ESCUELA SUPERIOR DE INGENIERÍA

GRADO EN INGENIERÍA INFORMÁTICA

Last Mile Transfer: enabling local data transfers on the global WLCG infrastructure

Owayss Kabtoul

ESCUELA SUPERIOR DE INGENIERÍA

GRADO EN INGENIERÍA EN INFORMÁTICA

Last Mile Transfer: enabling local data transfers on the global WLCG infrastructure

- Departamento: Ingeniería Informática
- Director del proyecto: Manuel Palomo Duarte
- Co-director del proyecto: Alejandro Álvarez Ayllón
- Autor del proyecto: Owayss Kabtoul

Puerto real a 2 de noviembre de 2017
Fdo: Owayss Kabtoul

Agradecimientos

El 19 de noviembre de 1957, tras haber ganado el Premio Nobel de literatura, escribió Albert Camus la siguiente carta a su maestro de colegio:

> *Querido señor Germain:*
>
> *Esperé a que se apagara un poco el ruido de todos estos días antes de hablarle de todo corazón. He recibido un honor demasiado grande, que no he buscado ni pedido. Pero cuando supe la noticia, pensé primero en mi madre y después en usted. Sin usted, sin la mano afectuosa que tendió al niño pobre que era yo, sin su enseñanza no hubiese sucedido nada de esto. No es que dé demasiada importancia a un honor de este tipo. Pero ofrece por lo menos la oportunidad de decirle lo que usted ha sido y sigue siendo para mí, y de corroborarle que sus esfuerzos, su trabajo y el corazón generoso que usted puso en ello continúan siempre vivos en uno de sus pequeños escolares, que, pese a los años, no ha dejado de ser un alumno agradecido. Un abrazo con todas mis fuerzas.*
>
> *Albert Camus.*

Si bien el presente trabajo dista de ser similar a las obras del autor francés, el mentor que he tenido no es nada menos de esa descripción.

A Manuel Palomo Duarte.

License

Abstract

The computing challenge at CERN is of a global nature. To make real world-wide distributed computing possible, more than 150 computer centers must be seamlessly integrated. This means integrating CPU, storage and network.
The File Transfer Service (FTS) is a tool that emerges to solve the data movement problem. It is used to schedule data transfers between different storage resources. Its optimizer takes care of increasing the parallelism to improve throughput, without exhausting the storage resources. It also has a web interface (WebFTS) which makes it quite easy for users to invoke reliable, managed data transfers on distributed infrastructure.
However, FTS only solves part of the problem, as an increasing number of grid users run simulations on their personal laptops, generating files that can amount to several gigabytes. Normally, users would want to move these files from their personal computers to a remote Grid storage for long-term archiving, sharing, or running further processing on them. The issue here is that these users might be sitting behind a firewall, which means that their computer will not be able to listen to inbound connections.
Last Mile Transfer is a solution that was developed to enable local file uploads on the Worldwide LHC Computing Grid (WLCG) infrastructure.

Keywords

Storage, Network, Distributed Systems, FTS, CERN, LHC, Proxy, Middleware, Go, Web.

Contents

List of Figures

List of Tables

List of Code

Chapter 1

Project Context

The present work is the outcome of a seven months long collaboration project between two institutions: the European Organization for Nuclear Research (CERN), and the University of Cádiz (UCA).
The project was named Last Mile Transfer (we will conveniently call it LMT throughout this work), and it was developed as a solution to enable Grid users to upload data from their local storage devices to a remote storage on the Worldwide LHC Computing Grid, be it to perform further computation on the data or for archiving purposes.
The work was presented at the IT Storage group monthly meeting on the 6th of October of this current year, 2017. The presentation included a live demo of the developed service and it was well received.
This first chapter introduces the problem context. In particular, it describes the aspects that are most relevant to the Last Mile Transfer project.

1.1 CERN

The name CERN is derived from the acronym for the French "Conseil Européen pour la Recherche Nucléaire", or European Council for Nuclear Research, a provisional body founded in 1952 with the mandate of establishing a world-class fundamental physics research organization in Europe. At that time, pure physics research concentrated on understanding the inside of the atom, hence the word "nuclear". [2]

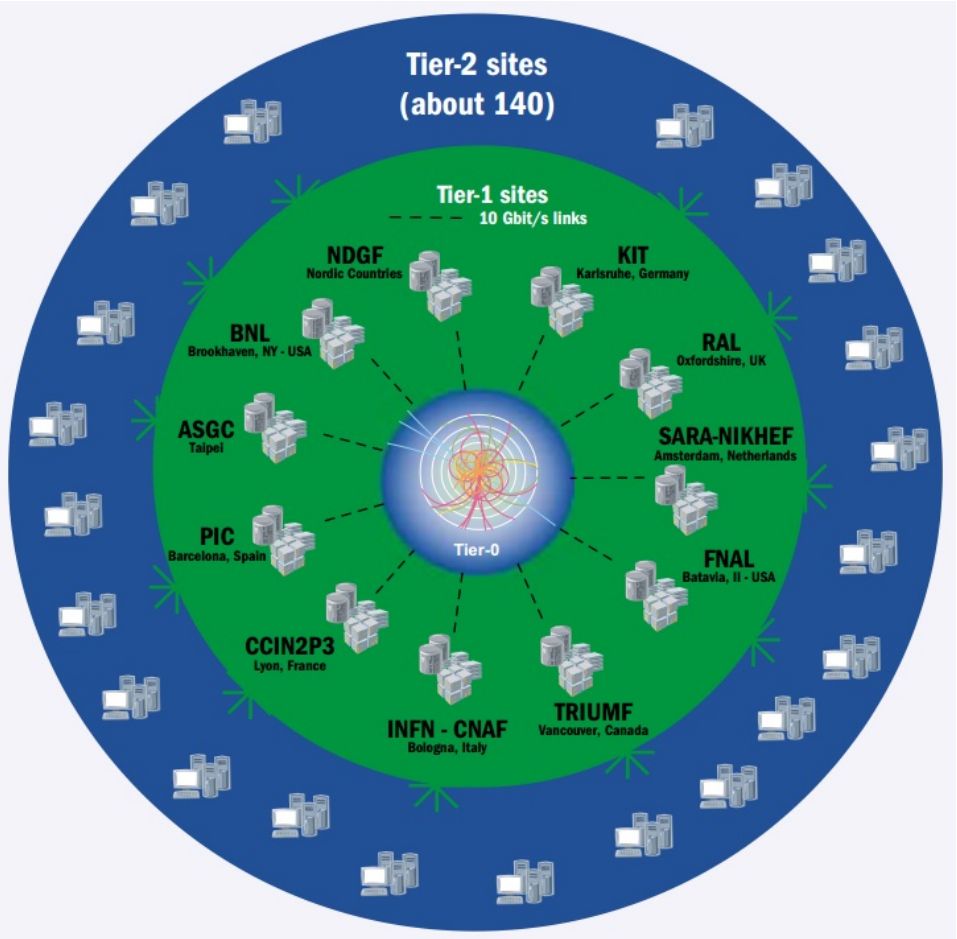


Figure 1.1: The Worldwide LHC Computing Grid

1.2 WLCG infrastructure

The Worldwide LHC Computing Grid (WLCG) project is a global collaboration of more than 170 computing centres in 42 countries, linking up national and international grid infrastructures.
The mission of the WLCG project is to provide global computing resources to store, distribute and analyse the $\sim$ 50 Petabytes of data expected in 2017, generated by the Large Hadron Collider (LHC) at CERN on the Franco-Swiss border. [3]

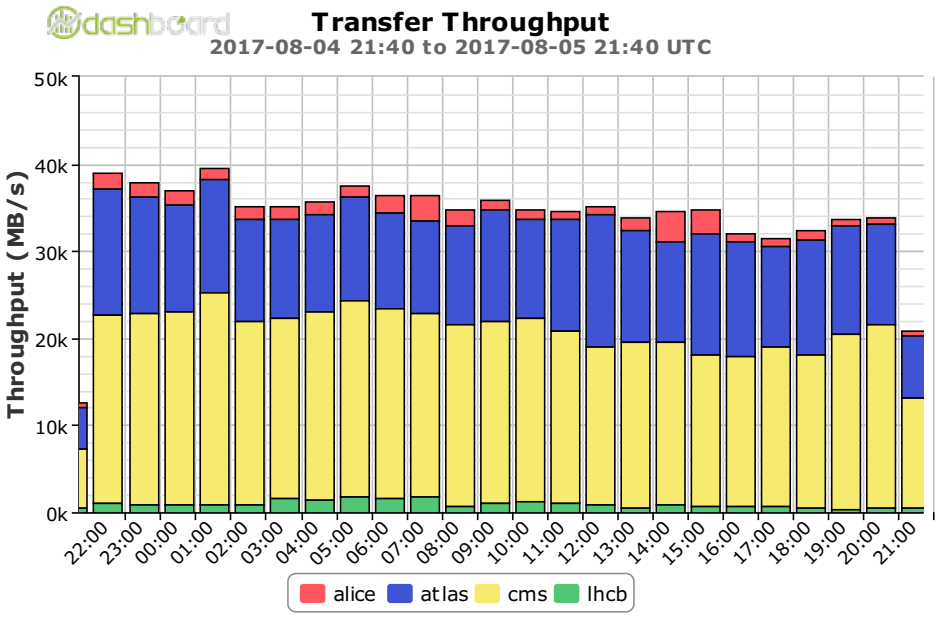


Figure 1.2: WLCG data transfer throughput

1.3 Glossary

1.3.1 Acronyms

API: Application Programming Interface.
CD: Continuous Delivery.
CERN: European Organization for Nuclear Research.
CI: Continuous Integration.
DAV: Distributed Authoring and Versioning.
DN: Distinguished Name.
FTS: File Transfer Service.
LAN: Local Area Network.
LHC: Large Hadron Collider.
LMT: Last Mile Transfer.
REST: Representational State Transfer.
SE: Storage Element.
TLS: Transport Layer Security.
VOMS: Virtual Organization Membership Service.
W3C: The World Wide Web Consortium.
WAN: Wide Area Network.
WLCG: Worldwide LHC Computing Grid.

1.3.2 Definitions

CA Certificate

Identifies the certification authority (CA) that issues server and client authentication certificates to the servers and clients that request these certificates. Because it contains a public key used in digital signatures, it is also referred to as a signature certificate. If the CA is a root authority, the CA certificate may be referred to as a root certificate. Also sometimes known as a site certificate.

Certification Authority

An entity entrusted to issue certificates that assert that the recipient individual, computer, or organization requesting the certificate fulfills the conditions of an established policy

GridFTP

GridFTP is an extension of the File Transfer Protocol (FTP) for grid computing. The protocol was defined within the GridFTP working group of the Open Grid Forum. [4]

GSIFTP

GSIFTP is a subset of the GridFTP protocol. It is essentially standard FTP enhanced to use GSI security. It does not includes many of the high-performance GridFTP protocol features, such as parallel data transfer, automatic TCP window/buffer sizing, and enhanced reliability.

