DFS Replication

A client implementation for Samba

Samuel Cabrero
SUSE Labs Samba team
scabrero@suse.com

Agenda

1. Introduction
2. DFS-R Configuration
3. Protocol overview
    1. Retrieving updates
    2. Processing updates
    3. Installing updates
4. Demo
5. Next steps

Introduction

Overview

- **DFS-R is a RPC protocol that replicates files between servers**
- **It is a optimistic and multi-master protocol**
  - Optimistic → Files can be updated without prior consensus
  - Multi-master → Files can be changed in any server.
- **Asynchronous, no restrictions on when the changes must be propagated**

- **Files are replicated when the application that modifies them closes the file**
- **When a file is closed, an update is generated and inserted in a database**

DFS-R configuration

Concepts

- **Replication Groups** or replica sets
- **Replicated folders** or content sets
- Computers are **members** of replication groups
- Members **subscribe** to replicated folders
- Topology, which define the **connections** between members, is common to the group
- Configuration is stored in AD
  - Global configuration

```
CN=DFSR-GlobalSettings,CN=System,DC=samba1,DC=ad
      CN=SambaXP_TestGroup                                    <= msDFSR-ReplicationGroup
            CN=Content                                        <= msDFSR-Content
                  CN=SambaXP_TestFolder1                      <= msDFSR-ContentSet
                  CN=SambaXP_TestFolder2                      <= msDFSR-ContentSet
            CN=Topology                                       <= msDFSR-Topology
                  CN=bc7d1b34-bdac-48e3-9a86-225bcfcc96d7     <= msDFSR-Member
                        CN=d7579220-73c3-4c00-b479-347c29576e90   <= msDFSR-Connection
                  CN=2d13008d-d7f7-47d8-b7df-5ae90c617cf4     <= msDFSR-Member
                        CN=b9c16269-1136-49d2-85cc-2cb75114d14c   <= msDFSR-Connection
```

  - Local configuration

```
CN=DFSR-LocalSettings,CN=WIN2K12R2-2,CN=Computers,DC=samba1,DC=ad
      CN=bc7d1b34-bdac-48e3-9a86-225bcfcc96d7                 <= msDFSR-Subscriber
            CN=18b1586c-232b-4459-98b3-390939c96b8c           <= msDFSR-Subscription
            CN=9fdb9b8c-f88e-4438-b689-14a06bbe5c1a           <= msDFSR-Subscription
```

The SYSVOL replication group

- **It is a special replication group**

```
CN=DFSR-GlobalSettings,CN=System,DC=samba1,DC=ad
      CN=Domain System Volume
      msDFSR-ReplicationGroupType: 1

      CN=SambaXP_TestGroup
      msDFSR-ReplicationGroupType: 0
```

- **Replication topology follows nTDSConnection from Configuration partition (AD replication)**

Management

- **PowerShell**
  - {New,Get,Set,Remove}-DfsReplicationGroup
  - {New,Get,Set,Remove}-DfsReplicatedFolder
  - {Add,Get,Set,Remove}-DfsrMember
  - {Get,Set}-DfsrMembership
  - {Add,Get,Set,Remove}-DfsrConnection

- **samba-tool**
  - samba-tool dfsr group {list,create,edit,delete}
  - samba-tool dfsr folder {list,create,edit,delete}
  - samba-tool dfsr member {list,add,delete}
  - samba-tool dfsr subscription {list,add,delete}
  - samba-tool dfsr connection {list,create,edit,delete}

Protocol overview

Overview

- **The protocol takes a three tiered approach**
  - The client determine which versions is missing
    - Asking for the server's Version Vectors (VV)
  - The client ask for the missing updates
    - Asking the server for the Updates
  - The client download the file's data

Version Vectors

- **Define a range of updates from the same server**
- **Pair of server's DB GUID – range of updates**
- **Versions [0 – 8] are reserved**
- **Version 1 represent the replicated folder root**

```
version_vector: struct frstrans_VersionVector
      db_guid                  : 6ff04912-7f6c-4147-a3f9-6231534d919b
      low                      : 0x0000000000000009 (9)
      high                     : 0x000000000000000b (11)
```

Updates

1. Get version vectors (VVs)

```
version_vector: struct frstrans_VersionVector
      db_guid                  : 6ff04912-7f6c-4147-a3f9-6231534d919b
      low                      : 0x0000000000000009 (9)
      high                     : 0x000000000000000b (11)
```

