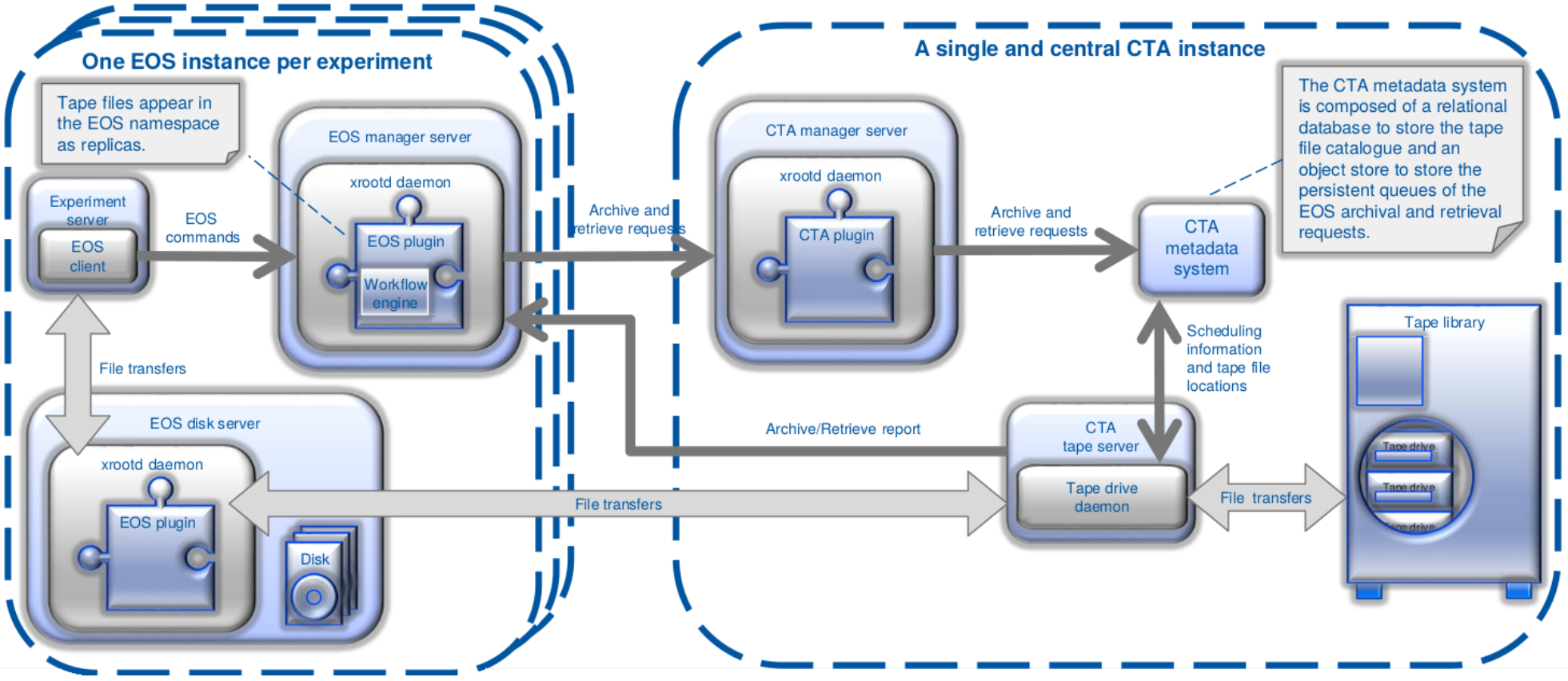
Building Client-Server APIs with the Scalable Service Interface (SSI)

Michael Davis Ph.D.