X.509

An ITU-T (Telecommunication Standardization Sector) standard for a PKI (Public Key Infrastructure) and PMI (Privilege Management Infrastructure). X.509 specifies, amongst other things, standard formats for public key certificates, certificate revocation lists, attribute certificates, and a certification path validation algorithm. [5]

X.509 Proxy Certificate

The term Proxy Certificate is used to describe a certificate that is derived from, and signed by, a normal X.509 Public Key End Entity Certificate or by another Proxy Certificate for the purpose of providing restricted proxying and delegation within a PKI (Public Key Infrastructure) based authentication system. [5]

Chapter 2

State of the Art

Having introduced the context of this project and briefly mentioned the problem we are trying to solve, we shall now discuss the current state of the File Transfer Service (FTS), its web interface (WebFTS), and then we will proceed to thoroughly discuss the problem of local data transfers on the Worldwide LHC Computing Grid.

2.1 File Transfer Service

FTS3 (File Transfer Service) is the service responsible for globally distributing the majority of the LHC data across the WLCG infrastructure. It is a low level data movement service, responsible for reliable bulk transfer of files from one site to another while allowing participating sites to control the network resource usage.

2.1.1 Overview

FTS3 is a bulk data mover, created to distribute globally the multiple petabytes of data from the LHC at CERN.
Its purpose is to efficiently schedule data transfers, maximizing the use of available network and storage resources while ensuring that any policy limits are respected.
FTS3 can be used in a number of different ways: [6]

- An individual or small team can access the web interface to FTS to schedule transfers between storage systems. They can browse the contents of the storage, invoke and manage transfers, and leave FTS to do the rest.

- A team's data manager can use the FTS command line interface or the REST API to schedule bulk transfers between storage systems.

- A data manager can install an FTS service for local users. The service is equipped with advanced monitoring and debugging capabilities which enable her to give support to her users.

- Maintainers of frameworks which provide higher level functionality can delegate responsibility for transfer management to FTS by integrating it using the various programatic interfaces available, including a REST API. The users thus continue to use a familiar interface while profiting from the power of FTS transfer management.

2.2 WebFTS

WebFTS is a file transfer and management solution which allows users to invoke reliable, managed data transfers on distributed infrastructures. [7]
Created following simplicity and efficiency criteria, WebFTS allows the user to access and interact with multiple storage elements. Their content becomes browsable and different filters can be applied to get a set of files to be transferred. Transfers can be invoked and capabilities are provided for checking the detailed status of the different transfers and resubmitting any of them with only one click.
The "transfer engine" used is FTS3, the service responsible for distributing the majority of LHC data across WLCG infrastructure. This provides WebFTS with reliable, multi-protocol (gridftp, srm, http, xrootd), adaptively optimised data transfers.
An architectural overview of WebFTS is shown in figure 2.1

2.3 The Problem

Grid users sometime run simulations on their local laptops, and generate files that can account to several gigabytes. Normally, users want to move these files from their laptop to a remote Grid storage, for long term archiving, for sharing, or running further processing on them.
Given that these users may be behind a firewall, using an unreliable connection, or just carrying around their laptop connecting and disconnecting intermittently, uploading these files can imply several manual attempts, which is less than ideal.
Also, it has to be considered that these remote storages are often not comparable to their cloud alternatives, as Dropbox or Google Drive. Meaning, the resources on the server side are limited, and the experience of uploading these files can suffer if a high number of users try to upload their files at the same time (e.g. when a conference is approaching).
They will eventually succeed, but again, manually re-submitting the transfer job themselves.

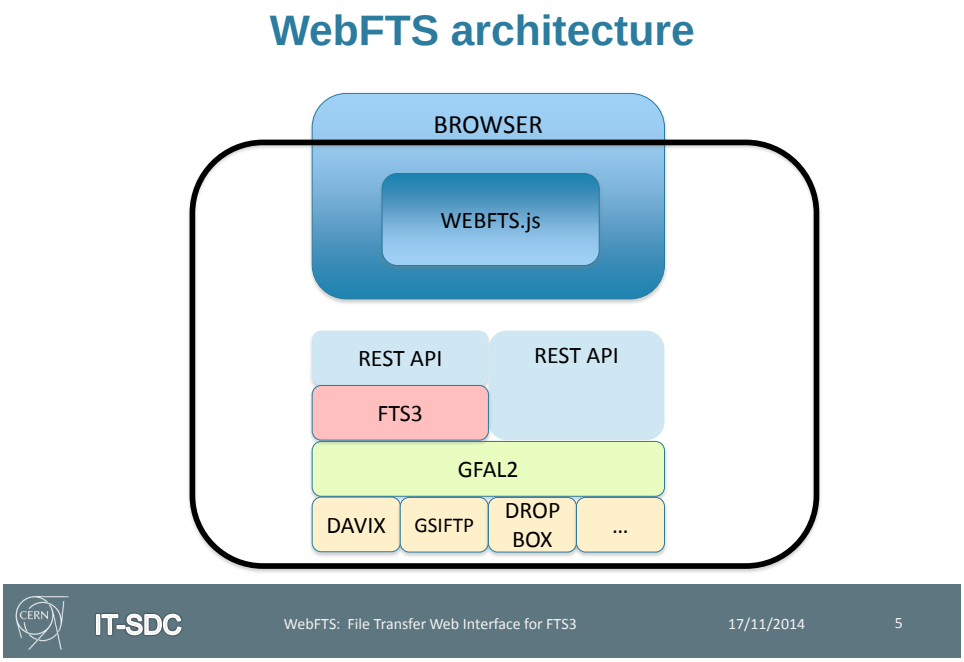


Figure 2.1: WebFTS Architecture - IT-SDC presentation

2.4 Why is FTS not capable of doing local uploads?

As we have discussed in the introductory chapter, FTS currently cannot do local uploads. Mainly, this is due to one issue: the firewall.

2.4.1 The firewall

Transfer done by the File Transfer Service (FTS) need what is known as server-to-server communications, meaning that the service (FTS) needs a server on the origin's end (the user's personal computer) to contact. But even if we were to implement such a client and install it on the user's laptop, the firewall is still an issue.
Normally, clients laptops will not be able to listen to inbound connections from the File Transfer Service (FTS), as the firewall blocks all incoming connections from the Wide Area Network (WAN). This is best illustrated in figure 2.2.

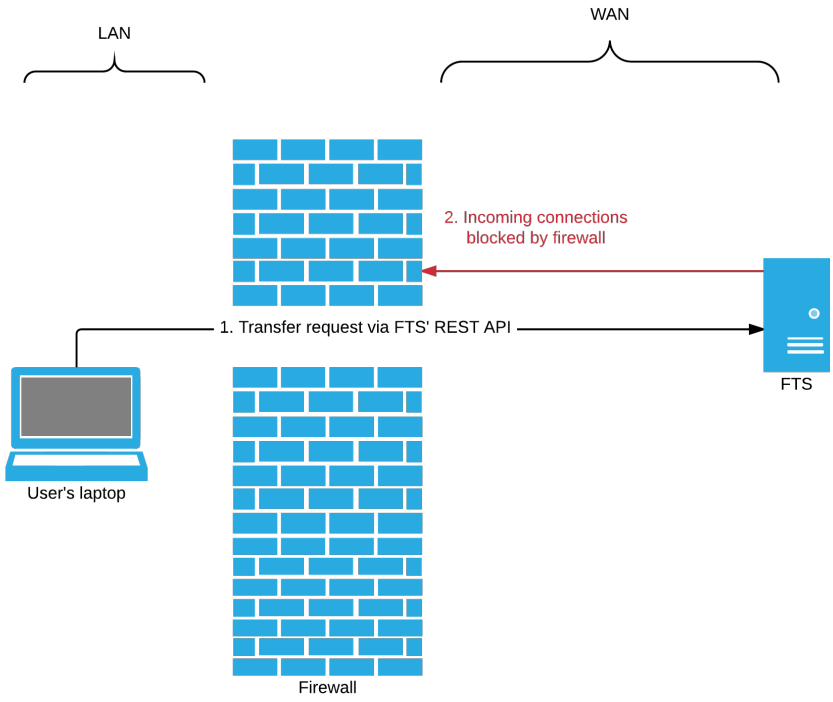


Figure 2.2: FTS' firewall issue

Chapter 3

Solution

This chapter discusses the technical components of the Last Mile Transfer solution. Section 3.2 briefly outlines how the LMT service fits into the whole WLCG puzzle, while section 3.3 introduces each component separately and explains how it is being used in the solution.

3.1 Proposed solution

The solution involves two parts:

- An on-demand proxy service that has been deployed on a CERN machine.
- An extension to WebFTS to support local file uploads through the browser and communicate with the LMT proxy service.

3.1.1 An on-demand proxy service

The main part of the Last Mile Transfer solution is the on-demand proxy service.
The fundamental idea here is to have the proxy service deployed on a CERN machine, and have it listen to incoming connection from WebFTS.
The proxy would register transfers for authenticated clients through a JSON-based communication protocol with WebFTS, and will be in charge of streaming the data from client (that is, the Grid user that submits a transfer job via the WebFTS interface) to the File Transfer Service (FTS).

3.1.2 Extending WebFTS

In order to implement this solution, work needs to be done on the WebFTS project. [7]
In particular, we need to extend WebFTS by providing a JavaScript library to interact with the proxy service. The idea here is that WebFTS will open

up a WebSocket connection to the proxy service, and using our own communication protocol (described later in section 3.3.2) it would communicate control messages over the WebSocket connection, and stream the file contents once the service asks for them.

3.2 How LMT works

On one end, LMT proxy service listens to requests from WebFTS at:

`wss://hostname:port/socket`

When a client WebFTS connects to LMT via a WSS (WebSocket Secure) connection, it sends the metadata for all the files it wishes to transfer via FTS. For each file, LMT would then create an endpoint of the form:

`/transfer/delegationID/filename`

and map it to that particular client. It then informs the client (WebFTS) of the endpoint it created for each file via a JSON-based protocol message sent over the same WebSocket connection.
A transfer message might look something like the following:

Listing 3.1: Transfer message

```
{
    "action":"transfer",
    "data":"https://lmt.cern.ch:8080/transfer/4d7dfd5d
        -f67a-461b-bc4e-20bf4a24c638/someFile.tar.gz"
}
```

The proxy maintains the WebSocket connection open, waiting for the File Transfer Service (FTS) to ask for the files.
WebFTS would receive those endpoint URLs, and proceed to submit a transfer job via FTS' REST API, with the source being the endpoint URLs it received from the proxy service.
On the other end, LMT will also be listening to incoming TCP connections at:

`https://hostname:port/transfer`

Depending on what type of storage system the destination endpoint has, one of two things will happen:

- `gsiftp` endpoint: FTS will be in charge of getting the data from the source, and streaming it to the destination.
- `davs` or `dCache` endpoint: the destination will initiate a 3rd party pull request.

