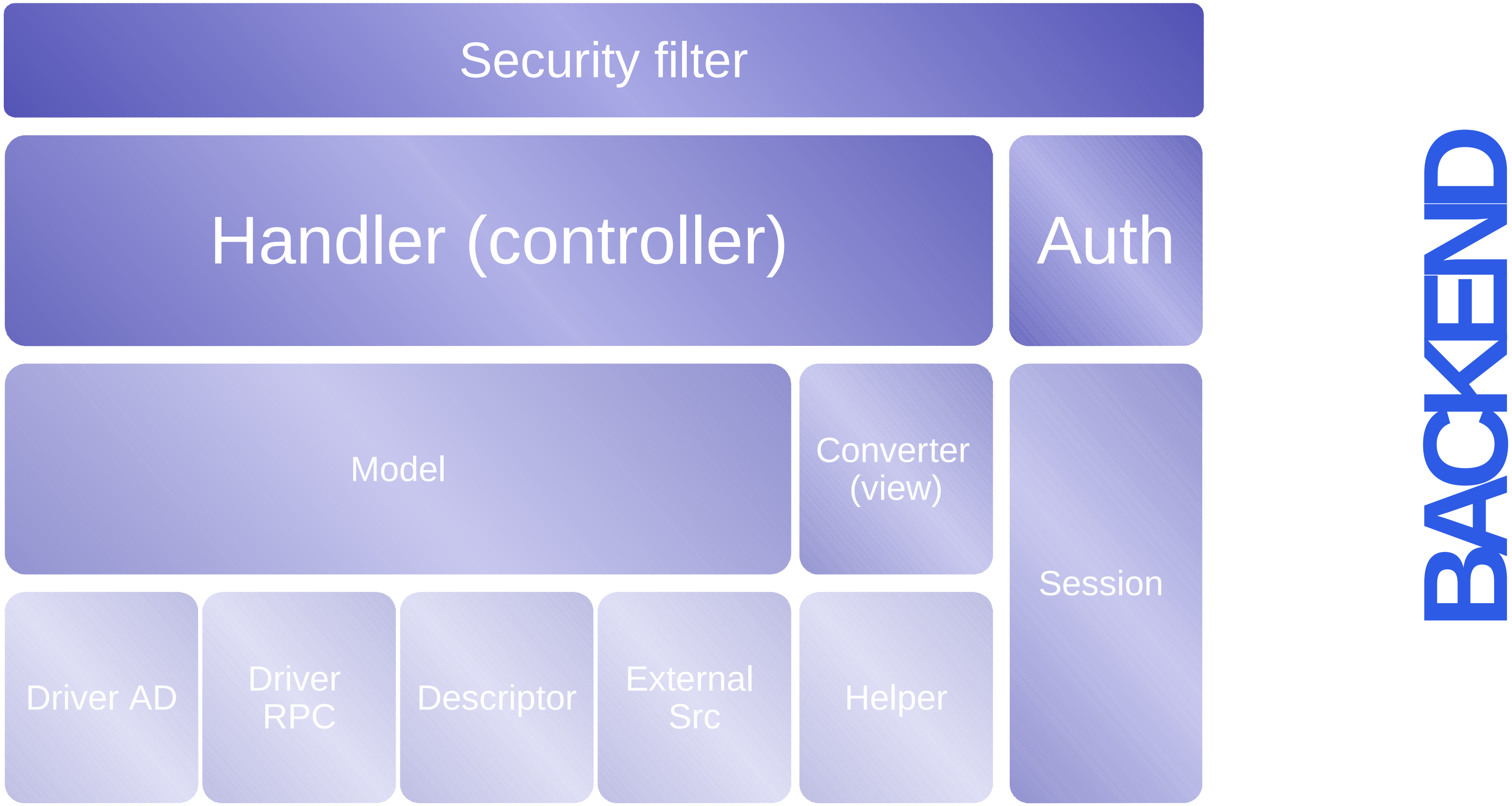
Extended:

{"telephoneNumber": ["000000"], "lastLogon": ["0"], "primaryGroupID": ["513"], "logonCount": ["0"], "description": ["test2"], "name": ["John Smith2"], "pwdLastSet": ["0"], "countryCode": ["0"], "userPrincipalName": ["User2@beolink.org"], "sAMAccountName": ["User2"], "whenChanged": ["20130506204926.0Z"], "badPwdCount": ["0"], "objectSid": ”[..]", "whenCreated": ["20130506204926.0Z"], "uSNCreated": ["3729"], "sn": ["Smith2"], "accountExpires": ["9223372036854775807"], "sAMAccountType": ["805306368"], "lastLogoff": ["0"], "badPasswordTime": ["0"], "cn": ["John Smith2"]}