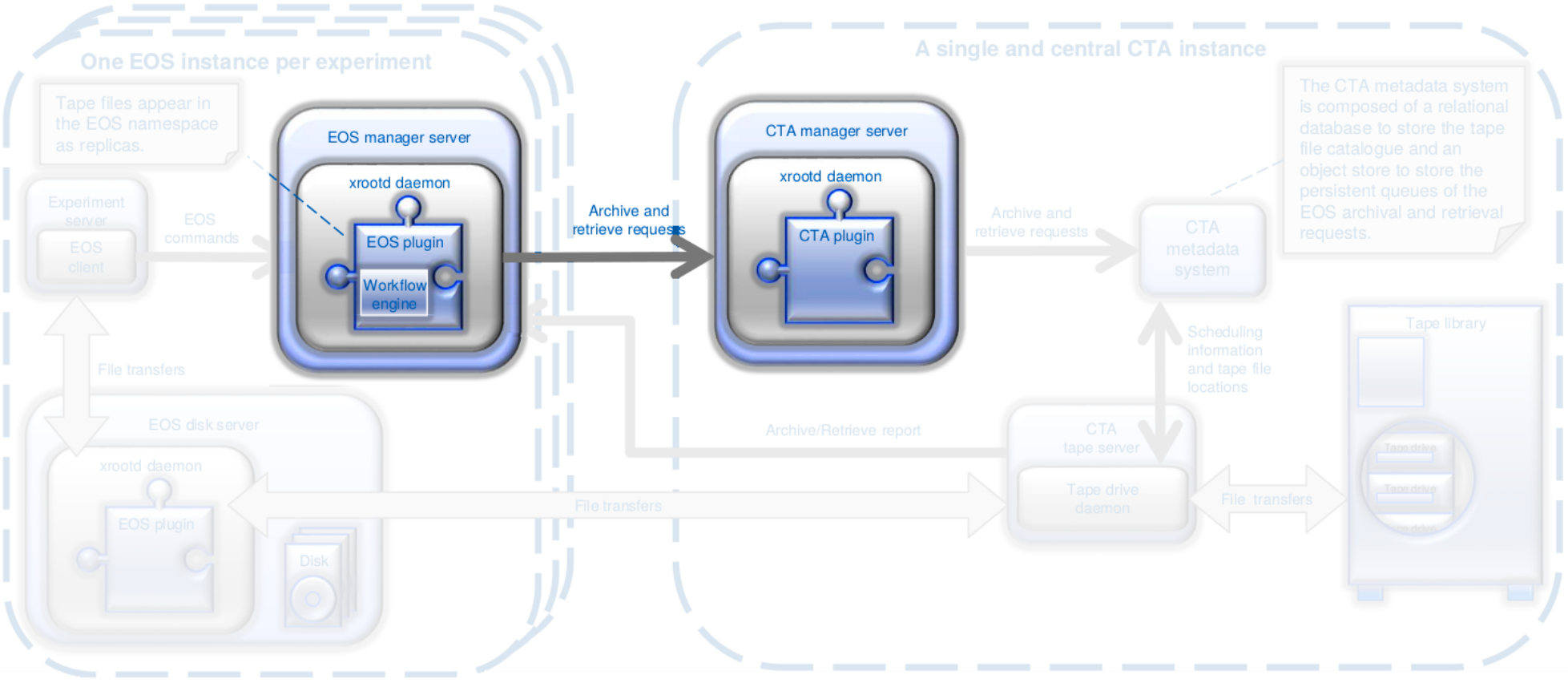
2nd EOS Workshop, CERN, Meyrin

6 February 2018

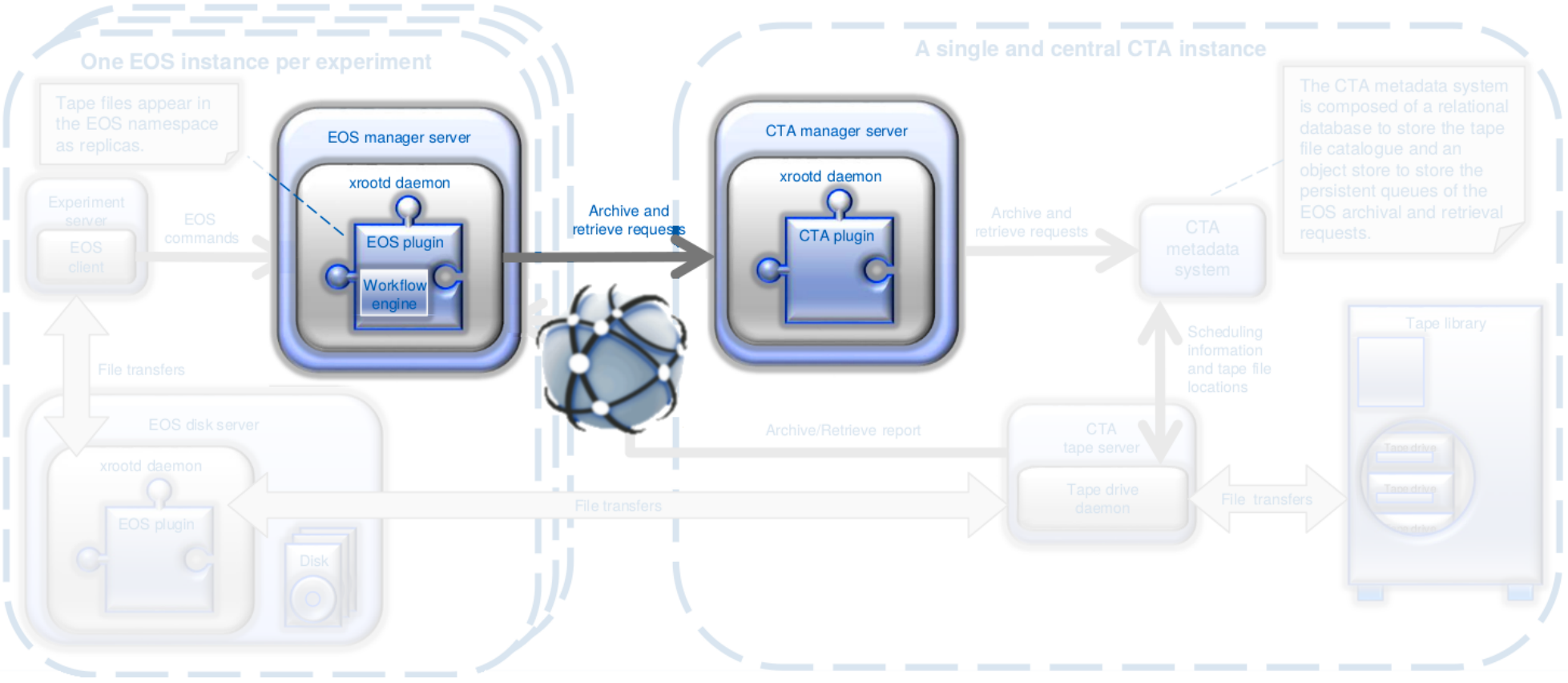
Motivation


One EOS instance per experiment
Tape files appear in the EOS namespace as replicas.
EOS manager server
xrootd daemon
EOS plugin
Workflow engine
Experiment server
EOS client
EOS commands
File transfers
EOS disk server
xrootd daemon
EOS plugin
Disk
Archive and retrieve requests
A single and central CTA instance
CTA manager server
xrootd daemon
CTA plugin
Archive and retrieve requests
CTA metadata system
The CTA metadata system is composed of a relational database to store the tape file catalogue and an object store to store the persistent queues of the EOS archival and retrieval requests.
Scheduling information and tape file locations
Archive/Retrieve report
CTA tape server
Tape drive daemon
File transfers
File transfers
Tape library
Tape drive
Tape drive