In both cases, the source for the transfer request will be an endpoint belonging to LMT, that is:

`https://lmt.cern.ch:8080/transfer/4d7dfd5d-f67a-461b-bc4e-20bf4a24c638/someFile.tar.gz`
When the LMT proxy receives a request asking for the files, it checks if:

1. The origin which sent the `HTTP` request has permissions to access the file. That is, if the identity of the `X509` delegation certificate the request has is the same as the one the client WebFTS had when it registered the files to be transferred.

2. The files exist, and the client has not closed the WebSocket connection.

If those conditions are met, LMT would then tell WebFTS to start streaming the files via the open WebSocket connection, and it pipes the files contents to the response body of the `GET` request it received.

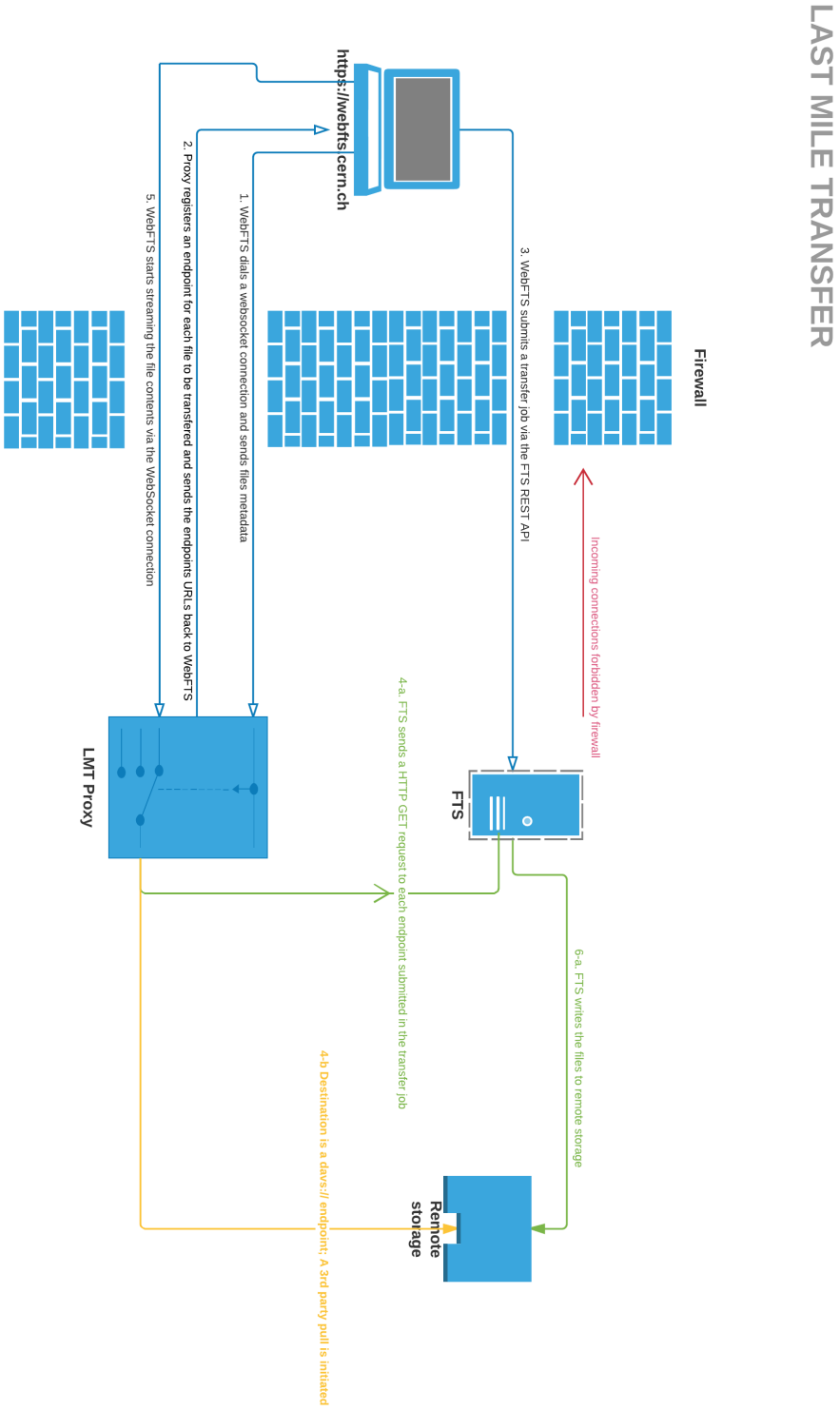


Figure 3.1: Last Mile Transfer - Architecture

3.3 Architecture overview

In this section we will explore the different protocols/technologies that make up the Last Mile Trasnfer (LMT) solution.

3.3.1 WebSockets

The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code. The WebSocket protocol was standardized by the IETF as RFC 6455 [8] in 2011, and the WebSocket API in Web IDL is being standardized by the W3C.
The security model used for this is the origin-based security model commonly used by web browsers. The protocol consists of an opening handshake followed by basic message framing, layered over TCP. The goal of this technology is to provide a mechanism for browser-based applications that need two-way communication with servers that does not rely on opening multiple HTTP connections (e.g., using XMLHttpRequest or <iframe>s and long polling). [8]

The WebSocket protocol was designed to work well with the existing Web infrastructure. As part of this design principle, the protocol specification defines that the WebSocket connection starts its life as an HTTP connection, guaranteeing full backwards compatibility with the pre-WebSocket world. The protocol switch from HTTP to WebSocket is referred to as a the WebSocket handshake.

Client handshake request
The browser sends a request to the server, indicating that it wants to switch protocols from HTTP to WebSocket. The client expresses its desire through the Upgrade header:

Listing 3.2: Client handshake request

```
GET ws://echo.websocket.org/?encoding=text HTTP/1.1
Origin: http://websocket.org
Cookie: __utma=99as
Connection: Upgrade
Host: echo.websocket.org
Sec-WebSocket-Key: uRovscZjNol/umbTt5uKmw==
Upgrade: websocket
Sec-WebSocket-Version: 13
```

Server handshake response
If the server understands the WebSocket protocol, it agrees to the protocol switch through the Upgrade header.

Listing 3.3: Server handshake response

```
HTTP/1.1 101 WebSocket Protocol Handshake
Date: Fri, 10 Feb 2012 17:38:18 GMT
Connection: Upgrade
Server: Kaazing Gateway
Upgrade: WebSocket
Access-Control-Allow-Origin: http://websocket.org
Access-Control-Allow-Credentials: true
Sec-WebSocket-Accept: rLHCkw/SKsO9GAH/ZSFhBATDKrU=
Access-Control-Allow-Headers: content-type
```

Once the server sends these headers, the handshake is complete and both parties can start exchanging data frames.
In the Last Mile Transfer (LMT) solution, a WebSocket channel is used for two purposes:

- Client-service communication over a JSON-based protocol: both parties send JSON text frames to signal different events.
- Data streaming: once the File Transfer Service (FTS) is ready, the files contents are streamed from client (WebFTS) to service (the LMT proxy) over the same already opened WebSocket connection.

3.3.2 JSON-based protocol

One of the advantages of implementing our architecture using the WebSocket protocol is that websockets give us the freedom to design our own communication protocol. This means that the WebSocket protocol is extremely useful for gluing together backend components, which is more or less what we are trying to achieve here; connecting the WebFTS interface to the actual FTS3 transfer engine.

Different kind of messages are used to trigger different events and communicate data. For instance, this message tells the WebFTS client that an endpoint for the transfer has been registered and has the URL specified in the `data` field of the JSON struct:

Transfer message

Listing 3.4: An example of a JSON protocol message

```
{
    "action":"transfer",
    "data":"https://lmt.cern.ch:8080/transfer/4d7dfd5d
        -f67a-461b-bc4e-20bf4a24c638/someFile.tar.gz"
```

```
}
```

3.3.3 HTTP

WebFTS, as mentioned in section 2.2, interacts with the underlying FTS3 engine via its RESTful API.
For instance, to submit a transfer job, WebFTS does a POST request to the corresponding FTS REST endpoint, with a JSON object that looks something like the following:

Listing 3.5: Bulk submission via the FTS RESTful API - GSIFTP origin

```
{
    "files": [
      {
        "sources": [
          "gsiftp://source.host/file"
        ],
        "destinations": [
          "gsiftp://destination.host/file"
        ],
        "metadata": "file-metadata",
        "checksum": "ADLER32:1234",
        "filesize": 1024,
        "activity": "Production"
      },
      {
        "sources": [
          "gsiftp://source.host/file2"
        ],
        "destinations": [
          "gsiftp://destination.host/file2"
        ],
        "metadata": "file2-metadata",
        "checksum": "ADLER32:4321",
        "filesize": 2048
      }
    ],
    "params": {
    }
  }
```

As can be clearly seen in the JSON object data from the above example, among the data parameters that WebFTS sends in its POST request to the FTS RESTful endpoint is an array called `sources`, which holds, for each file

to be transfered, its origin URL. In the case of the example shown above, the origin was a GSIFTP endpoint. But how would this work for a file located on a user's local storage?

For local uploads, the origin will be the endpoint that the Last Mile Transfer (LMT) proxy registered for that file. As described in section 3.2, the LMT proxy registers an endpoint for each transfer request it receives from WebFTS, and then communicates back the endpoint URL to WebFTS over the WebSocket connection. Having now received the endpoint URL, WebFTS can now submit the transfer job the same way it normally would, using an asynchronous HTTP POST request to FTS' RESTful API endpoint. The JSON object for a local upload request would look something like the following:

Listing 3.6: Bulk submission via the FTS RESTful API - local upload

```
{
    "files": [
      {
        "sources": [
          "https://lmt.cern.ch:8080/transfer/4
              d7dfd5d-f67a-461b-bc4e-20bf4a24c638/
              file1"
        ],
        "destinations": [
          "gsiftp://destination.host/file"
        ],
        "metadata": "file-metadata",
        "checksum": "ADLER32:1234",
        "filesize": 1024,
        "activity": "Production"
      },
      {
        "sources": [
          "https://lmt.cern.ch:8080/transfer/4
              d7dfd5d-f67a-461b-bc4e-20bf4a24c638/
              file2"
        ],
        "destinations": [
          "gsiftp://destination.host/file2"
        ],
        "metadata": "file2-metadata",
        "checksum": "ADLER32:4321",
        "filesize": 2048
```

```
            }
        ],
        "params": {
        }
    }
```

Notice how the `sources` array now holds a URL that refers to a LMT proxy endpoint.