2. Compute VV delta between the known VV and received VV

3. Get updates in the computed delta

```
frs_update: struct frstrans_Update
      present                  : 0x00000001 (1)
      name_conflict            : 0x00000000 (0)
      attributes               : 0x00000010 (16)
      fence                    : Thu Jan  1 00:00:00 1970 UTC
      clock                    : Wed Apr 25 10:16:15 2018 UTC
      create_time              : Wed Apr 25 10:15:55 2018 UTC
      content_set_guid         : 18b1586c-232b-4459-98b3-390939c96b8c
      sha1_hash                : 6f7860df40d05f1187414712fa730c8ad1d8c7a8
      rdc_similarity           : 00000000000000000000000000000000
      uid_db_guid              : 18b1586c-232b-4459-98b3-390939c96b8c
      uid_version              : 0x0000000000000001 (1)
      gsvn_db_guid             : ae0da2be-8a27-4e0d-9ecd-06f64efcf24a
      gsvn_version             : 0x0000000000000020 (32)
      parent_db_guid           : 00000000-0000-0000-0000-000000000000
      parent_version           : 0x0000000000000000 (0)
      name                     : 'Folder2'
      flags                    : 0x00000000 (0)
```

Updates

- **Each created file is assigned a Unique Identifier (UID)**
  - A UID is a pair GUID – Version number
  - The UID is used to track the object for its entire lifetime (moved or renamed)

```
uid_db_guid             : ae0da2be-8a27-4e0d-9ecd-06f64efcf24a        <= DB GUID
uid_version             : 0x0000000000000022 (34)                     <= Version number
```

- **A particular version of a file is identified by its Global Version Sequence Number (GVSN)**

```
gsvn_db_guid            : ae0da2be-8a27-4e0d-9ecd-06f64efcf24a        <= DB GUID
gsvn_version            : 0x0000000000000023 (35)                     <= Version number
```

- **When a file is modified the GVSN is incremented**

```
                FILE CREATED ON MEMBER 1

uid_db_guid    : ae0da2be-8a27-4e0d-9ecd-06f64efcf24a
uid_version    : 0x000000000000002c (44)

gsvn_db_guid   : ae0da2be-8a27-4e0d-9ecd-06f64efcf24a
gsvn_version   : 0x000000000000002c (44)
```

```
                FILE MODIFIED ON MEMBER 2

uid_db_guid    : ae0da2be-8a27-4e0d-9ecd-06f64efcf24a
uid_version    : 0x000000000000002c (44)

gsvn_db_guid   : d8f38038-ad91-4d15-9b0b-30feac8d65cf
gsvn_version   : 0x000000000000000f (15)
```