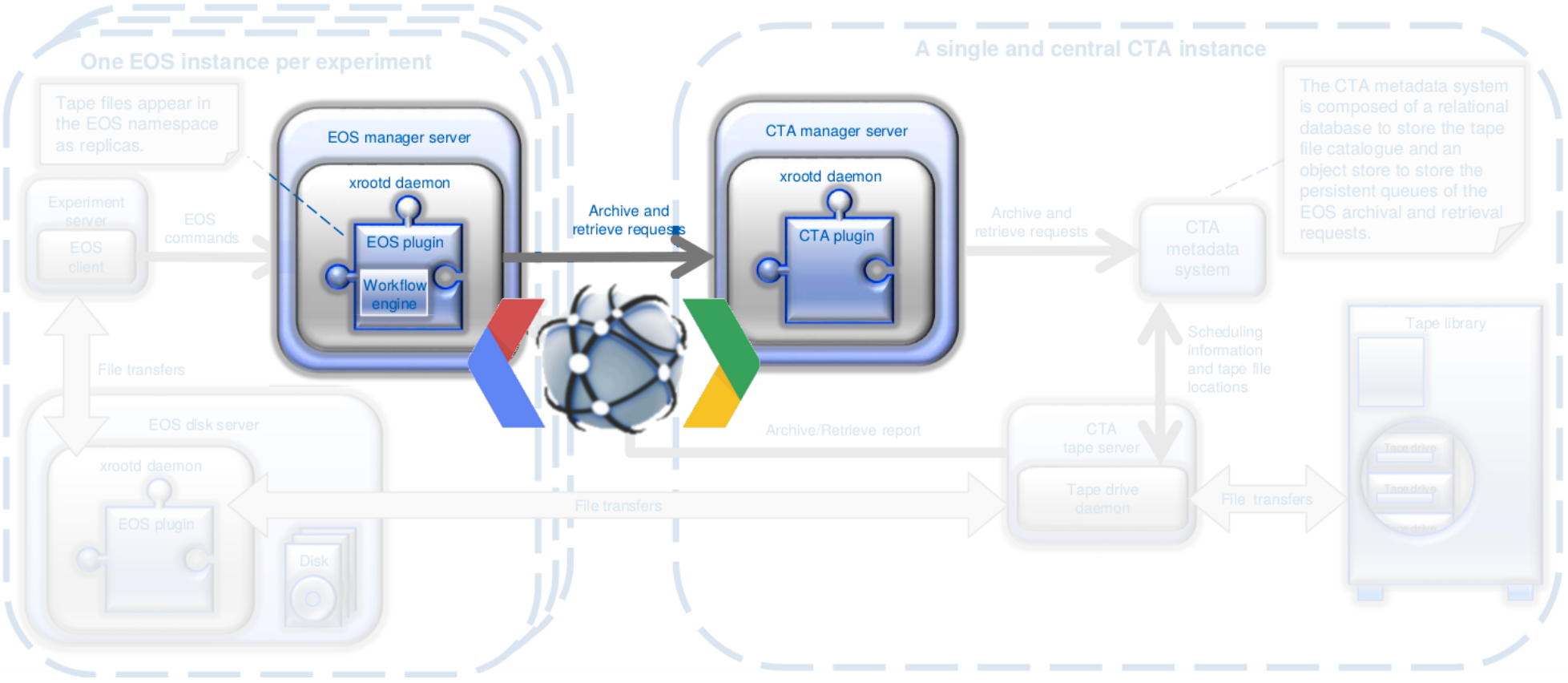
Motivation


One EOS instance per experiment
Tape files appear in the EOS namespace as replicas.
Experiment server
EOS client
EOS commands
EOS manager server
xrootd daemon
EOS plugin
Workflow engine
Archive and retrieve requests
A single and central CTA instance
CTA manager server
xrootd daemon
CTA plugin
Archive and retrieve requests
CTA metadata system
The CTA metadata system is composed of a relational database to store the tape file catalogue and an object store to store the persistent queues of the EOS archival and retrieval requests.
Scheduling information and tape file locations
Tape library
File transfers
EOS disk server
xrootd daemon
EOS plugin
Disk
Archive/Retrieve report
CTA tape server
Tape drive daemon
File transfers
File transfers

Motivation


One EOS instance per experiment
Tape files appear in the EOS namespace as replicas.
Experiment server
EOS client
EOS commands
EOS manager server
xrootd daemon
EOS plugin
Workflow engine
Archive and retrieve requests
CTA manager server
xrootd daemon
CTA plugin
A single and central CTA instance
Archive and retrieve requests
CTA metadata system
The CTA metadata system is composed of a relational database to store the tape file catalogue and an object store to store the persistent queues of the EOS archival and retrieval requests.
Scheduling information and tape file locations
Tape library
File transfers
EOS disk server
xrootd daemon
EOS plugin
Disk
Archive/Retrieve report
CTA tape server
Tape drive daemon
File transfers
File transfers