3.3.4 Public Key Infrastructure (PKI)

Since the introduction of the X.509 standard for public key infrastructure (PKI) in 1988 [9], X.509 PKI and digital certificates have become a critical part of security for enterprises, governments and consumers the world over. At CERN, access to WLCG resources is authenticated using an X.509 and public key infrastructure.
Public key cryptography relies on a public and private key pair to encrypt and decrypt content. The keys are mathematically related, and content encrypted by using one of the keys can only be decrypted by using the other. The private key is kept secret. The public key is typically embedded in a binary certificate, and the certificate is published to a database that can be reached by all authorized users.
The X.509 public key infrastructure (PKI) standard identifies the requirements for robust public key certificates. A certificate is a signed data structure that binds a public key to a person, computer, or organization. Certificates are issued by certification authorities (CAs). All who are party to secure communications that make use of a public key rely on the CA to adequately verify the identities of the individuals, systems, or entities to which it issues certificates. The level of verification typically depends on the level of security required for the transaction. If the CA can suitably verify the identity of the requester, it signs (encrypts), encodes, and issues the certificate.
A certificate is a signed data structure that binds a public key to an entity. The Abstract Syntax Notation One (ASN.1) syntax for the version 3 X.509 certificate is shown in the following example.

Listing 3.7: X.509 signed certificate structure

```
--------------------------
-- X.509 signed certificate
--------------------------
SignedContent ::= SEQUENCE
{
  certificate         CertificateToBeSigned,
  algorithm           Object Identifier,
  signature           BITSTRING
}

---------------------------------
-- X.509 certificate to be signed
---------------------------------
CertificateToBeSigned ::= SEQUENCE
{
  version                  [0] CertificateVersion DEFAULT v1,
  serialNumber             CertificateSerialNumber,
  signature                AlgorithmIdentifier,
  issuer                   Name
  validity                 Validity,
```

```
  subject                 Name
  subjectPublicKeyInfo    SubjectPublicKeyInfo,
  issuerUniqueIdentifier  [1] IMPLICIT UniqueIdentifier
      OPTIONAL,
  subjectUniqueIdentifier [2] IMPLICIT UniqueIdentifier
      OPTIONAL,
  extensions              [3] Extensions OPTIONAL
}
```

Since its inception in 1998, three versions of the X.509 public key certificate standard have evolved. As shown by the following illustration, each successive version of the data structure has retained the fields that existed in the previous versions and added more. [10]

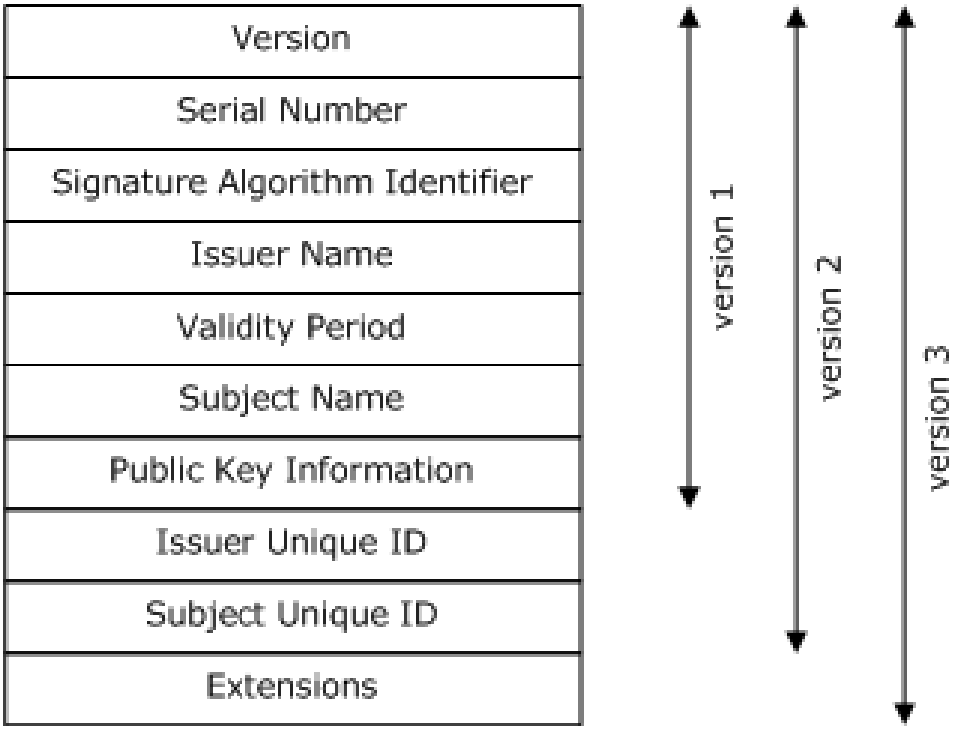


Figure 3.2: X.509 public key certificates evolution since version 1

Proxy certificates

Proxy Certificates serve to bind a unique public key to a subject name, as a public key certificate does. The use of the same format as X.509 public key certificates allows Proxy Certificates to be used in protocols and libraries in many places as if they were normal X.509 public key certificates which significantly eases implementation.
However, unlike a public key certificate, the issuer (and signer) of a Proxy

Certificate is identified by a public key certificate or another Proxy Certificate rather than a certification authority (CA) certificate. This approach allows Proxy Certificates to be created dynamically without requiring the normally heavy-weight vetting process associated with obtaining public key certificates from a CA. [11]
Proxy delegation is needed for WebFTS to gain access to Grid resources on the user's behalf. Figure 3.3 shows a diagram of how proxy delegation is done through the browser, in order to delegate access permissions to the FTS service.

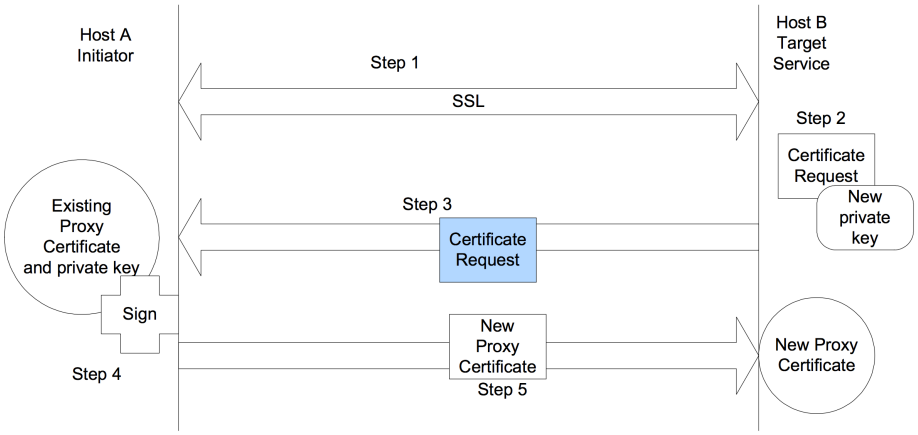


Figure 3.3: X.509 Proxy certificate delegation over a secure network connection

In the Last Mile Transfer (LMT) solution, two attributes of proxy certificates are used for two different purposes: the certificate's `DelegationID`, and its `Identity`.
The identity attribute is computed using the code shown in listing 3.8, and is later used in the LMT proxy to authorize access to submitted transfer requests: whenever the File Transfer Service (FTS) contacts the proxy and asks for a certain file, the LMT proxy checks whether the identity of the service matches the one the client (WebFTS) had when it registered the transfer.
Proxy certificates are used to check the identity of clients and services that connect to the proxy server.

Listing 3.8: Code used to get the identity of a X.509 proxy certificate

```
// getIdentity returns the original user identity.
func (p *X509Proxy) getIdentity() (pkix.Name, error) {
  cert := p.getEndUserCertificate()
```

```
  if cert == nil {
    return pkix.Name{}, errors.New("Could not get the
       end user certificate")
  }
  return cert.Subject, nil
}
```

As for the `DelegationID` attribute, it is used in conjunction with the file name to uniquely identify transfers in an internal map that the LMT proxy keeps of open connections.

Listing 3.9: Code showing the internal structure of the proxy's transfers map

```
// client represents a WebSocket client.
type client struct {
  ID string
  Ws *websocket.Conn
}
// fileData represents the details of the file to be
   transfered.
type fileData struct {
  Name string `json:"name,omitempty"`
  Size int64  `json:"size,omitempty"`
}
// transfer represents a transfer request submitted by
    a client.
type transfer struct {
  client   *client
  fileData *fileData
  identity string
  endPoint string
}
// Transfers maps a transfer request submitted via
   websocket to an endpoint.
var Transfers map[string]*transfer

func init() {
        Transfers = make(map[string]*transfer)
}
```

Code listing 3.10 shows how the keys for the `Transfers` map get created.

Listing 3.10: Code showing how transfers get registered in LMT's transfers map

```
// TransferID concatenates the user's delegation ID
    and
// the filename the user's wish to transfer to create
// the transfer endpoint.
func TransferID(delegationID, filename string) string
    {
  return fmt.Sprintf("%s/%s", delegationID, filename)
}

// registerClient creates a new client and adds it to
    the Transfers map.
func registerClient(ws *websocket.Conn, f *fileData) *
    client {
        req := ws.Request()
        dumpReq, err := httputil.DumpRequest(req, true
            )
        if err != nil {
                log.Error(err)
        }
        log.WithFields(logrus.Fields{
                "event": "ws_http_request",
                "data":  string(dumpReq),
        }).Info(string(dumpReq))

        identity, err := X509Identity(req)
        if err != nil {
                log.Error(err)
        }

        delegationID, err := X509DelegationID(req)
        if err != nil {
                log.Error(err)
        }
        // add new transfer to the map.
        transferID := TransferID(delegationID, f.Name)
        c := &client{
                ID: transferID,
                Ws: ws,
        }
```