Updates

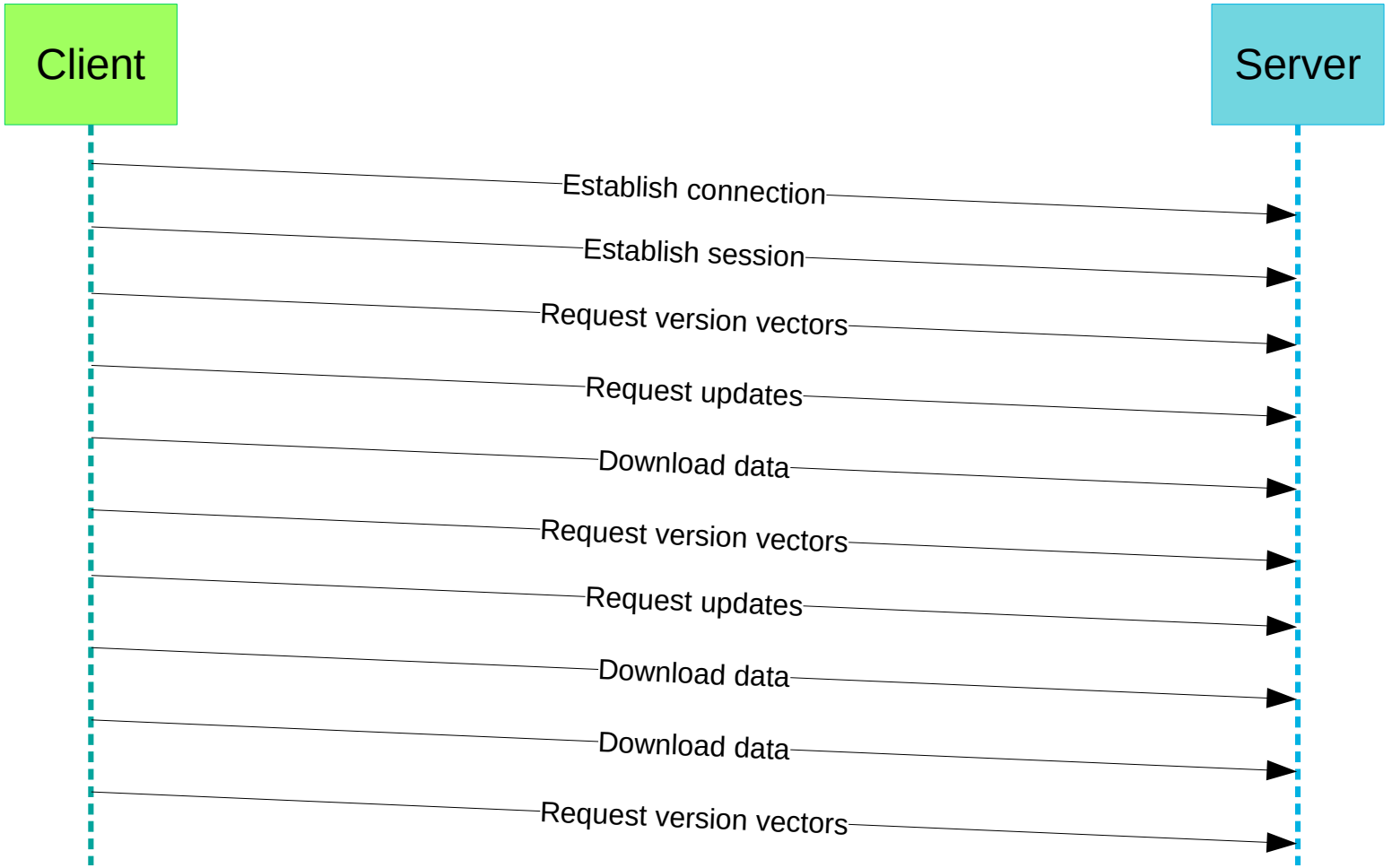
- **Folders are replicated in the same way as files**

```
attributes              : 0x00000020 (32)      <= Files
attributes              : 0x00000010 (16)      <= Folders
```

- **Updates does not contain the file path, but the parent's UID**

```
parent_db_guid          : ae0da2be-8a27-4e0d-9ecd-06f64efcf24a
parent_version          : 0x0000000000000025 (37)
```

Overview


Client
Server
Establish connection
Establish session
Request version vectors
Request updates
Download data
Request version vectors
Request updates
Download data
Download data
Request version vectors