Motivation


One EOS instance per experiment
Tape files appear in the EOS namespace as replicas.
Experiment server
EOS client
EOS commands
EOS manager server
xrootd daemon
EOS plugin
Workflow engine
Archive and retrieve requests
CTA manager server
xrootd daemon
CTA plugin
Archive and retrieve requests
A single and central CTA instance
CTA metadata system
The CTA metadata system is composed of a relational database to store the tape file catalogue and an object store to store the persistent queues of the EOS archival and retrieval requests.
Scheduling information and tape file locations
Tape library
File transfers
EOS disk server
xrootd daemon
EOS plugin
Disk
Archive/Retrieve report
CTA tape server
Tape drive daemon
File transfers
File transfers

Scalable Service Interface (SSI)

XRootD Scalable Service Interface

- Component of the XRootD framework
  - Implemented as an XRootD plugin
  - Available in version $\geq$ 4.7.0
  - Version $\geq$ 4.8.1 recommended
- Request–response framework
- Asynchronous client-server communication
- Remote Object Execution model
- Harnesses XRootD base features

Google Protocol Buffers

- XML Lite
  - "…a language-neutral, platform-neutral, extensible mechanism for serializing structured data"

```
message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;
}
```

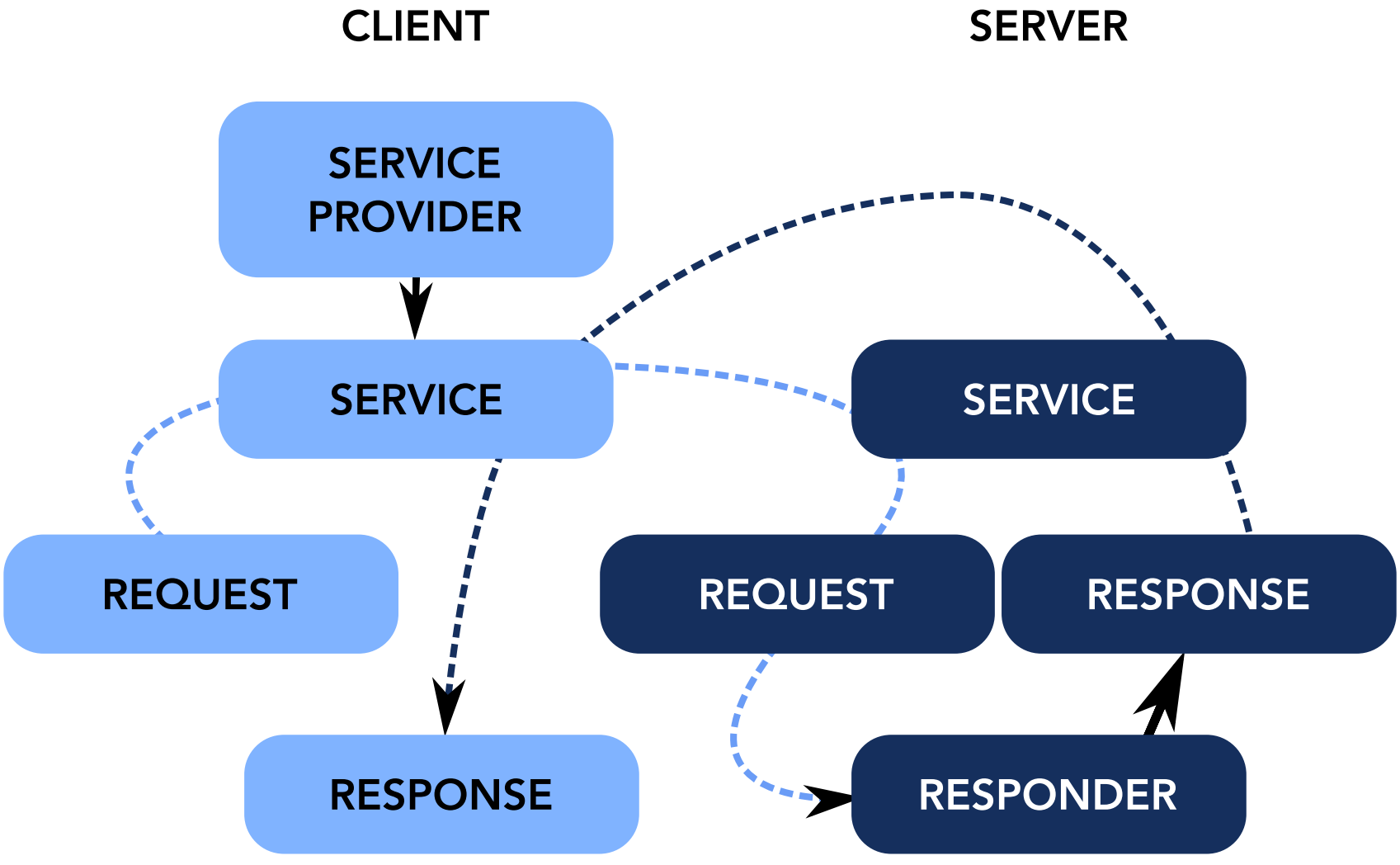
- `protoc` compiler turns the Protocol Buffer definition into C++