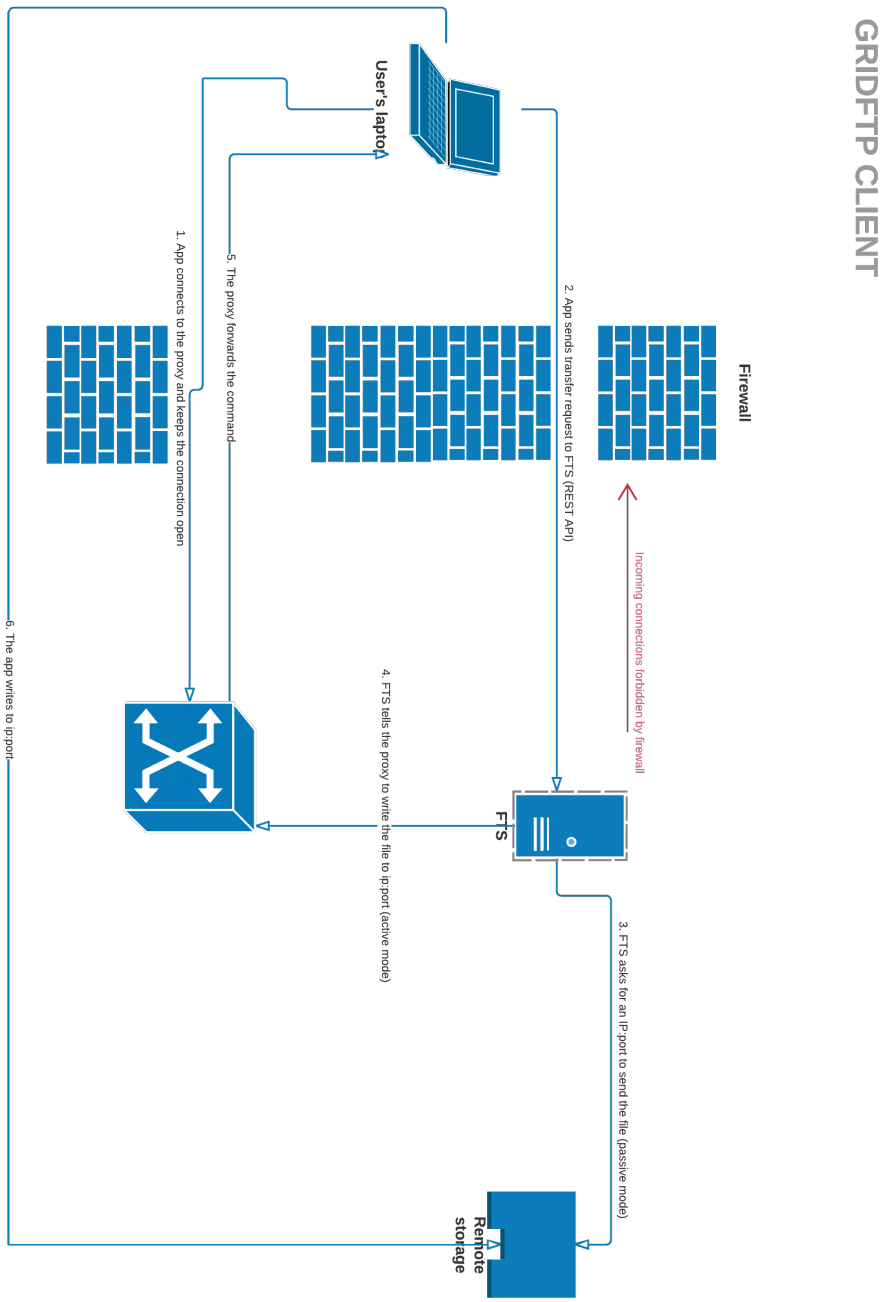
```
        Transfers[transferID] = &transfer{
                client:    c,
                fileData: f,
                identity: voms.NameRepr(&identity),
                endPoint: fmt.Sprintf("%s/%s", BaseURL
                    , transferID),
        }
        return c
}
```

3.4 Alternative solutions

Implementing a FTP desktop client might be the first answer that comes to mind when one considers the problem at hand. As a matter of fact, it is the approach used by some commercial solutions, most notably the one developed by the Globus Project, called Globus Connect. [12]

3.4.1 GridFTP desktop client

Transferring via FTP [13] involves in fact two connections: a control channel, and a data channel. The control channel is used to issue commands, and recover status code and simple responses. The data channel is used when a control command involves the transmission of large amount of data. For instance, listing a directory, or downloading/uploading a file.
The data channel can be established in an 'active' fashion (the server connects back to an ip:port provided by the client), or 'passive' (the client connects to an ip:port provided by the server). Passive allows transfers to happen even through firewalls and NATs.
Most of the transfers done today on the Grid uses GridFTP [14] (a superset of FTP), which implements the transferring with one of the storages as Active (connects into a given IP:port), and the other as Passive (waits for a connection in a given IP:port). This is called File eXchange Protocol , and is part of GridFTP.
This means we can actually proxy only the control channel, and handle the transfers in a manner where the user's host always does outbound connections.
The amount of data transferred via the control channel will be only a few kilobytes at most per transfer, independently of the file size.
The transfer will actually happen directly from the laptop to the server, or the other way around.



Advantages of this approach:

- Takes advantage of an existing, well known protocol as FTP. storage.
- Each required block is kept independent, so we can start with a bare minimum client, and get the whole chain working before thinking of

more fancy stuff (i.e. detect type of connectivity before attempting to transfer, retries, copying whole folders...)

Difficulties:

- The proxy will be a sort of multiplexer, so it will have to look at the initial exchange of commands to know where to forward the control traffic (i.e. based on a randomly unique generated user).
- Normally, third party copies always sends the PASV command to the destination storage. This works well for sending file from the client to the server, but not the other way around.
- The connection between the client and the proxy can be lost. The client needs to be able to retry a transfer. desirable.
- The client must be multi-platform: GNU/Linux, MacOSX and Windows the very least (we do not consider that anyone would have this much data on a mobile platform.) dubious anyone would have this much data on a mobile platform.)
- Security considerations in general.

Chapter 4

Architecture Analysis

4.1 Architecture Requirements

Rational Unified Process (RUP) [15] gives the following definition for any requirement:

> A requirement describes a condition or capability to which a system must conform; either derived directly from user needs, or stated in a contract, standard, specification, or other formally imposed document. [16]

An architectural requirement, in turn, is any requirement that is architecturally significant, whether this significance be implicit or explicit.

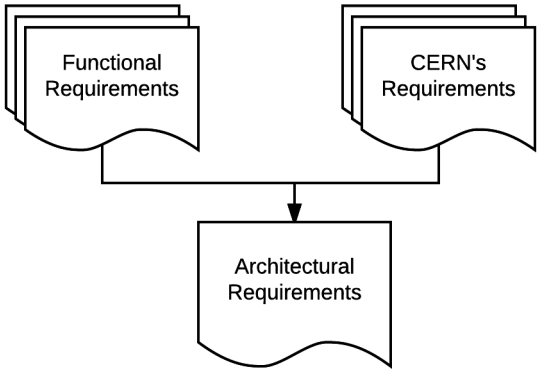


Figure 4.1: Inputs and outputs for determining architectural requirements

4.2 Functional Requirements

Functional requirements define the functions of a system or its components. Below are the ones that the Last Mile Transfer solution needs to meet:

	Create a proxy service for FTS
Description	The proxy should enable the File Transfer Service (FTS) to get a file from a user's laptop via their browser.

Table 4.1: Functional requirement-1

	Integrate the proxy service with WebFTS
Description	The service should be seamlessly integrated with WebFTS, and transfers should have the same behavior independent of their origin, be it local or remote.

Table 4.2: Functional requirement-2

4.3 Non-functional requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. In this section we will go through the non-functional requirements that CERN had for the Last Mile Transfer (LMT) project.

4.3.1 Compatibility

For the Last Mile Transfer project, cross-platform compatibility is a must; meaning that the solution needs to work for all Grid users, whether they are using MacOS, Windows, or any distribution of the GNU/Linux operating system.
Since the client-side solution is web-based, that is, any Grid user can access the WebFTS site via a web browser, the only extra requirement in order for local file uploads to function correctly, is that the web browser supports the WebSocket protocol.

As we can see from table 4.3, the WebSocket protocol is supported across all modern web browsers.

IE	Edge	Firefox	Chrome	Safari	Opera
			49		
		52	60		
	15	55	61	10.1	
11	16	56	62	11	48
		57	63	TP	49
		58	64		50
		59	65		

Table 4.3: Support for the WebSocket protocol across different popular web browsers [1]

4.3.2 Availability

Availability is related to an application's reliability. If an application is not available for use when needed, then it's unlikely to be fulfilling its functional requirements. Availability is relatively easy to specify and measure. In terms of specification, many IT applications must be available at least during normal business hours. Most Internet sites desire 100% availability, as there are no regular business hours on-line. For a live system, availability can be measured by the proportion of the required time it is available.
High availability for the Last Mile proxy is desirable, so we took the architectural decision necessary so that the solution supports mechanisms such as horizontal scaling [17].

4.3.3 Integration

Integration is concerned with the ease with which an application can be usefully incorporated into a broader application context. The value of an application or component can frequently be greatly increased if its functionality or data can be used in ways that the designer did not originally anticipate. [18]

Integration with WebFTS

WebFTS [19] is a file transfer and management solution which allows users to invoke reliable, managed data transfers on distributed infrastructures through a web browser.
The integration requirements for the Last Mile Transfer project resolve around the goal of having it seamlessly integrated with WebFTS.

4.3.4 Security

The security measurements that were undertaken for the Last Mile Transfer solution are the same ones that are used across all FTS projects in general.

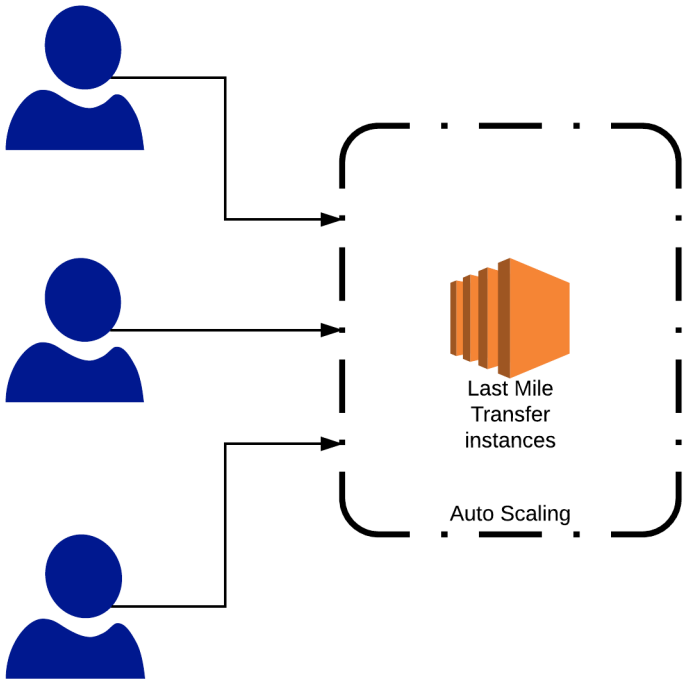


Figure 4.2: Horizontally scaling the LMT proxy as the number of requests increases

All communications from and to the The Last Mile Transfer proxy are secured by TLS [20], to ensure the integrity and privacy of communications. Aside from that, X.509 proxy certificates [21] are used to authenticate requests from both client (WebFTS) and service (FTS3).