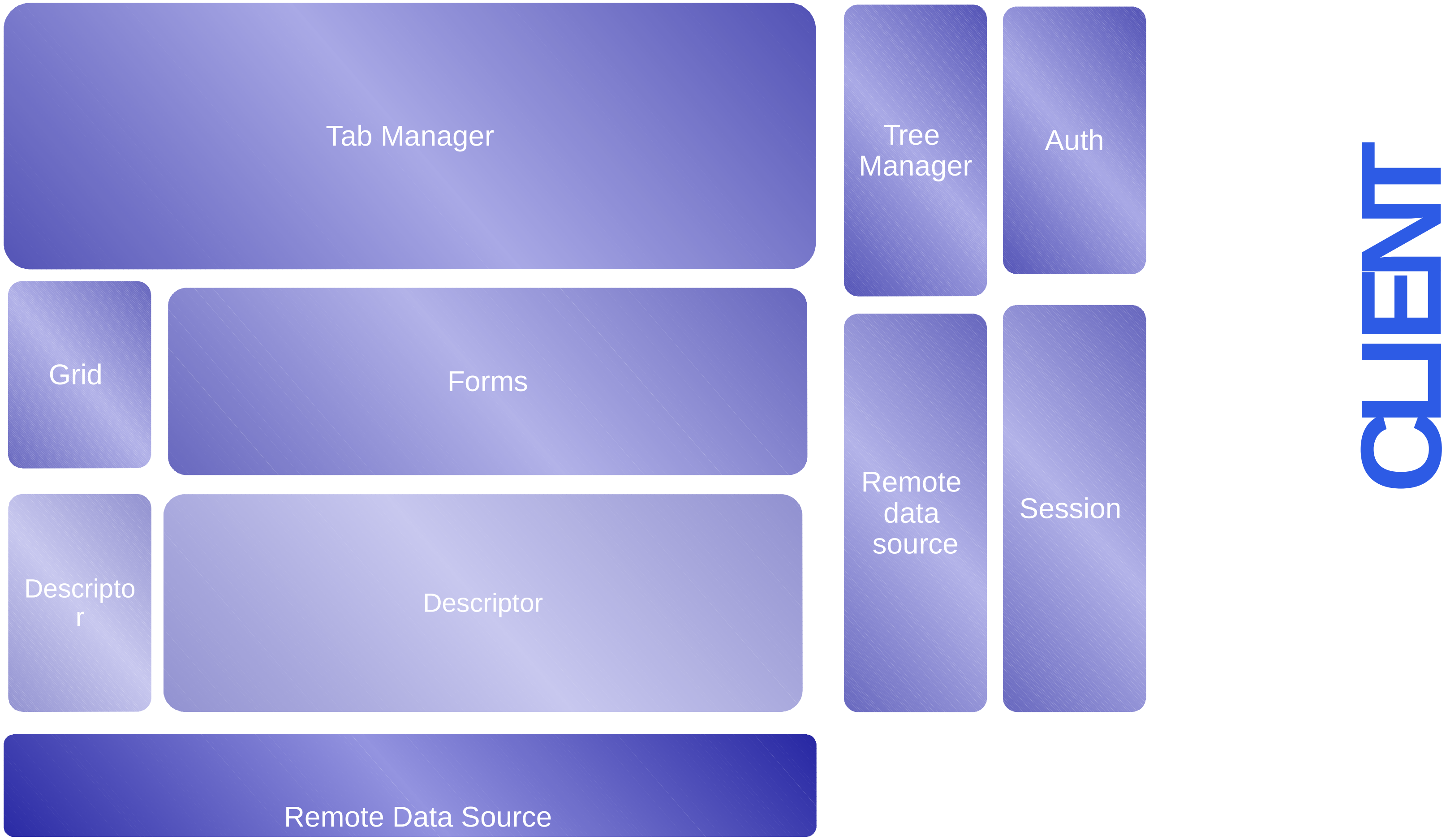
Dump:

…

Internals: Code organization



Internals: Code organization



Tab Manager

Grid

Forms

Descripto
r

Descriptor

Remote Data Source

Tree
Manager

Auth

Remote
data
source

Session

CLIENT

Internals: Descriptor

CLIENT

```
Dns_grid = ['name','description','dnsRecord','cn']
Dns_form_global=[‘Name’:'name’, ’Description‘:’description', ‘’Record :'dnsRecord', ‘’CommonName:'cn']
```

Server

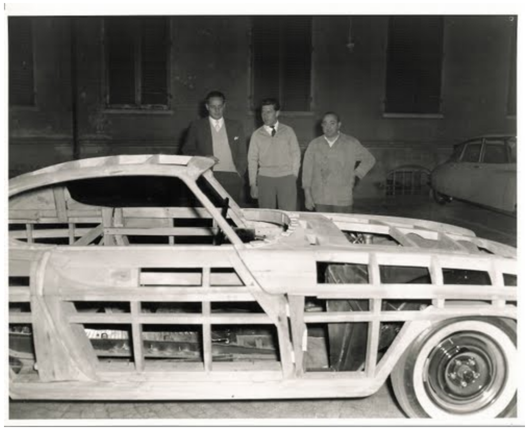
Ldapom

```
import ldapom
lc = ldapom.LdapConnection(uri='ldap://localhost:1389', base='dc=example,dc=com',
        login='cn=admin,dc=example,dc=com', password='admin')

node = lc.get_ldap_node('cn=f1ori,ou=people,dc=example,dc=com')
node
<LdapNode: cn=f1ori,ou=people,dc=example,dc=com>
node.givenName # show name
<LdapAttribute: givenName=Richter>
node.givenName = 'Meier' # change givenname
node.save() # save all changes
```