Overview

- The XRootD Scalable Service Interface (SSI)
- XrdSsiPb Framework
  - XRootD SSI
  - Google Protocol Buffers 3
- Case Study
  - EOS–CTA Interface

XRootD SSI Concepts

Overview



XRootD SSI Concepts

Key Classes

- XrdSsiServiceProvider (client-side only)
- XrdSsiService
  - Resource `\ctafrontend`
  - Endpoint `host:port`
- XrdSsiRequest
  - Request::ProcessResponse()
  - Request::ProcessResponseData()
  - Detached Requests
- XrdSsiResponder (server-side only)

XRootD SSI Concepts

Response Types

- Metadata Responses
- Data Responses (known length)
- Stream Responses (arbitrary length)
- File Responses
- Error Responses
- Asynchronous Alerts

XRootD SSI Concepts

Design Considerations

- Minimize data copying
  - Callbacks
  - Pointers to buffers
  - Creating and destroying objects
- Asynchronous Threads
  - Requests and Responses may be handled by different threads, but not necessarily
  - In some cases, SSI methods perform an immediate callback using the calling thread

XRootD SSI Concepts

Security : Authentication and Identification

- Uses standard XRootD authentication suite
  - UNIX
  - KRB5
  - SSS
  - ...
- Clients use the authentication method(s) specified by `XrdSecPROTOCOL`
- Identification using SSS keys:

```
0 u:username g:groupname n:keyname N:45230...
```

XRootD SSI Concepts

Server Clustering

- Builds on clustering features of the base XRootD framework
- Multiple servers clustered by a redirector or a manager node
- Each server in the cluster:
  - A given resource can be present, or not
  - The server can be active or suspended
- Resource Affinity
  - Manage load balancing of requests across the active servers that provide a requested resource

XrdSsiPb Framework

XRootD + SSI + Google Protocol Buffers 3

- Generic binding between:
  - User-defined Protocol Buffer definitions
  - Google Protocol Buffer 3 implementation
  - XRootD SSI framework
- Abstraction layer

XrdSsiPb Framework

XRootD + SSI + Google Protocol Buffers 3

- Implements a subset of XRootD SSI
  - Metadata-only responses
  - Data responses
  - Stream responses (record-based)
- Not implemented (yet)
  - File responses
  - Detached requests
  - Clustering

XrdSsiPb Framework
Synchronization

- XRootD SSI is asynchronous
- XrdSsiPb implements synchronous calls using promises and futures
  - **Metadata Promise** fulfilled when metadata (or error) is received
  - **Data Promise** fulfilled when all data has been received from a data or stream response
- Callback every time a data record is received