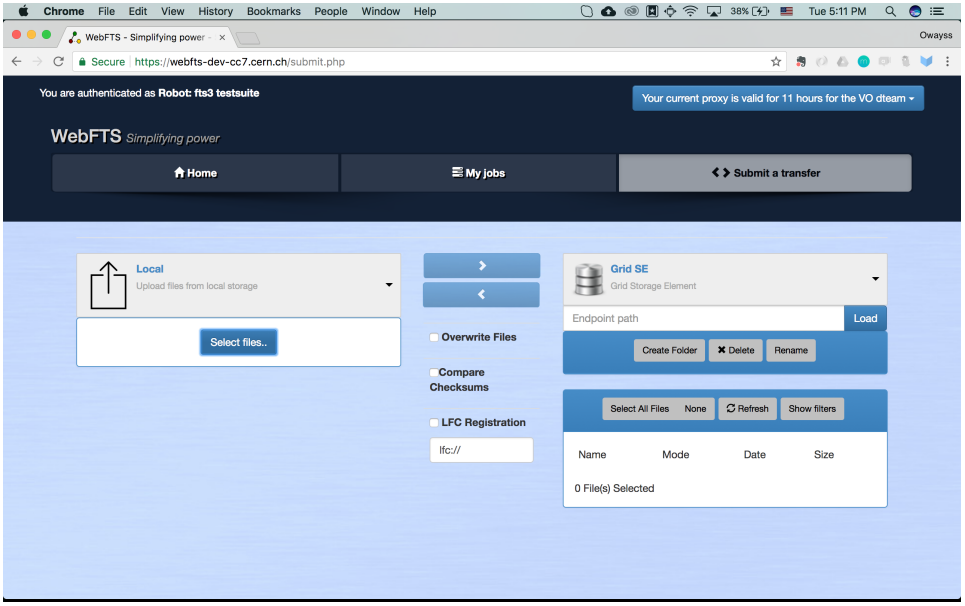


Figure 4.3: WebFTS - Local upload

4.4 Summary of the non-functional requirements

Table 4.4 shows a summary of all non-functional requirements for the Last Mile Solution project, and how they have been dealt with.

Quality attribute	Stimulus	Response
Availability & Scalability	The LMT proxy server is down, or there too many concurrent connections	The LMT solution can be scaled both horizontally (by provisioning new LMT instances) and vertically (adding more memory to a particular LMT instance). Although vertical scaling will almost certainly not be necessary, as Go routines scale up very well and have very little memory impact.
Security	Unauthorized requests made to the proxy	The LMT proxy will reject any request that does not have a valid X.509 proxy certificate. Moreover, all sensitive information is routed through a WebSocket Secure (WSS) channel.
Integration	LMT must seamlessly integrate within the WebFTS solution	Local file uploads are done in the same manner as any other type of transfer carried out via WebFTS, thus making sure the User Experience (UX) is the same for all different storage types.

Table 4.4: A summary of all the non-functional requirements for the Last Mile Transfer project

4.5 Risk Management

Risk management is the identification, assessment, and prioritization of risks (defined in ISO 31000 [22] as *the effect of uncertainty on objectives*) followed by coordinated and economical application of resources to minimize, monitor, and control the probability and/or impact of unfortunate events [23] or to maximize the realization of opportunities.
Having established the context of the Last Mile Transfer project and specified the project's requirements, a risk management plan that included the following potential risks was made.

Risk	The proposed solution turns out to be unfeasible, due to some technical error made in the architecture design stage.
Likelihood	Although the architecture for the proposed solution has been carefully crafted, this was still considered a serious risk.
Impact	The impact of this scenario would be huge, as it would imply re-thinking the whole problem and coming up with a different solution.
Action plan	Release an early prototype providing a proof of concept as early in the development stage as possible.

Table 4.5: Risk-1

Risk	Late delivery due to technical difficulties or lack of knowledge about the problem domain.
Likelihood	The likelihood of occurrence was considered high since the solution involves working with a complicated distributed system.
Impact	This scenario would have a big impact on the Last Mile Transfer project, as the collaboration contract between the two institutions (CERN and the University of Cádiz) has a strict deadline that needs to be met.
Action plan	Schedule a first visit to the IT department in the initial stage of the project, in order to get familiar with the problem domain, and to get feedback from the IT Storage group on the proposed solution.

Table 4.6: Risk-2

Chapter 5

Project Development

5.1 Version Control

Version control systems are a category of software tools that help a software team manage changes to source code over time. Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members. [24]
For both parts of the Last Mile Transfer (LMT) solution, the open-source distributed version control system Git [25] was used.
The code for the LMT proxy server lives on a separate repository within the FTS project on CERN's GitLab [26], while the LMT extension for WebFTS has been developed on a new branch of the original WebFTS code repository [27].

5.2 Testing

Software testing is the process of validating and verifying that a software program/application/product: [28]

- meets the business and technical requirements that guided its design and development;
- works as expected; and
- can be implemented with the same characteristics.

In this section we will go through the different levels of testing used in the Last Mile Transfer project.

5.2.1 Unit testing

Kent Beck [29] introduced the concept of unit testing in Smalltalk, and it has carried on into many other programming languages, making unit testing an extremely useful practice in software programming.
A unit test is a piece of a code (usually a method) that invokes another piece of code and checks the correctness of some assumptions afterward. If the assumptions turn out to be wrong, the unit test has failed. A "unit" is a method or function. [30]
For the Last Mile Transfer project, we used the built-in testing command that the Go programming language has.
In Go, tests are written in separate files that have the `_test` suffix in their name. For instance, the tests for the `proxy/client.go` code can be found in the `proxy/client_test.go` file.

5.2.2 Automated tests

Test automation is the use of special software (separate from the software being tested) to control the execution of tests and the comparison of actual outcomes with predicted outcomes. [31]
Since all of the CERN projects are hosted on the GitLab platform, we used the GitLab Continuous Integration & Deployment solution for test automation.

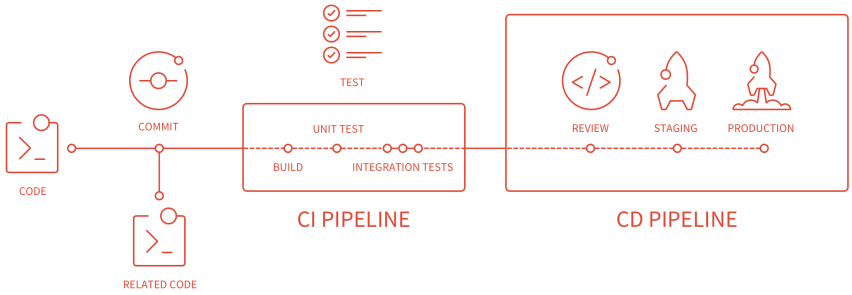


Figure 5.1: GitLab Continuous Integration & Continuos Deployment pipeline

Continuous Integration is a software development practice in which you build and test software every time a developer pushes code to the application, and it happens several times a day. [32]
To use the GitLab CI we create a `.gitlab-ci.yml` file at the root directory of our repository, configure our GitLab project to use a Runner [33], then each commit or push, triggers our CI pipeline.

Below is the `.gitlab-ci.yml` file for the `LMT` project.

Listing 5.1: GitLab CI configuration file

```
image: golang:1.8

stages:
  - test
  - build

before_script:
  - go get -u github.com/golang/dep/cmd/dep
  - export GOPATH=$(dirname $CI_PROJECT_DIR
    )/go
  - mkdir -p $GOPATH/src
  - cd $GOPATH/src
  - ln -s $CI_PROJECT_DIR
  - cd $CI_PROJECT_NAME
  - dep ensure -update

test:
  stage: test
  script:
    - go test

build:
  stage: build
  script:
    - go build
```

5.2.3 User Acceptance Testing (UAT)

Acceptance testing is a formal type of software testing that is performed by end user when the features have been delivered by developers. The aim of this testing is to check if the software confirms to their business needs and to the requirements provided earlier.
User Acceptance testing is a must for any project; it is performed by clients/end users of the software. User Acceptance testing allows SMEs (Subject matter experts) from client to test the software with their actual business or real-world scenarios and to check if the software meets their business requirements.
As mentioned in section 1, the project was presented at the IT Storage group monthly meeting on the 6th of October of this current year, 2017. The presentation included a live demo of the developed service and it was

well received.
The solution was also tested by the IT-ST-AD section members at CERN, and was later deployed to a staging machine where further tests are still being made.

5.3 Technologies used

5.3.1 Programming languages

Go

Go is a programming language created at Google in 2009 by Robert Griesemer, Rob Pike, and Ken Thompson. [34] It is a compiled, statically typed language in the tradition of Algol and C, with garbage collection, limited structural typing, memory safety features and CSP-style concurrent programming features added. The compiler and other language tools originally developed by Google are all free and open source.
Go was chosen for the development of the Last Mile Transfer (LMT) proxy service as it scales very well for this type of workload; a single Go process can easily handle hundreds of thousands of Goroutines [35], where each one would be a connection handler for a given client.

JavaScript

JavaScript, the language of the web, was used to implement the WebFTS extension part of the Last Mile Transfer (LMT) solution, that is; the client-side code.
To enable WebFTS to maintain bidirectional communications with the LMT proxy service, we used the WebSocket API introduced in HTML5. [36]

5.3.2 Tooling

Dependency management

In a world without dependencies, projects would exist in isolation, wouldn't need help from an internal/external teams or vendors, wouldn't be affected by business changes, and requirements would remain constant. If this were true, project and program managers could just skip managing dependencies. [37]
In reality, this is not true, and almost all projects have external and complicated dependencies. For the Last Mile Transfer proxy, we used the `dep` [38] tool to manage Go package dependencies.

Reproducible Builds

A build is reproducible if given the same source code, build environment and build instructions, any party can recreate bit-by-bit identical copies of all specified artifacts.
The relevant attributes of the build environment, the build instructions and the source code as well as the expected reproducible artifacts are defined by the authors or distributors. The artifacts of a build are the parts of the build results that are the desired primary output. [39]
To ensure reproducible builds of the Last Mile Transfer (LMT) proxy, all packages have been versioned and added to the project repository in a `/vendor` directory, below is the `Gopkg.lock` containing all the metadata for the packages that the LMT proxy depends on:

Listing 5.2: Gopkg.lock file for the LMT proxy

```
# This file is autogenerated, do not edit; changes may be
    undone by the next 'dep ensure'.


[[projects]]
  name = "github.com/Sirupsen/logrus"
  packages = ["."]
  revision = "f006c2ac4710855cf0f916dd6b77acf6b048dc6e"
  version = "v1.0.3"

[[projects]]
  name = "github.com/gorilla/context"
  packages = ["."]
  revision = "1ea25387ff6f684839d82767c1733ff4d4d15d0a"
  version = "v1.1"

[[projects]]
  name = "github.com/gorilla/mux"
  packages = ["."]
  revision = "24fca303ac6da784b9e8269f724ddeb0b2eea5e7"
  version = "v1.5.0"

[[projects]]
  branch = "master"
  name = "gitlab.cern.ch/flutter/go-proxy"
  packages = ["."]
  revision = "919a10a449ccedd5e01b1badaf7e82cd45de46dc"

[[projects]]
  branch = "master"
  name = "golang.org/x/crypto"
  packages = ["ssh/terminal"]
  revision = "847319b7fc94cab682988f93da778204da164588"

[[projects]]
```

```
  branch = "master"
  name = "golang.org/x/net"
  packages = ["websocket"]
  revision = "0744d001aa8470aaa53df28d32e5ceeb8af9bd70"

[[projects]]
  branch = "master"
  name = "golang.org/x/sys"
  packages = ["unix","windows"]
  revision = "429f518978ab01db8bb6f44b66785088e7fba58b"

[[projects]]
  branch = "v2"
  name = "gopkg.in/yaml.v2"
  packages = ["."]
  revision = "eb3733d160e74a9c7e442f435eb3bea458e1d19f"

[solve-meta]
  analyzer-name = "dep"
  analyzer-version = 1
  inputs-digest = "
      a8f094ad256fbecc0cd8a9fb4d49a12a6bea7028c573762e2a
      314a225510a80f"
  solver-name = "gps-cdcl"
  solver-version = 1
```

5.4 Code Quality

5.4.1 Structured logs

Structured logging is the practice of using a consistent, predetermined message format containing semantic information. The format could be XML, JSON, or any other format. Structured logging can be used for two different purposes:

- Process log files for analytics or business intelligence.
- Being able to search and correlate log messages is very valuable to development teams during the development process and for troubleshooting production problems.