Status

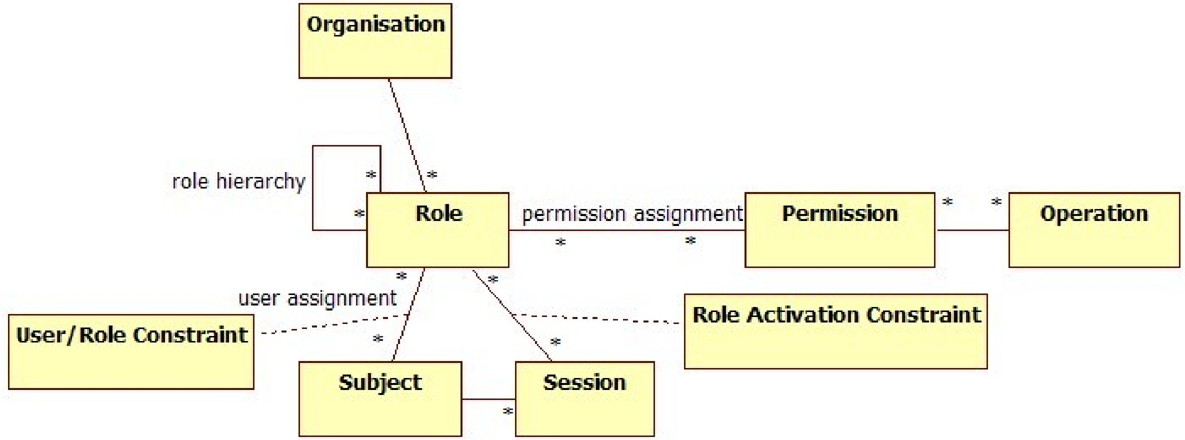
- **Requirements**
  - > samba 3.2 / samba4
  - > python 2.6
  - python ldap
  - Registry for config file

- **More prototype than product**
  - Status section completed
  - Users / groups readonly sections
  - Most Configuration section is ro
  - Centralized cache not implemented
  - Security !!!!
  - Migrate to Samba Python Binding in progress



Step ahead

- ❑ **Complete Identity Management**
  - ❑ RBAC model on the top
  - ❑ Many self service operation
  - ❑ Workflow approval
  - ❑ …