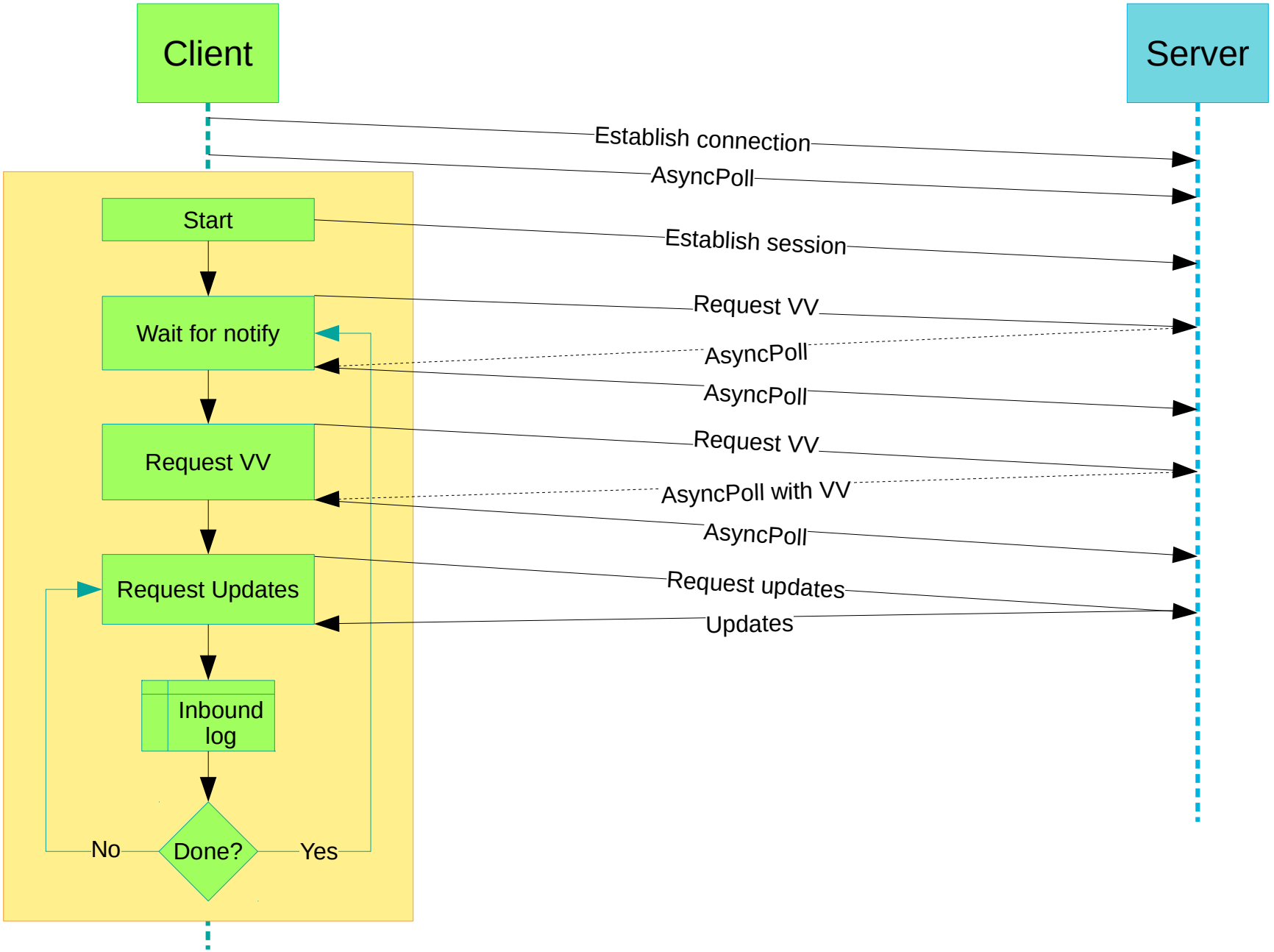
Retrieving updates

- The process of retrieving all the updates has its own state machine
- Retrieved updates are queued to be processed in another loop
- File data download can proceed as an independent process of synchronizing version vectors and updates
- To enable replication across multiple folders, client and server isolate the activities belonging to one folder in a DFS-R session

Asynchronous notifications

- **The client is who drives the protocol**
- **The client requests to be notified when the server's VV changes (AsyncPoll)**
- **The AsyncPoll response carries the VV**
- **Only one pending Async poll per connections, shared among sessions**