A structured logging technique was adopted throughout the LMT proxy codebase. An example of this is shown in listing 5.3.

Listing 5.3: Example of a structured log message from the LMT codebase

```
log.WithFields(logrus.Fields{
  "event": "access_forbidden",
  "data":   string(voms.NameRepr(&identity)),
```

```
}).Error(errAccessForbidden)
```

5.4.2 Static analysis

We used the Go Report Card [40] open-source project to test some quality aspects of our LMT proxy code. Go Report Card uses several quality measures, such as `gofmt` to check for source code formatting, `go vet` to examine source code and report suspicious constructs, `go lint` to point out style mistakes, and `go cyclo` to calculate cyclomatic complexities of functions in Go source code. Below are the results from running the tool on our LMT proxy repository.

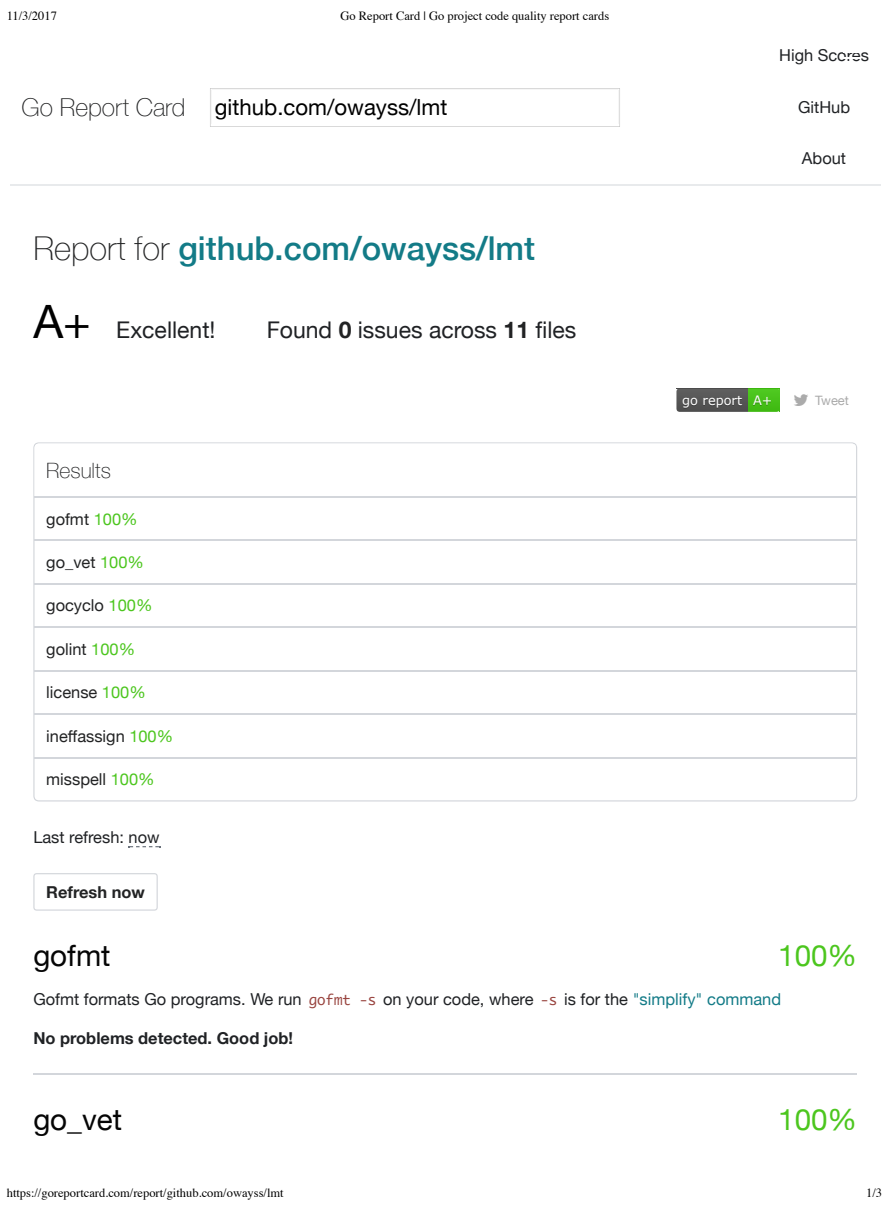
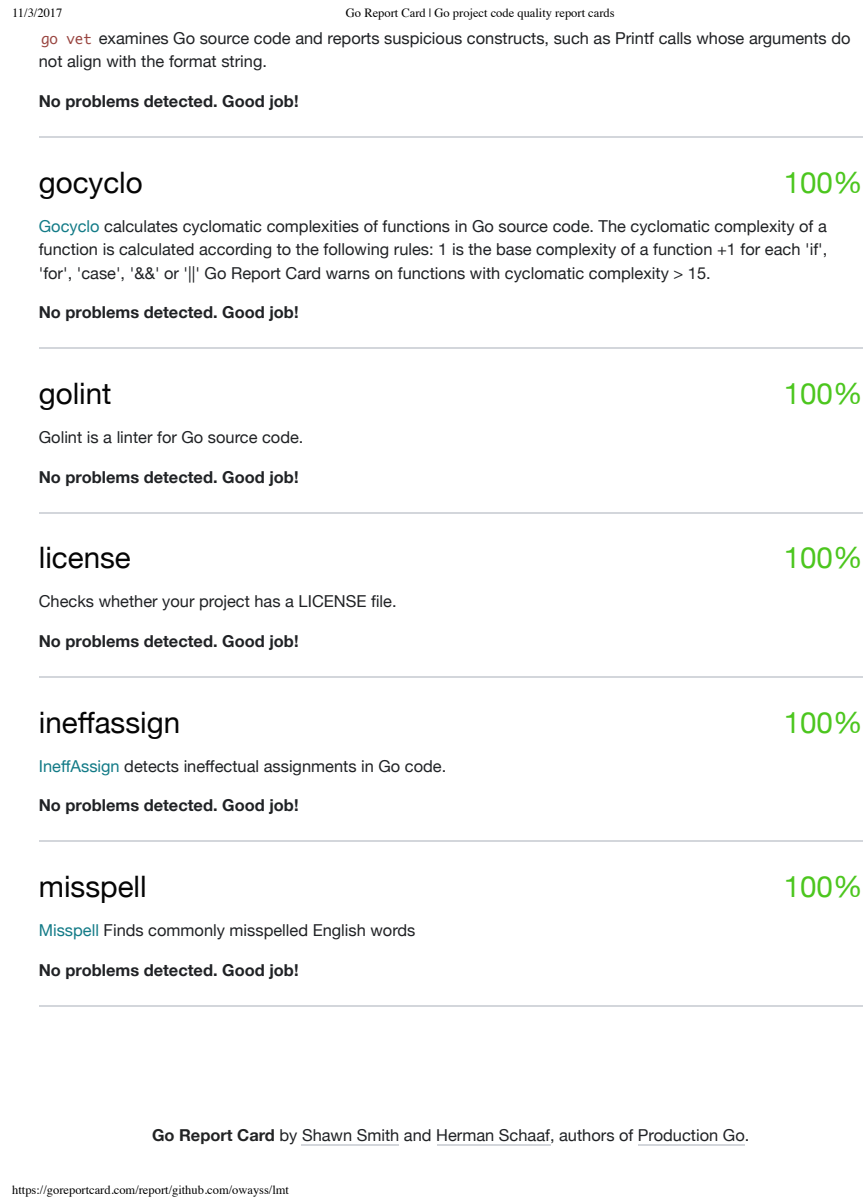


Figure 5.2: Code quality report generated for the LMT proxy project, page 1



11/3/2017 Go Report Card | Go project code quality report cards

`go vet` examines Go source code and reports suspicious constructs, such as Printf calls whose arguments do not align with the format string.

No problems detected. Good job!

gocyclo 100%

Gocyclo calculates cyclomatic complexities of functions in Go source code. The cyclomatic complexity of a function is calculated according to the following rules: 1 is the base complexity of a function +1 for each 'if', 'for', 'case', '&&' or '||' Go Report Card warns on functions with cyclomatic complexity > 15.

No problems detected. Good job!

golint 100%

Golint is a linter for Go source code.

No problems detected. Good job!

license 100%

Checks whether your project has a LICENSE file.

No problems detected. Good job!

ineffassign 100%

IneffAssign detects ineffectual assignments in Go code.

No problems detected. Good job!

misspell 100%

Misspell Finds commonly misspelled English words

No problems detected. Good job!

Go Report Card by Shawn Smith and Herman Schaaf, authors of Production Go.

https://goreportcard.com/report/github.com/owayss/lmt 2/3

Figure 5.3: Code quality report generated for the LMT proxy project, page 2

5.5 Git statistics

Since all of our code is kept in git repositories, we can use the `git log` command to obtain some statistics regarding commits made.

The LMT proxy repository

Listing 5.4: Git stats - LMT proxy repository

```
~/go/src/gitlab.cern.ch/fts/lmt
git shortlog -s -n
67  Owayss Kabtoul

git log --author="Owayss Kabtoul" --pretty=tformat: \\
--numstat | awk '{ add += $1; subs += $2; loc += $1 - $2 } END
   { printf \\
"Added lines: %s\nRemoved lines: %s\nTotal lines: %s\n", add,
   subs, loc }' -
Added lines: 412309
Removed lines: 1338
Total lines: 410971
```

The WebFTS repository

Listing 5.5: Git stats - WebFTS repository

```
~/cern/webfts
git shortlog -s -n
108  Andres Abad Rodriguez
107  andrea-manzi
 65  aabadz
 62  amanzi
 58  christina-skarpi
 15  Owayss Kabtoul
 14  Andrea
  4  andreamanzi
  2  root
  1  Martin Hellmich

git log --author="Owayss Kabtoul" --pretty=tformat: \\
--numstat | awk '{ add += $1; subs += $2; loc += $1 - $2 } END
   { printf \\
"Added lines: %s\nRemoved lines: %s\nTotal lines: %s\n", add,
   subs, loc }' -
Added lines: 418
Removed lines: 262
Total lines: 156
```

Chapter 6

Project management

In this chapter we will discuss aspects relevant to the development of the Last Mile Transfer project, such as the development methodology used, the different stages of development that the project went through, as well as the tools and techniques used for time management.