Try it… one day

First alpha will be ready for end of May (June) ✗

FeedBack

Code, ideas, testing, insults **…everything**

CloudVFS

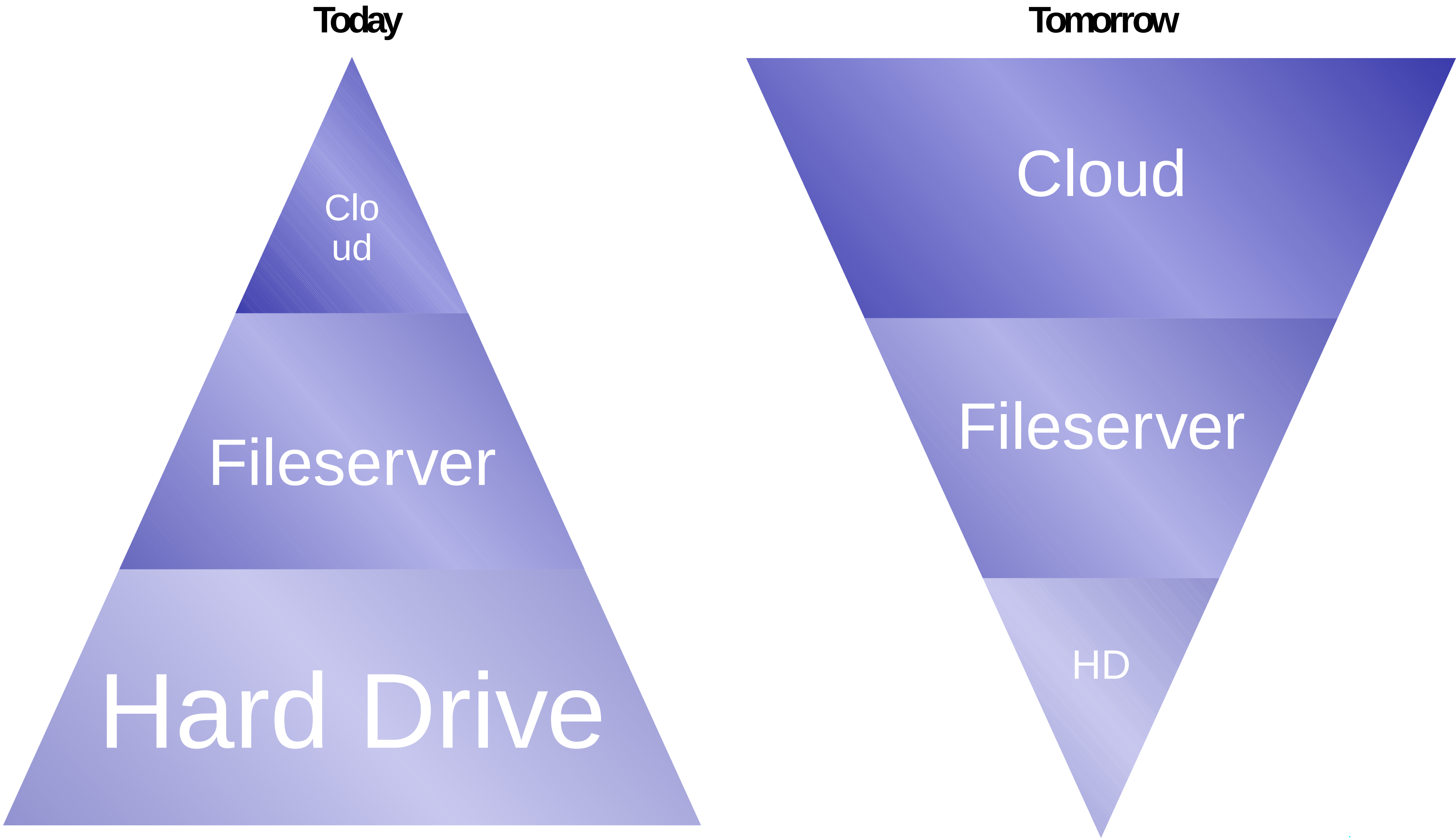
- S3FS was already taken

Where is your Data ?

Where is your music, photo, video, documents, projects, code, password, backups, ...

?

Many places


Today
Clo
ud
Fileserver
Hard Drive
Tomorrow
Cloud
Fileserver
HD

One day ..

