Using samba base libraries

SSSD as an example application

Simo Sorce

Samba Team,
Red Hat, Inc.

samba eXperience

Introduction

The goal of this talk is to show how some of the samba base libraries can be used and integrated successfully in other projects.

We use SSSD as an example application.

Why samba libraries?

There are a few reasons why I decided to use the samba libraries in this project.

Part of it is because I wanted to use LDB as our internal directory. LDB depends on talloc, tdb and tevent.

Part of it is because I think these libraries are extremely useful and powerful, and has been largely tested within samba.

Also initially SSSD was going to use a process model close to that of samba4. It meant a lot of infrastructure was already in place w/o having to reinvent all from scratch.

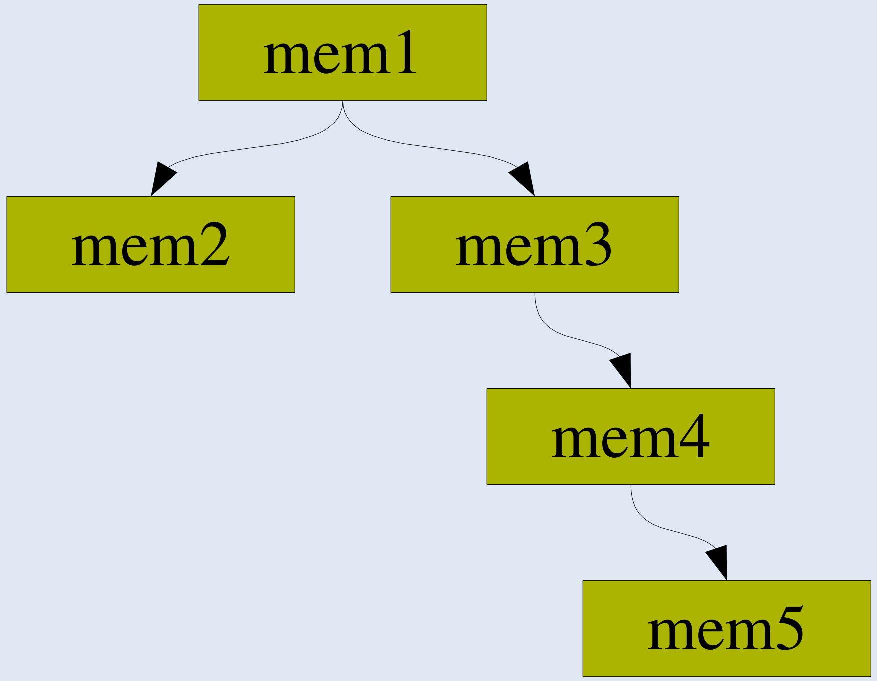
So what is great with these libraries ?