Notifications



Processing updates

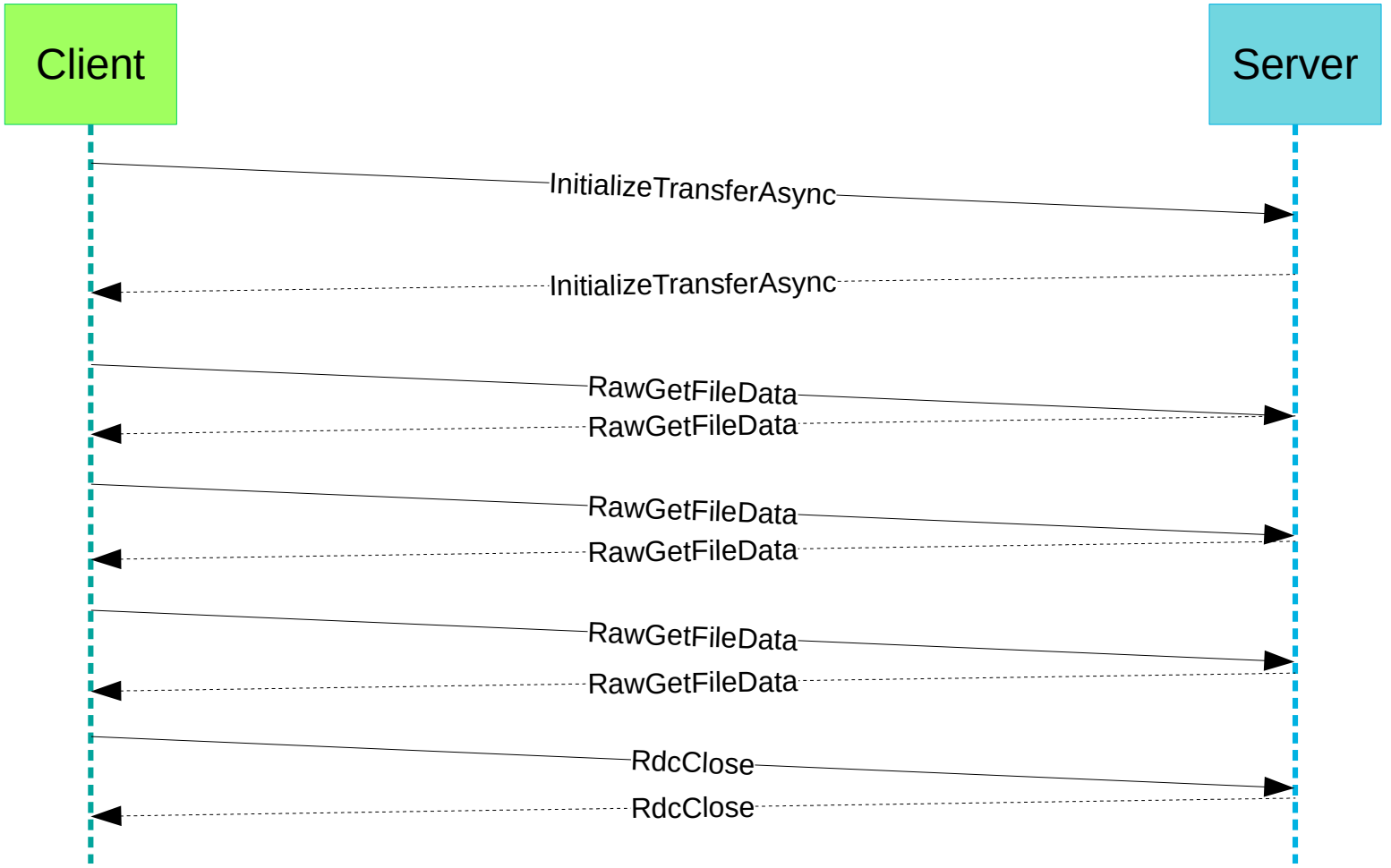
Processing updates

- **Recap**
  1. We got the server's version vector
  2. We computed the delta between server's VV and our known VV
  3. We got the missing updates and queued them
- **Two level queue**
  - Pending VV → Traversed in order
  - Pending updates → Out of order
- **Pick a candidate update**
  - Updates must be installed in ancestral order to prevent conflicts
  - Determine if it is necessary to download the data
  - The downloaded data is staged to a file
  - The update is installed in persistent storage
- **When all updates pertaining to a VV are installed, update the stored VV**

Downloads

- **While processing updates, the client may download file data**
- **Four ways**
  - RdcGetFileData → Require RDC (Remote Differential Compression)
  - RdcGetFileDataAsync → Require RDC and DCE-RPC byte pipes
  - RawGetFileData
  - RawGetFileDataAsync → Require DCE-RPC byte pipes
- **A file starts with an initialization of file transfer**
  - InitializeFileTransferAsync, which carries the first 256KB of data
- **Client request subsequent chunks**
  - RawGetFileData, chunk size 256KB
- **And ends with a close call**
  - RdcClose
- **Downloaded data is staged to a file**

Download data



Installing updates

The meet module

- **The client runs as a Samba4 server service task**
  - Pick update to install and download data to a stage file
- **The staged data must be installed to the final location through the VFS layer**
- **There is a new smbd process, the meet module**
- **The dfsr service and the meet module communicate through IRPC**
- **The meet module needs read access to the DFS-R database to recursively build the target path from parent’s UID.**

Approach

1. Creates a connection to go through VFS layer

2. Handle tombstone updates (file deletions)

3. Uncompress staged data

- XPRESS format (LZ77 + Huffman coding)