Why we have a RAID for HD and don’t we have for Cloud ?

Why the advanced NAS can distributed the data across different pool and don’t we have the similar solution for Cloud ?

CloudVFS

- Cloud storage Link

  The share is the bucket or container of the cloud storage

- Backup / Disaster recovery

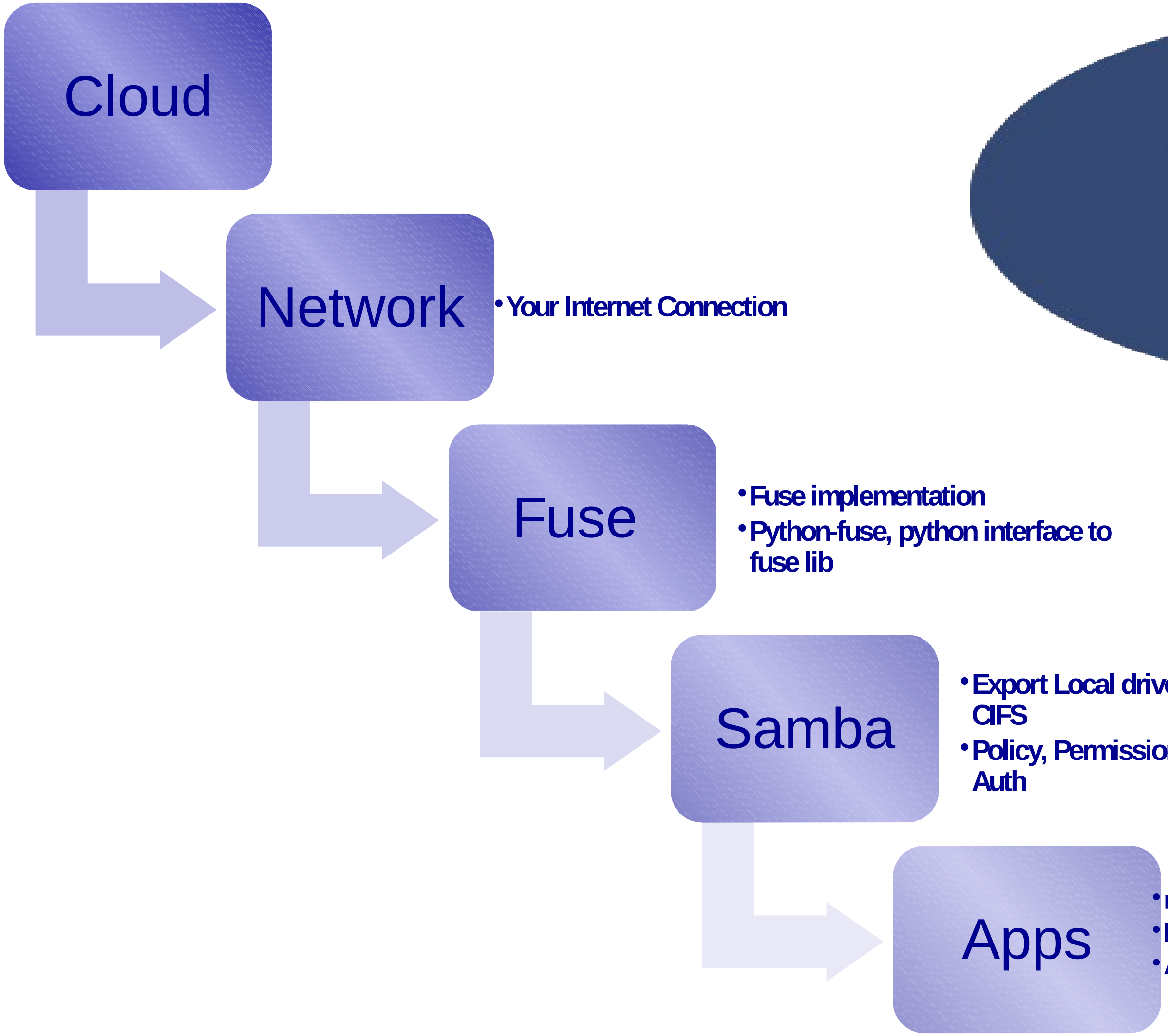
  The module copy all the traffic on the share to a cloud storage