XrdSsiPb Framework
Error Handling

- XRootD uses error codes, Protobuf uses C++ exceptions
- XrdSsiPb converts XRootD errors into exceptions
  - Consistent error-handling mechanism
  - The server catches exceptions and passes them back to the client as metadata
  - The client unpacks metadata messages and throws exceptions to the client application
- Asynchronous Alerts also available

Example Client/Server

Protocol Buffer definitions

```
message Request {
  enum CommandType {
    SEND_METADATA = 0;
    SEND_DATA     = 1;
    SEND_STREAM   = 2;
  }
  CommandType cmd = 1;
  int32 repeat    = 2;
  Data record     = 3;
}
```

Example Client/Server

Protocol Buffer definitions

```
message Response {
  enum ResponseType {
    RSP_INVALID       = 0;
    RSP_SUCCESS       = 1;
    RSP_ERR_PROTOBUF  = 2;
    RSP_ERR_SERVER    = 3;
    RSP_ERR_USER      = 4;
  }
  ResponseType type   = 1;
  string message_txt  = 2;
}
```

Example Client/Server

Protocol Buffer definitions

```
// Payload for a Data response,
// or a single record in a Stream response

message Data {
  double      test_double = 1;
  int64       test_int64  = 2;
  bool        test_bool   = 3;
  string      test_string = 4;
}
```

Example Client/Server

Protocol Buffer definitions

```
message Alert {
  string message_txt = 1;
}
```

Example Client/Server

Client/Server API header

```
#include "XrdSsiPbServiceClientSide.hpp"
#include "test.pb.h"

typedef XrdSsiPb::ServiceClientSide<test::Request,
                                    test::Response,
                                    test::Data,
                                    test::Alert>
        XrdSsiPbServiceType;

typedef XrdSsiPb::Request<test::Request,
                          test::Response,
                          test::Data,
                          test::Alert>
        XrdSsiPbRequestType;
```

Example Client/Server

Client code

```
// Instantiate the Service
XrdSsiPbServiceType test_service(endpoint, resource);

// Send the Request and get a Response
test_service.Send(request, response);

switch(response.type())
{
    case Response::RSP_SUCCESS:       // do something
    case Response::RSP_ERR_PROTOBUF: // throw exception
    case Response::RSP_ERR_USER:
    case Response::RSP_ERR_SERVER:
}
```