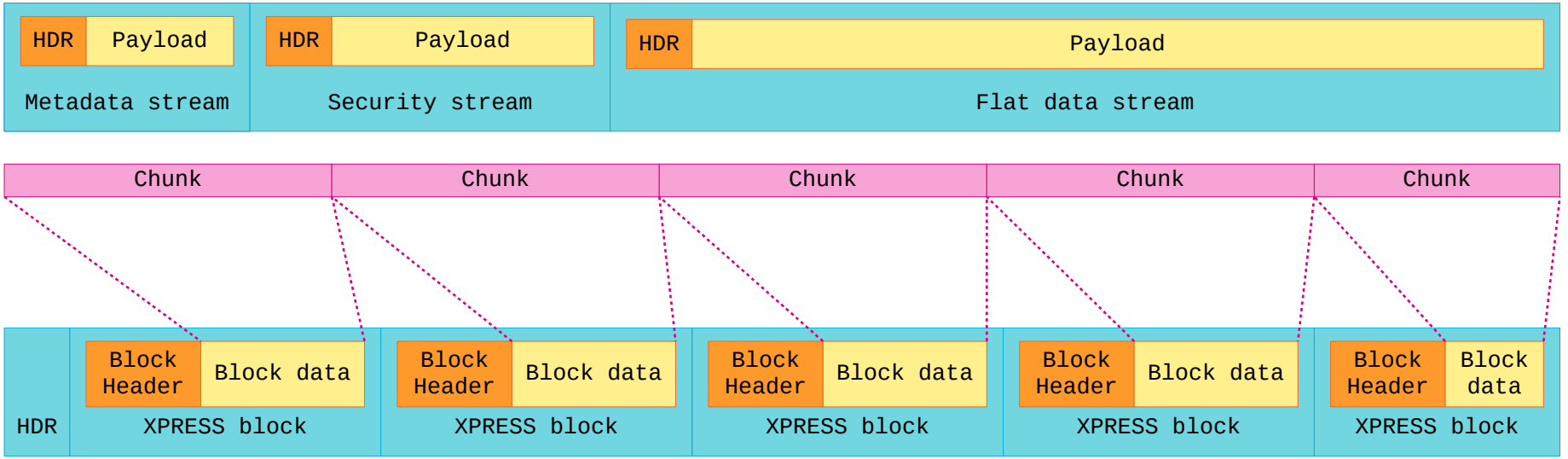
4. Process the uncompressed stream

1. Metadata stream → Create, rename or move the file
2. Security stream → Sets the security descriptor
3. Flat data stram → [MS-BKUP] format. The content.
4. Other stream types (reparse data and compression data not handled yet)

5. Send result to dfsr service

Staged data format

- **Two layers**
  - A sequence of streams
    - Metadata
    - Compression data
    - Reparse data
    - Flat data
    - Security data
  - Encapsulated on a compressed data format, even if uncompressed