- Transparent distribution of the data

  Mix up local and cloud storage space

- S3FS was already taken

Why don’t you use FUSE?



Cloud

Network

- Your Internet Connection

Fuse

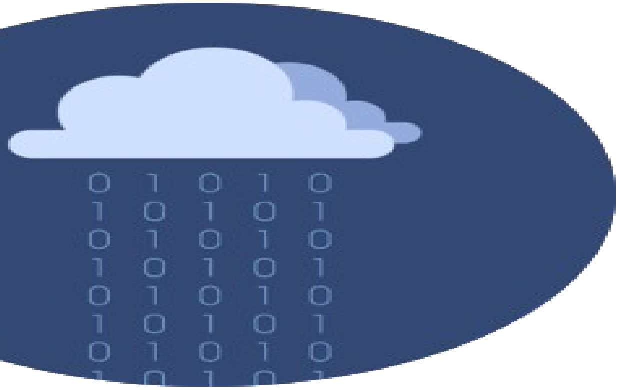
- Fuse implementation
- Python-fuse, python interface to fuse lib

Samba

- Export Local drive via CIFS
- Policy, Permission, Auth

Apps

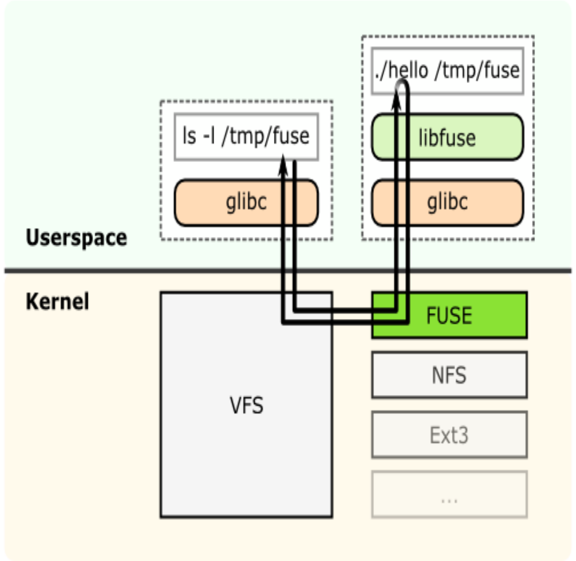
- rsync
- lvm
- Amanda/Bacula



Fuse

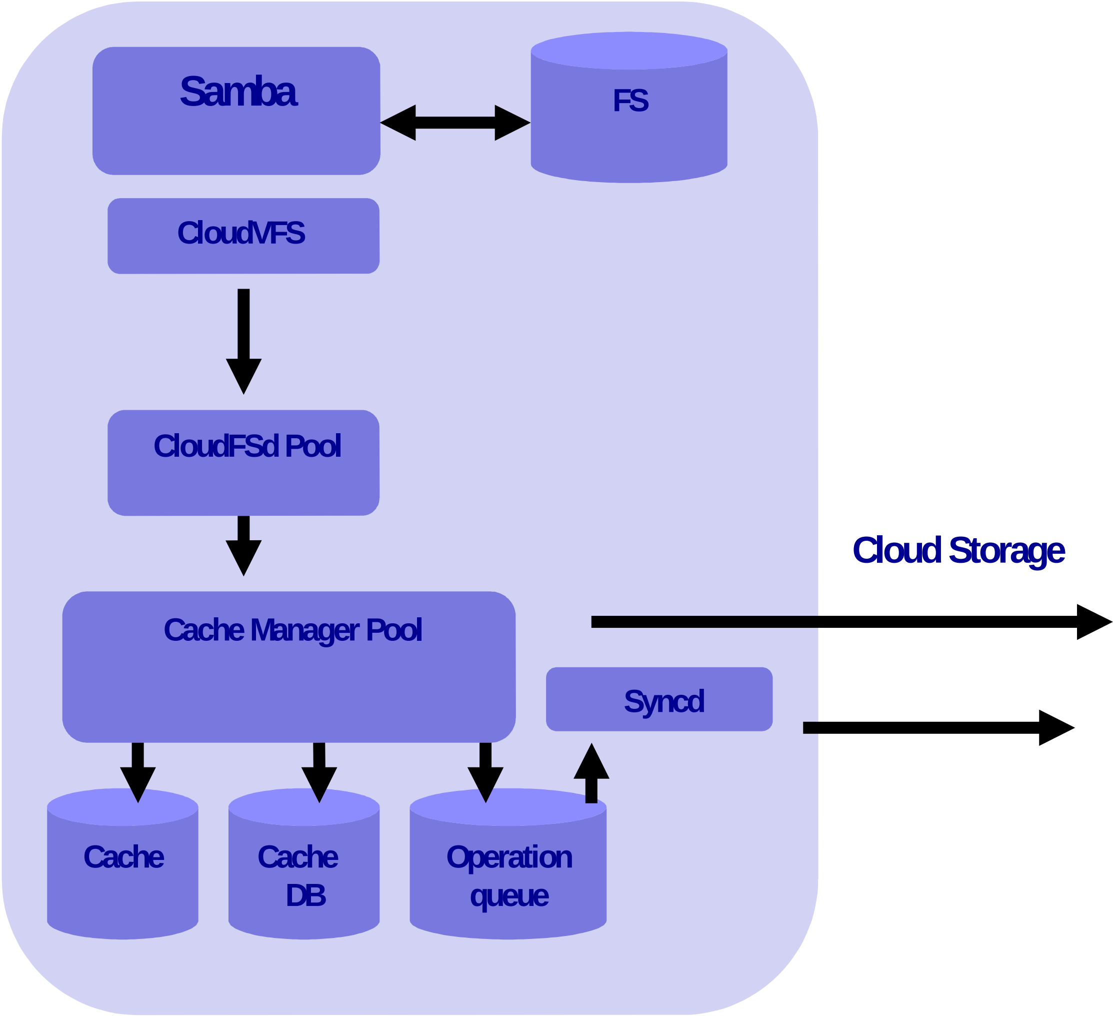
FUSE , filesystem in user space

- ❑ A lot of calls and layers
- ❑ No control of network layer
- ❑ No Queue primitives
- ❑ Difficulties in Error management
- ❑ No Cache optimization
- ❑ No Lock optimization
- ❑ No control on the commit