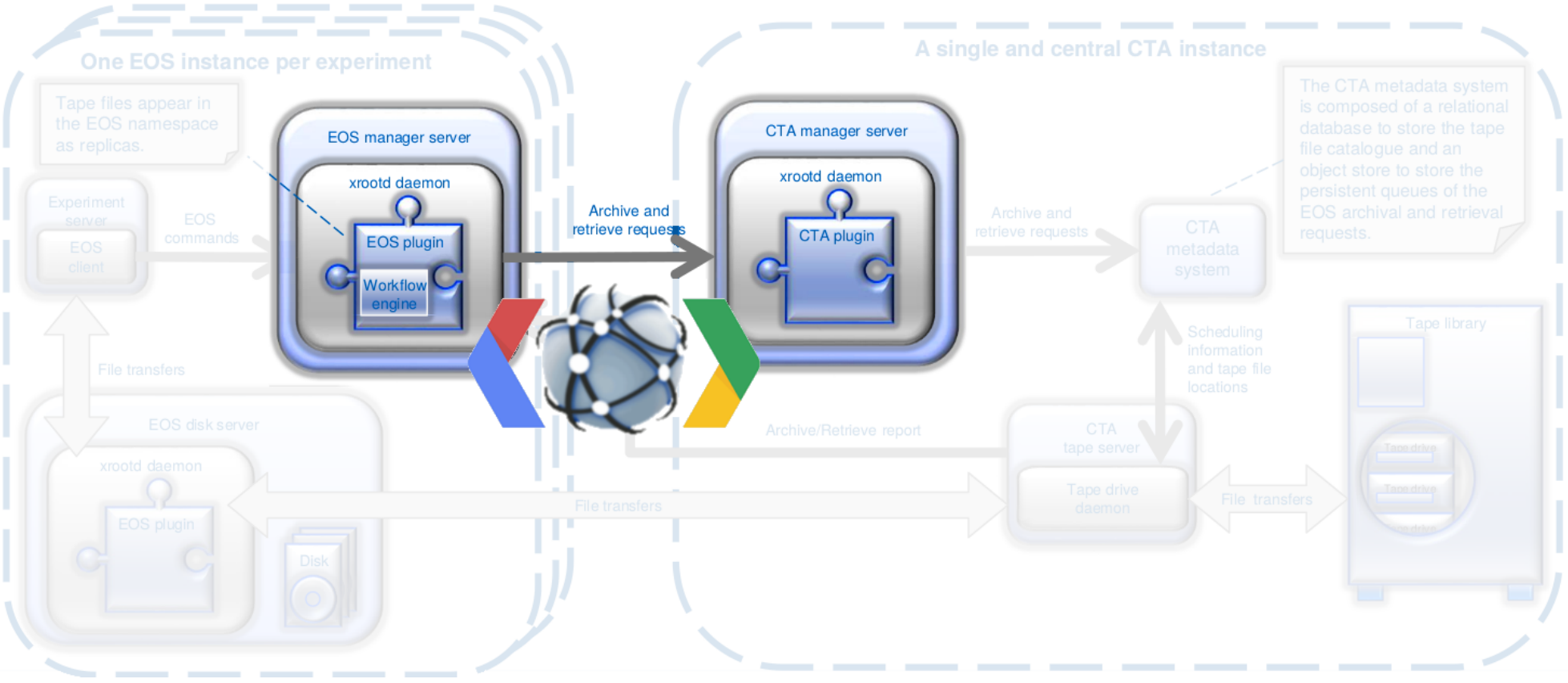
Adding XrdSsiPb to your Project

Git submodule in the EOS project

```
$ git submodule add https://:@gitlab.cern.ch:8443/eos/
    xrootd-ssi-protobuf-interface.git

Cloning into '.../xrootd-ssi-protobuf-interface'...
```

EOS-CTA Interface



EOS WorkFlow Engine

- `CLOSEW`
  - An EOS file has been written to disk
  - Trigger **archive** workflow
- `PREPARE`
  - A file in the EOS namespace has been requested
  - No copy exists on disk
  - Trigger **retrieve** workflow
- `DELETE`
  - An EOS file has been deleted from disk
  - Trigger **delete** workflow

Protocol Buffer Definitions

Request

```
message Workflow {
  enum EventType {
    CLOSEW  = 1;
    DELETE  = 2;
    PREPARE = 3;
  }
  EventType event = 1;
  ...
}

message Notification {
  Workflow wf        = 1;
  Metadata file      = 2;
  Metadata directory = 3;
  ...
}
```

Protocol Buffer Definitions

Response

```
message Response {
  enum ResponseType {
    RSP_INVALID       = 0;
    RSP_SUCCESS       = 1;
    RSP_ERR_PROTOBUF  = 2;
    RSP_ERR_CTA       = 3;
    RSP_ERR_USER      = 4;
  }
  ResponseType type   = 1;
  string message_txt  = 2;
}
```

- Interface between EOS and CTA Front End only requires metadata responses

Enabling the new WorkFlows

- Thanks to **Jozsef Makai** for client implementation in the EOS MGM !
- WorkFlows can be enabled for a directory using EOS extended attributes

```
$ eos attr ls /eos/users/test
CTA_StorageClass="single"
sys.workflow.closew.default=
  "proto/cta:localhost:10955 <parent/file>"
sys.workflow.sync::prepare.default=...
sys.workflow.sync::delete.default=...
...
```

Summary

- XRootD SSI provides an efficient Request-Response framework
  - Uses standard XRootD features for authentication, clustering, etc.
- XrdSsiPb is a generic binding for XRootD SSI + Protocol Buffers 3
  - Suitable for use cases where Requests and Responses can be specified as Protocol Buffers
  - EOS–CTA interface is implemented as EOS WorkFlow Engine (client) + XrdSsiPb + CTA Front End (server)