Talloc

Talloc - memory hierarchies

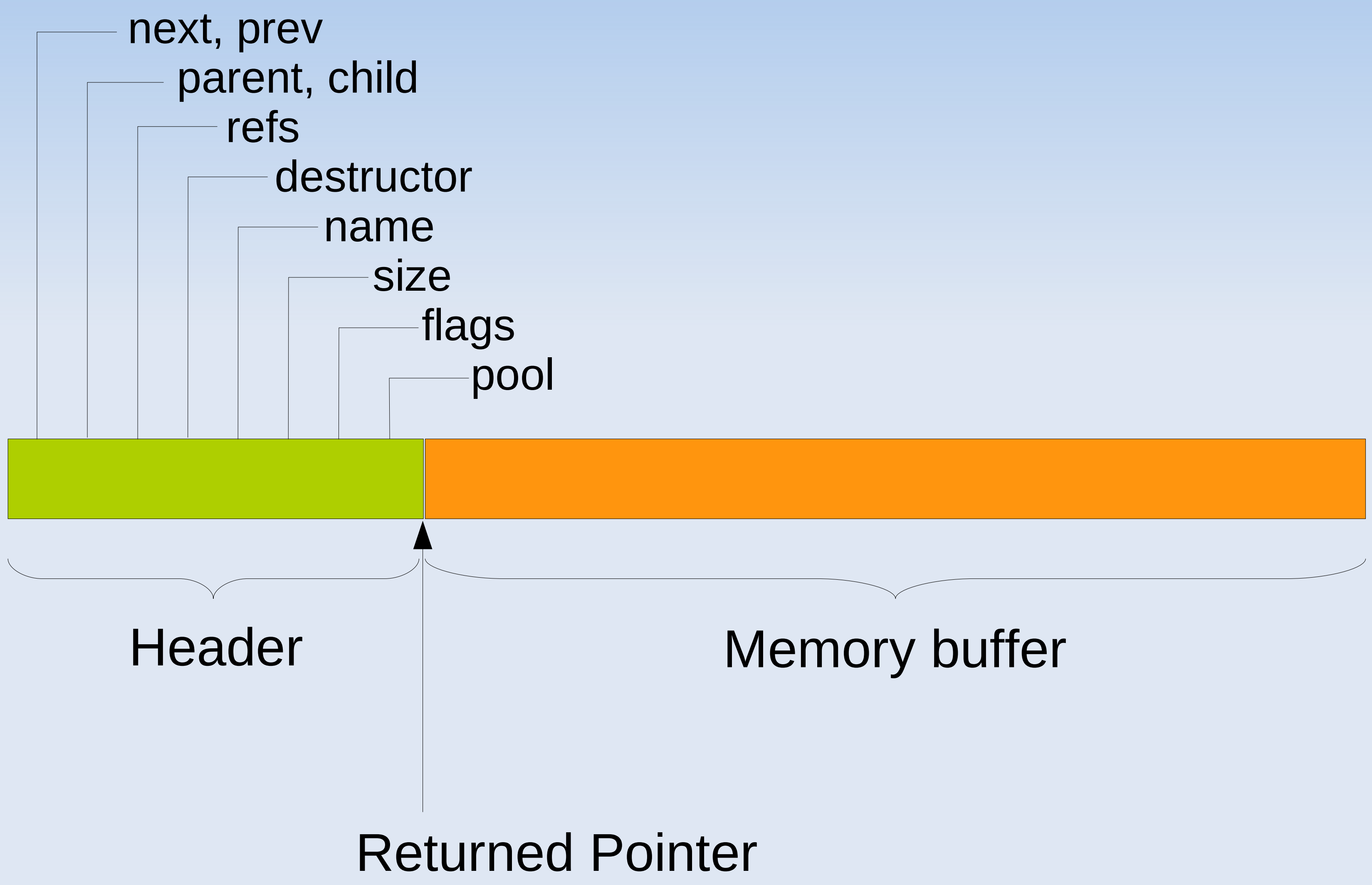
Talloc is samba's memory allocation library.

- Does not replace malloc, but (so far) uses malloc underneath
- Each memory allocation with talloc generates a talloc context
- Each talloc context can be a child of another context
- Each talloc context can be a parent of other contexts



```
mem1 = talloc_new(NULL);
mem2 = talloc(mem1, foo_t);
mem3 = talloc(mem1, bar_t);
mem4 = talloc(mem3, baz_t);
mem5 = talloc_strdup(mem4, “last”);
```