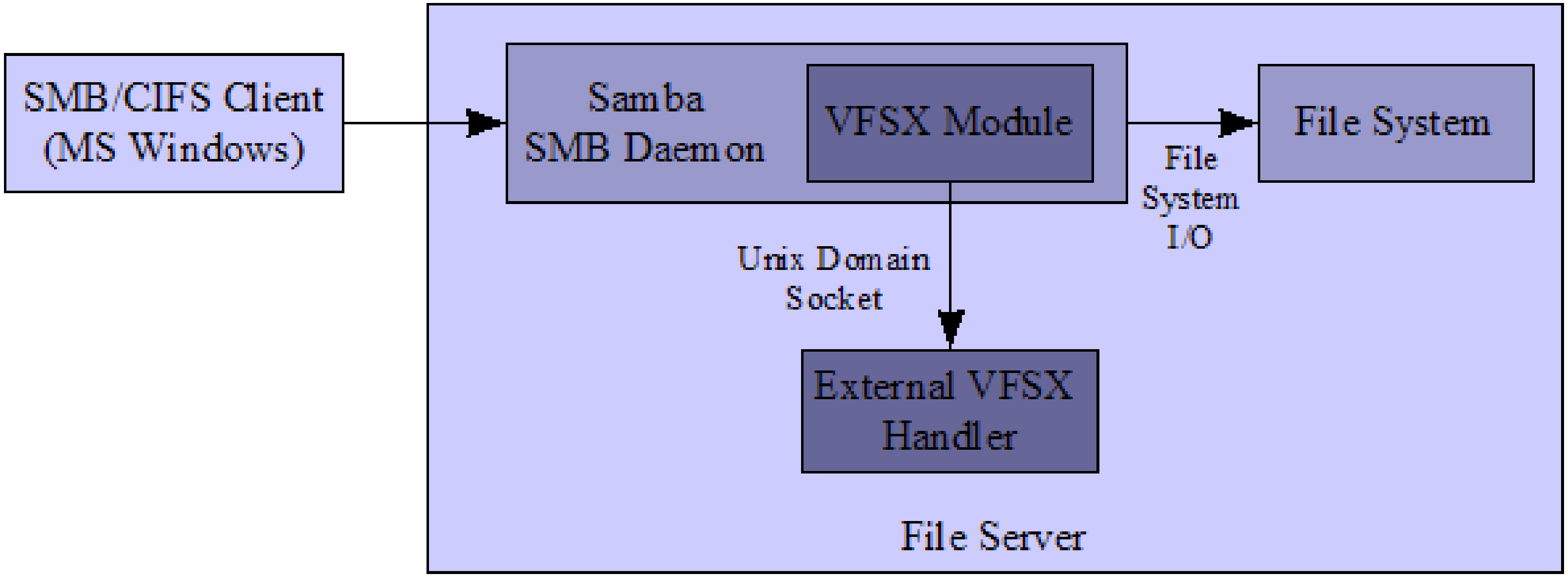
CloudVFS Daemon

Element	Configuration
Interface	Socket
Process	Many socket reader
Thread	Any operation is handle by a single thread
Cache	Async operation
Network	Async operation



Base on VFSX idea

VFSX is a transparent Samba Virtual File System (VFS) module which forwards operations to a process on the same machine for handing outside of the Samba daemon process…



Solution II (Experimental)

1 Intercept

```
static int vfsx_mkdir(vfs_handle_struct *handle, const char *path, mode_t mode)
{
        int result = -1;
        int count;
        char buf[VFSX_MSG_OUT_SIZE];

        count = snprintf(buf, VFSX_MSG_OUT_SIZE, "mkdir:%s:%s:%s,%d", handle->conn->user, handle->conn->origpath, path, mode);
        if (vfsx_execute(buf, count) == VFSX_SUCCESS_TRANSPARENT) {
                result = SMB_VFS_NEXT_MKDIR(handle, path, mode);
        }
        return result;
}
```

2 Check Socket

```
if (!connected) {
        sd = socket(AF_UNIX, SOCK_STREAM, 0);
        if (sd != -1) {
        strncpy(sa.sun_path, VFSX_SOCKET_FILE,                    strlen(VFSX_SOCKET_FILE) + 1);
        sa.sun_family = AF_UNIX;
        ret = connect(sd, (struct sockaddr *) &sa, sizeof(sa));
        …
```

3 Write/Read on the socket

```
memset(out, 0, VFSX_MSG_OUT_SIZE);
        strncpy(out, str, strlen(str) + 1);
        ret = write(sd, out, VFSX_MSG_OUT_SIZE);
        if (ret != -1) {
            memset(in, 0, VFSX_MSG_IN_SIZE);
            ret = read(sd, in, VFSX_MSG_IN_SIZE);
            if (ret != -1) {
                result = atoi(in);
```

VFS Samba Module

Solution II (Experimental)

Smb.conf

```
[myshare]
comment = My share
path = /home/myuser/shared/
valid users = …
….
read only = No
vfs objects = cloudvfs
```

Samba Conf

Solution II (Experimental)

Python Server

```
…
  while True:
         msg = self.request.recv(512)
         if not msg: break
         log.debug(msg)
         # Handle message-parsing and operation execution error here.
         # Socket communication errors should be propagated.
         try:
         (operation, user, origpath, args) = self.__parseMessage(msg)
         result = self.__callOperation(operation, user, origpath, args)
         except Exception, e:
         result = VFSOperationResult(FAIL_ERROR)
         log.exception(e)
         self.request.send("%d" % result.status)

         # The client probably closed the connection.
         self.request.close()
         log.debug("Close Connection”)

def __parseMessage(self, msg):
         parts = msg.split(":")
         (operation, user, origpath) = parts[0:3]
          log.debug("  operation = '%s' user = '%s' origpath = '%s'" %
                          (operation, user, origpath))
           args = []
           if len(parts) > 3:
         args = parts[3].split(",")
         log.debug("  args = '%s'" % parts[3])
           return (operation, user, origpath, args)
```

Message Format:

"user:operation:origpath:arg1,arg2,arg3”

Open Points

- ❑ Init Phase (sync)
- ❑ Optimization write on close
- ❑ Bandwidth Management
- ❑ Replication with multiple site
- ❑ Sanity Check
- ❑ Monitoring
- ❑ Managements tools
- ❑ Samba4