Demo

Setup

SambaXP_TestGroup
{8db97569-cfd6-4b13-a99a-b4c6267b07f6}

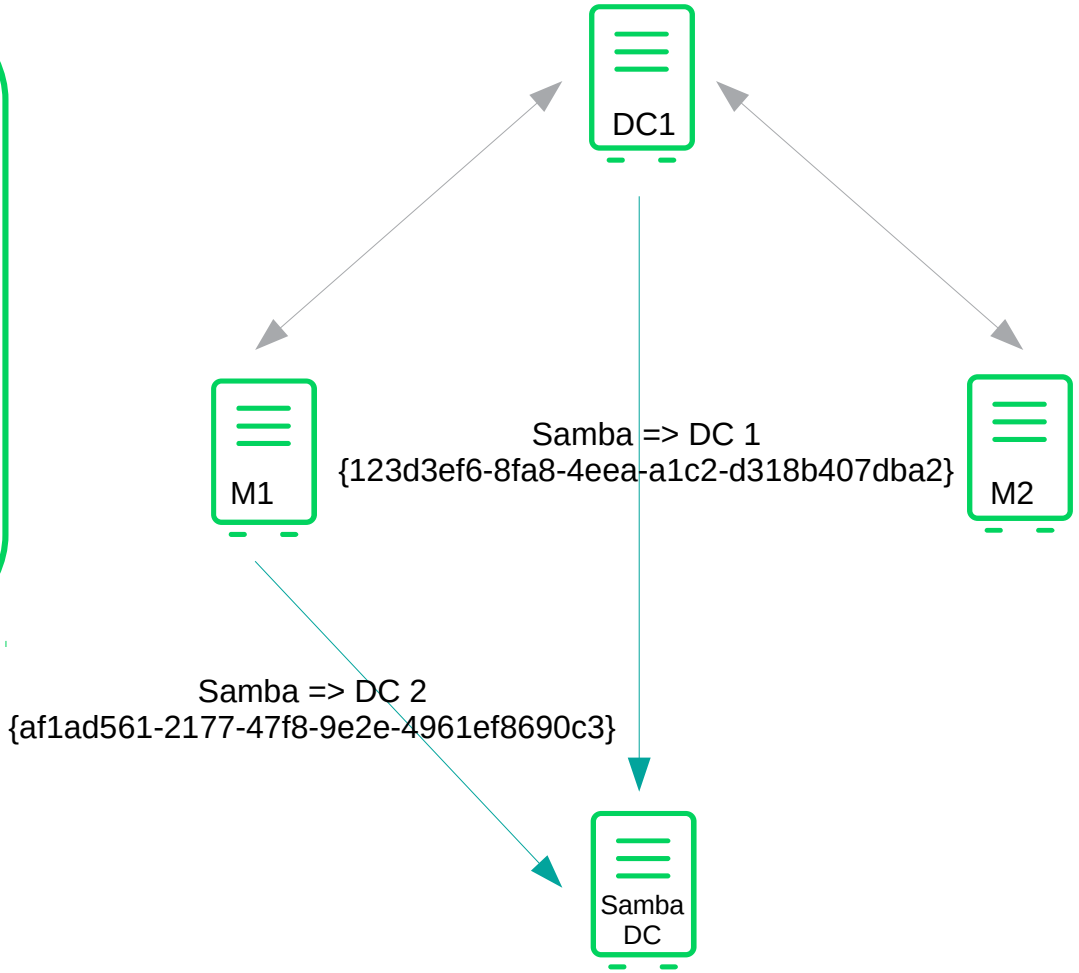
SambaXP_TestFolder1
{9fdb9b8c-f88e-4438-b689-14a06bbe5c1a}

SambaXP_TestFolder2
{18b1586c-232b-4459-98b3-390939c96b8c}

WIN2K12R2-1 DB Guid
{ae0da2be-8a27-4e0d-9ecd-06f64efcf24a}

WIN2K12R2-2 DB Guid
{d8f38038-ad91-4d15-9b0b-30feac8d65cf}

WIN2K12R2-3 DB Guid
{6ff04912-7f6c-4147-a3f9-6231534d919b}



Samba configuration

```
[global]
    netbios name = MONCAYO
    workgroup = SAMBA1
    realm = SAMBA1.AD
    server role = active directory domain controller

    #### DFS-R ####
    server services = +dfsr
    dfsrsrv: sysvol_join = yes
    #### DFS-R ####

    #### New log categories ####
    log level = 2 dfsr:10 dfsr_meet:10
```

Code

- **Available on https://github.com/kernevil/samba/tree/dfs-r**
- **55 patches**
  - 19 are the management tool (samba-tool dfsr)
- **42 files changed, 9939 insertions(+), 10 deletions(-)**

Next steps

Client side

- **Protocol**
  - Slow sync sub-protocol
  - Remote Differential Compression (RDC)?
- **Receiving updates**
  - Credit system to throttle update retrieval / install
- **Processing updates**
  - Verify hashes to skip data download on match
- **Downloading data**
  - DCE-RPC byte pipes
- **Installing updates**
  - Set timestamps from metadata stream
  - Handle compression data stream
  - Handle reparse data stream
  - Handle flat data stream as a MS-BKUP stream
- **Find a way to test the client in selftests without a windows server**

Server side

- **Big uncertainty yet**
- **How to “catch” file closes?**
  - Specially when samba is not “on the path”
  - File system event notifications, like inotify?
  - Kernel support?
- **Force the windows clients to not use RPC byte pipes and RDC**
  - The use of RPC byte pipes can be avoided reporting ourselves as Windows 2003 on the connection response
  - Is RDC mandatory? If not, how to tell the client to not use it?
- **Database backend**
  - TDB?
  - SQLite?


SUSE®
We adapt. You succeed.