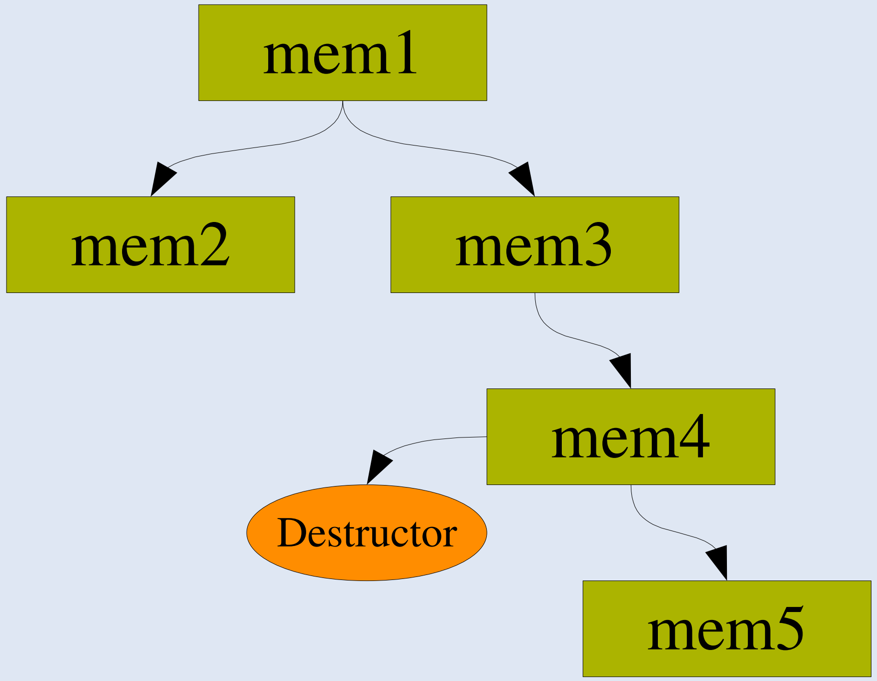
Talloc - anatomy



Talloc – memory objects

Why are memory hierarchies useful ?

- Deep function nesting and cleanup on error conditions
- State/Context based code (object oriented :-)
- Async code
- Destructors



```
int destructor4(void *mem) {
    struct fid *mem4 = mem;

    close(mem4->file_descriptor);
    Return 0;
}
...

talloc_set_destructor(mem4, destructor4);

...

talloc_free(mem1);
```

Tevent

Tevent

Tevent is a recent addition to the pool of 'T'-libraries.

It is an implementation of an event engine:

- extremely useful in event driven, async programs (such as Samba or SSSD)

- supports different backends for polling file events (based on epoll, or select, or DIY)

- support non-file based events too
    - timers
    - signals

- uses talloc for all memory management

Tevent - The event context

The main actor in tevent is the event context
- holds all references to the currently instantiated events
- returned by the init function:

```
struct tevent_context *tev;
TALLOC_CTX *mem_ctx;
...
tev = tevent_context_init(mem_ctx);
```

It is needed to instantiate new events:

```
/* a new timer event */
timer = tevent_add_timer(tev, mem_ctx, timeval, callback, private_data);
```

It is normally the key context in the main loop:

```
/* this loops until there are events */
tevent_loop_wait(tev);
```

Tevent - Events

Tevent supports 3 main events types

file related events (uses epoll/select):

```
void recv_data(struct tevent_context *ev, struct tevent_fd *fde,
               uint16_t flags, void *private_data);
...
fd = socket(AF_UNIX, SOCK_STREAM, 0);
connect(fd, (struct sockaddr *)&namedpipe, sizeof(namedpipe));
fde = tevent_add_fd(tev, mem_ctx, fd, TEVENT_FD_READ, recv_data, pvt);
```

Once you have a file handler you can tell tevent to check if the file descriptor is readable and/or writeable and to call a function when that happens.

Tevent – Events (2)

Timer events:

```
void run_alarm(struct tevent_context *ev, struct tevent_timer *te,
               struct timeval current_time, void *private_data);
...
timer = tevent_add_timer(tev, mem_ctx, timeval, run_alarm, pvt);
```

A timer event lets you tell tevent to run a function after waiting an arbitrary amount of time.
Very useful to handle situations where you have to wait before retrying an operation over the network
Also extremely useful to schedule task you want to run once in a while (all kind of checks against remote servers, or other resources that cannot be polled through a file descriptor)

Tevent – Events (3)

signals (in a safe way):

```
void my_sighup(struct tevent_context *ev, struct tevent_signal *se,
               int signum, int count, void *siginfo, void *private_data);
...
sig = tevent_add_signal(tev, mem_ctx, SIGHUP, 0, my_sighup, pvt);
```

Signals are normally extremely annoying to deal with.
You can't do many operations within a signal handler (you can't use malloc() for example).
With tevent it becomes very simple to handle signals as just any other event.
Tevent catches and stores the fact that an event has happened and calls your signal handler as soon as control is returned to the main event loop.