* Under development

Roadmap

- **0.1 Not Released**
  Prototype
  Transparent mode
  Only S3 supported

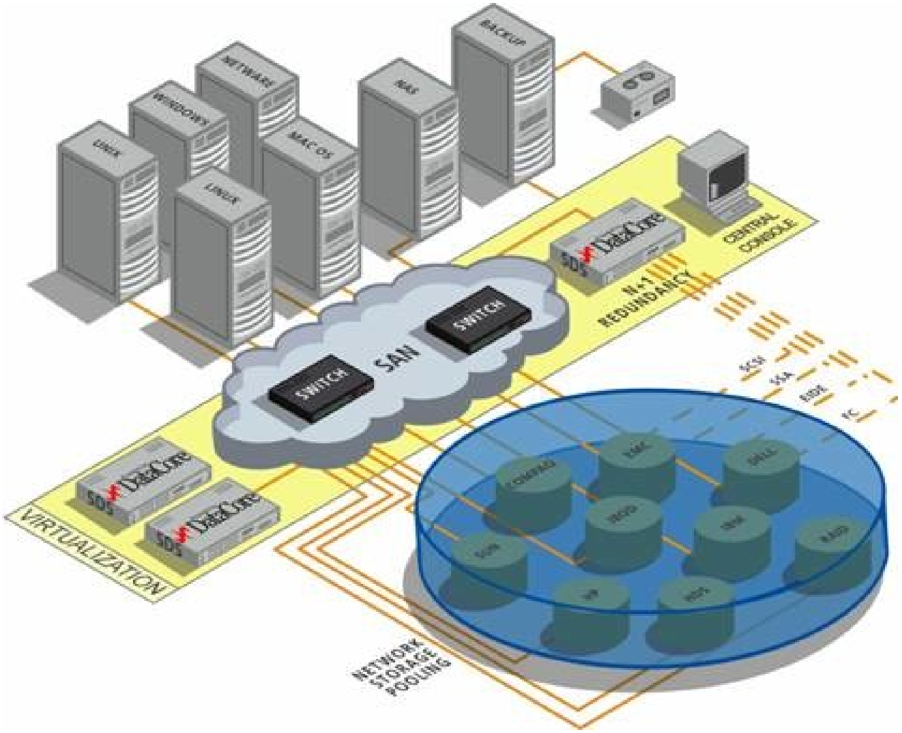
- **0.2 First Public**
  Disaster recovery (transparent replication)
  Others Cloud storage

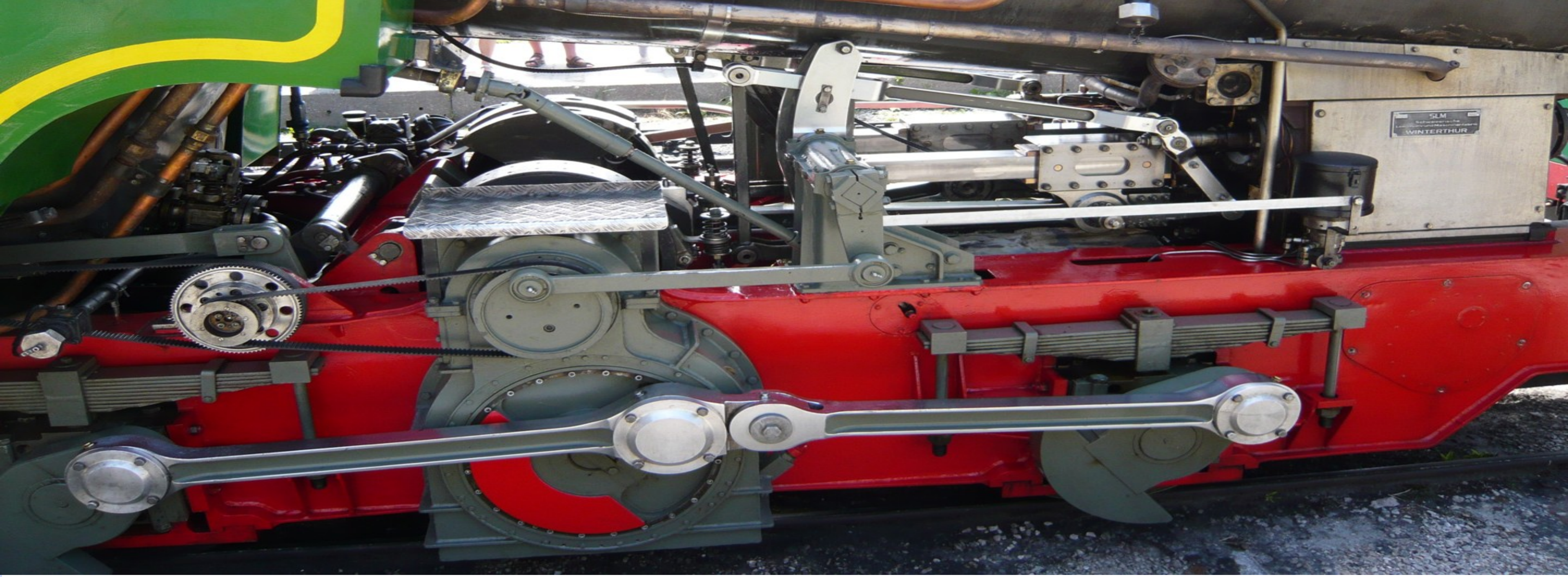
- **0.3 xxx**
  Automatic distribution
  Replication drive by users (with specific permission to user or group/xattr)
  Optimization

- **Next**
  Keep in sync the cache across different fileserver, permission, access based on the devices, C version, samba4 VFS, complete module …

All that is under development with very limited resources

What is the future ?





Thank you

http://www.beolink.org

manfred.furuholmen@gmail.com
peppeguarino@gmail.com

Beolink.org