TDB

TDB – Trivial/Tiny DataBase

- TDB is a very mature component of Samba
  - Used in samba to share all kind of information between process
  - Used also as storage for data (like accounts, secrets, printers)
  - The API/ABI and file formats are quite stable *.

- It is a very fast database.
  - Uses MMAP (also support file operations).
  - It's basically a huge hash table.
  - Supports multiple write locks on separate records.
  - Since some time now it also support transactions.

TDB – Trivial/Tiny DataBase

But this is not a general purpose database
- Similar to BDB
- Supports only single key operations
- Data is treated as a blob.
- The app is responsible for formatting the data records.
- Max theoretical size 4GB
- But only if you have 4GB of address space available ...

Continued development
- Recently there has been interest in making it thread safe
- Some work is being done on supporting nesting transactions
- (Howard Chu (OpenLDAP) sent some interesting patches)

LDB

LDB – Ldap-like DataBase

- LDAP-like
  - Data stored in entries.
  - Entries are organized in a directory.
  - Each entry can have any number of attributes.
  - Powerful search filters.
  - Supports plug-ins to add functionality.

- But unlike LDAP
  - Can store everything in a single local file
    - (a tdb for example)
  - Does not enforce a schema
  - Can work both as a server and as a client library
    - (EX: against a remote LDAP server)
  - Supports transactions
    - (only when a tdb is used for now)

Integration Challenges

Integration challenges

Library API/ABI stability

Interfaces documentation

Libraries packaging

Build system

Mixing std malloc and talloc

DBUS

DBUS – The System bus

DBUS is not a samba library

But it is a very interesting IPC mechanism

It is also needed to “talk” to the rest of a modern linux system.

Unfortunately it provided a few integration challenges:

- Integration with Tevent – done!
- Integration in a fork-w-no-exec model – problematic!

For some reason DBUS upstream insist that just forking w/o immediately exec()ing something is not a working model they want to support.

SSSD

What is SSSD

SSSD stands for System Security Service Daemon.

It is a client agent that implements NSS and PAM (*)

- isolate applications from the network
- implement a better, intelligent caching
- implement offline capabilities for all backends
- hopefully improve authentication configurations

Initial urge was to replace modules like nss_ldap.

Many aspects of SSSD have been inspired by Winbindd
But it is **not** a Winbindd replacement
I plan to use libwbclient to use winbindd as a backend in future.

(*) It also implements other interfaces over DBUS but they are not yet stable

Why nss_ldap is bad ?

Problems with nss_ldap:

- includes ldap libraries
  - Requires to link them statically to avoid symbol issues
  - Requires a rebuild when there is an issue with ldap libraries

- lives within the process space
  - Very complex code, a bug means we can crash the app.

- no pooling of connections
  - A server with many processes may end up with **many** connections to the remote LDAP server unnecessarily

- access credentials
  - nss_ldap lives in the user process space, any credential needed to access the ldap server is exposed to all users

Nsswitch.conf is never reloaded, requiring a restart of all applications on configuration changes or modules updates

What is “wrong” in nsswitch ?

Things nsswitch gets wrong in today's networked computing:

- It assumes data is always present, available, and reachable without significant delays.
- It has no concept of identity domains, all users/groups are stored in a single name space.
- It assumes enumerating users/groups is possible/desirable
- Configuration is loaded only at application startup
- All code runs within the application process.

What is “wrong” in PAM ?

Things PAM gets wrong:

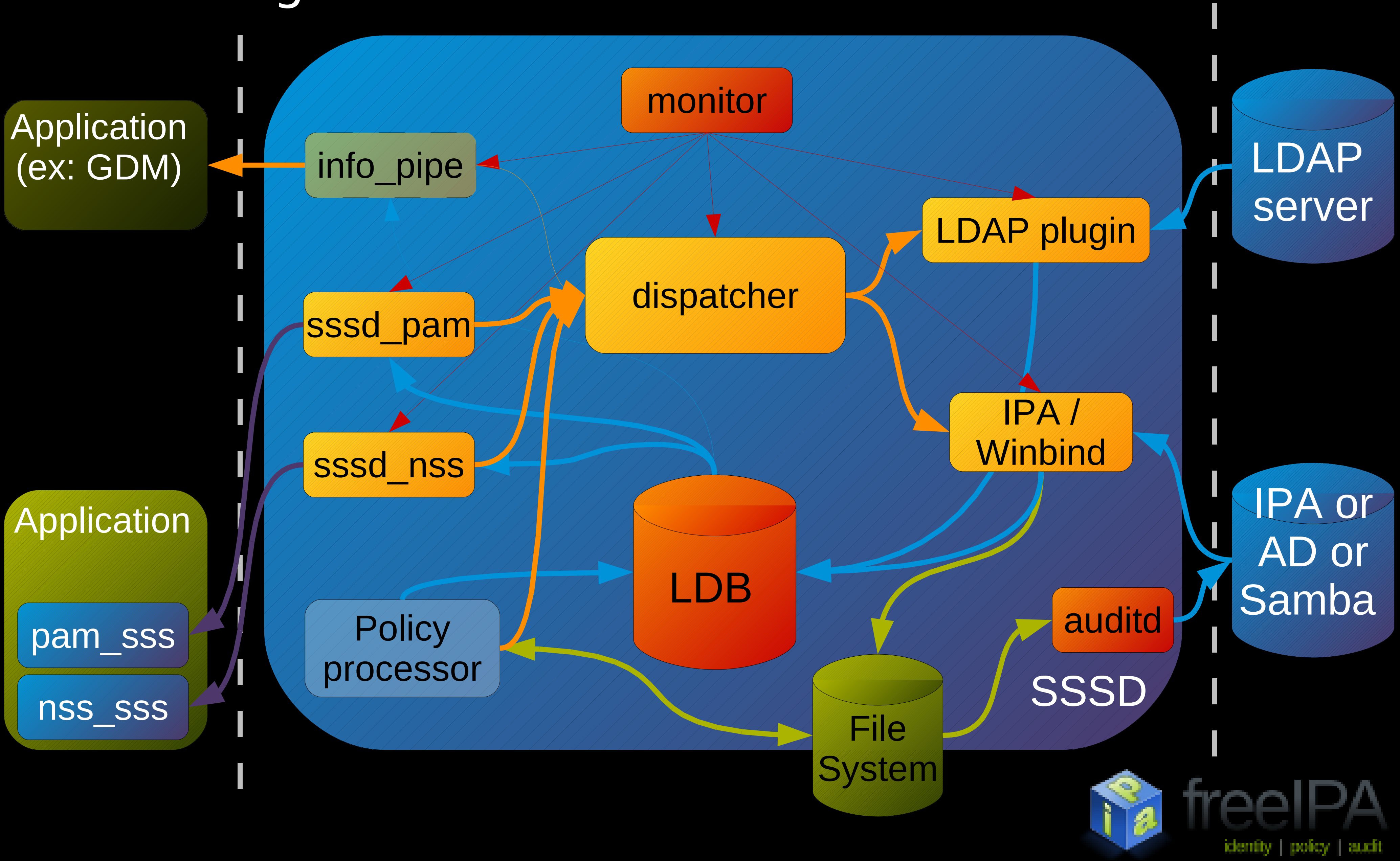
- It assumes a text interface and a login/password access scheme
- It has no concept of identity/authentication domains.
- Has a static interface, does not cope with dynamic environments
- It is too complex and yet does not provide much flexibility

SSSD - Status

SSSD is very young
(Name may still be subject to change)

- Still a lot of work in progress
  - Supports only passwd and group tables for NSS
  - Does not have any working native remote backend yet
- But has some good things already
  - Support caching users and groups
  - Support group nesting (implemented in the LOCAL backend)
  - Offline mode
  - Credentials caching

SSSD diagram



Thank you!

Thank YOU for attending this talk.

A big thank you to SerNet and the SambaXP Sponsors and my employer (Red Hat) for the opportunity they give us all years to meet this wonderful community.

Contacts

Simo Sorce

Samba Team <idra@samba.org>
Red Hat, Inc. <ssorce@redhat.com>