6.1 Development Cycle

The ISO/IEC 12207-1 standard defines the life cycle model as a

> framework of processes and activities concerned with the life cycle that may be organized into stages, which also acts as a common reference for communication and understanding. [41]

The development of the present project followed the iterative and incremental model, a structured and agile software development methodology.

6.2 Stages

6.2.1 First stage: Preliminary analysis

- Getting to understand the problem domain: mainly the WLCG infrastructure.
- Understanding how the File Transfer Service works.
- Thinking deeply about the problem, and coming up with different solutions.
- Discussing all the possible solutions with the IT-ST-AD [42] group at CERN.
- Deciding on the architecture, and the different technologies to be used for the development of this project.

6.2.2 Second stage: A proof of concept

- Develop a deep understanding of how the WebSocket protocol works.
- Consider the different available implementations of the WebSocket protocol.
- Implement a PoC: a small mock site to transfer a file through a WebSocket connection.
- Getting familiar with the GFAL2 libraries.
- Doing manual integration tests using the GFAL2 library bindings.
- Review the proof.

6.2.3 Third stage: The proxy service

- Experimenting with different implementations of web servers in Go.
- Getting familiar with the WebSocket API.
- Designing a JSON-based protocol for use in client-service communications.
- Developing a scalable on-demand proxy service in Go.
- Testing the service.

6.2.4 Fourth stage: Integration with WebFTS

- Reading the WebFTS code base.
- Integrating the JavaScript client with the WebFTS project.
- Testing the interface.
- Doing end-to-end tests.

6.2.5 Fifth stage: Deployment

- Solving dependency management issues.
- Packaging the server into one statically compiled binary.
- Learning about Docker images.
- Writing a minimal Docker container image for the proxy service.
- Deploying the proxy service to a CERN machine.
- Deploy a new version of WebFTS to a staging machine.

6.3 Project schedule

This section describes the different time management techniques used in the Last Mile Transfer project.

6.3.1 First approach: A PERT diagram

The program (or project) evaluation and review technique, commonly abbreviated PERT, is a statistical tool, used in project management, which was designed to analyze and represent the tasks involved in completing a given project. [43]
PERT charts are generally used before a project begins to plan and determine the duration of each task—so they don't have to show the actual dates of your project. They also do a better job of showing whether certain tasks need to be completed in order or whether they can be completed simultaneously.
In the early stages of the Last Mile project, the PERT chart shown in figure 6.1 was used.

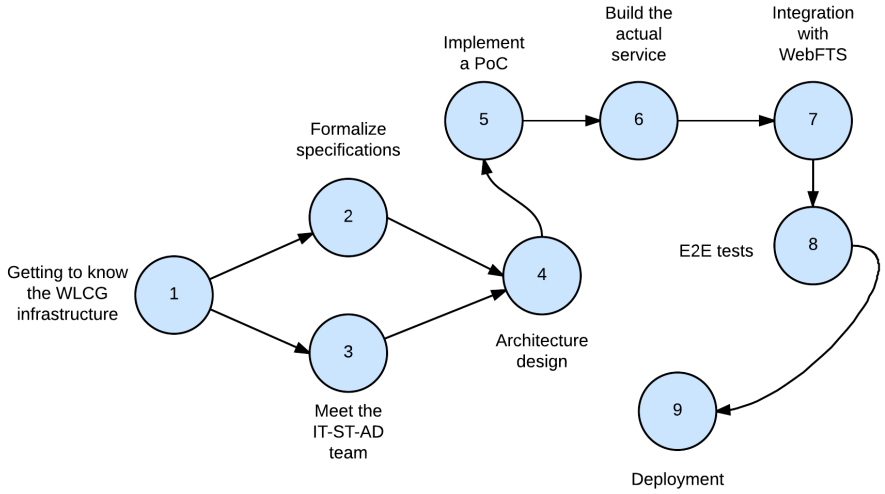
6.3.2 Gantt chart

A Gantt chart is a type of bar chart that illustrates a project schedule. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements comprise the work breakdown structure of the project. Modern Gantt charts also show the dependency (i.e., precedence network) relationships between activities. Gantt charts can be used to show current schedule status using percent-complete shadings and a vertical "TODAY" line. [44]

6.3.3 Time table

Stage	Start date	End date	Duration
Preliminary analysis	1/4/2017	6/5/2017	36 days
A proof of concept	7/5/2017	7/7/2017	61 days
The proxy service	8/7/2017	30/9/2017	84 days
Integration with WebFTS	1/10/2017	13/10/2017	13 days
Deployment	14/10/2017	28/10/2017	14 days

Table 6.1: Time schedule for the Last Mile Transfer project



Activity	Duration (weeks)	Immediate Predecessor Activities
1	2.5	-
2	2	1
3	1	1
4	2	2, 3
5	.5	4
6	4	5
7	3	6
8	1	7
9	1	8

Figure 6.1: Initial PERT diagram for the Last Mile Transfer project

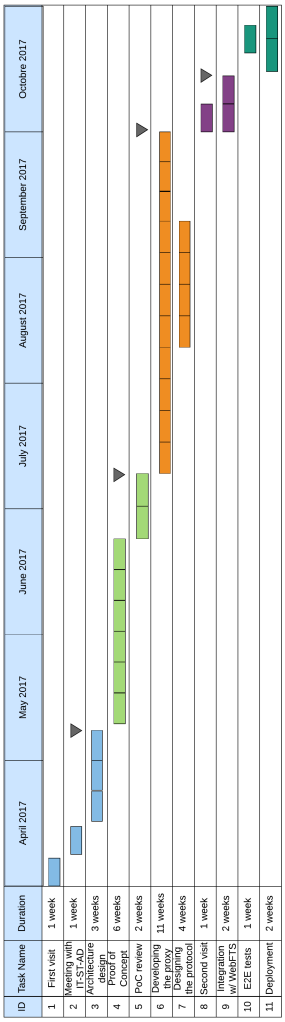


Figure 6.2: Gantt chart for the Last Mile Transfer project

6.4 Costs

To estimate the cost of the Last Mile Transfer project we will have to estimate the costs of two types of resources: human resources, and non-human ones.

6.4.1 Human resources

In terms of human labour, the Last Mile Transfer project spanned the period of 7 months (208 days, as can be observed from table 6.1), working for 5 hours a day. This adds up to 1040 hours in total.
To estimate the hourly cost of the human labour involved, we are going to consider the document [45], which establishes that the salary for a R&D associate working five hours a week at a university in the autonomous community of Andalusia is of €496.03.
Therefore, the cost of the human labour involved can be estimated as:

$$HumanResourcesCost(C_h) = 7 \times 496.03 = 3427.21$$

6.4.2 Non-human resources

The project has been entirely developed on a single 2015 Macbook Pro machine [46], which had the cost of €1190. Another expense that should be taken into account is the cost of the internet connection, which was of €35 per month, which adds up to €245 for the total development period, which lasted 7 months.
Therefore, the cost of non-human resources involved in this project is:

$$NonHumanResourcesCost(C_n) = 1190 + 245 = 1435$$

Project total cost can then be calculated as the sum of the human and non-human resources cost:

$$TotalCost(C) = HumanResourcesCost(C_h) + NonHumanResourcesCost(C_n)$$

$$TotalCost(C) = 3427.21 + 1453 = 4880.21$$

Hence, the total cost of the Last Mile Transfer project is estimated as €4880.21.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we addressed the problem of local data transfers on World-wide LHC Computing Grid infrastructure, a functionality that Grid users have long been asking for.
The proposed solution uses the WebSocket protocol, leverages the power of FTS3 service, and builds upon the existing WebFTS solution to enable Grid users to transfer data from their local storage to remote storage elements on the WLCG network, and it does so securely, reliably, and efficiently.
We hope that this project contributes to the adoption of the open-source File Transfer Service (FTS) project across other institution outside of CERN, and that it helps to increase the use of WebFTS as an intuitive and easy to use solution for end-users.

7.2 Future work

In this section, we will give some recommendations for future work on the Last Mile Transfer (LMT) project.

7.2.1 WebDAV support

Web Distributed Authoring and Versioning (WebDAV) is an extension to Hypertext Transfer Protocol (HTTP) that defines how basic file functions such as copy, move, delete, and create are performed by using HTTP. [47]
It is defined in RFC 4918 [48] by a working group of the Internet Engineering Task Force.
The key to scalable data distribution is the 3rd party copy (3pc) - passing

data directly from source to destination, bypassing the client. FTS will do this automatically for protocols with native support, in particular gridFTP and xrootd.
Currently, whenever the LMT proxy receives a COPY request to do a 3rd party pull, it will respond with a `405 HTTP Method Not Allowed` status code. Consequently, the destination will then send a regular `HTTP GET` request, and the LMT proxy would start streaming the requested content to the `GET` request's response body.
It would be useful to add support for WebDAV methods to the LMT proxy server. In fact, there is already a Go library [49] with an implementation of WebDAV methods that looks very promising, and could be used in the future development of this project.

7.2.2 File downloads

Though not the main use case of the Last Mile Transfer solution, it might be of use to some Grid users to be able to download a file from a remote storage point on the WLCG network to their local storage.
There is no technical reason for why this should not be possible using the same protocol that was developed for this project. In fact, using the Last Mile Transfer proxy, file downloads should be quite an easy feature to add in the future: work need only be done on the WebFTS extension, and file streaming from a remote origin to a user's local storage can happen through the LMT proxy, exactly in the same manner it works for file uploads.

Bibliography

[1] "WebSocket cross-browser support." http://caniuse.com/#feat=websockets. Accessed: 2017-10-31.

[2] "CERN." http://home.cern/. Accessed: 2017-09-21.

[3] "Worldride LHC Computing Grid." http://wlcg-public.web.cern.ch/. Accessed: 2017-09-21.

[4] "Open Grid Forum." https://www.ogf.org/ogf/doku.php. Accessed: 2017-11-1.

[5] "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile." https://www.ietf.org/rfc/rfc5280.txt. Accessed: 2017-09-21.

[6] "FTS Documentation." http://fts3-docs.web.cern.ch/fts3-docs/. Accessed: 2017-09-21.

[7] "WebFTS Documentation." http://fts3-docs.web.cern.ch/fts3-docs/docs/webfts.html. Accessed: 2017-10-30.

[8] "The WebSocket Protocol." https://tools.ietf.org/html/rfc6455. Accessed: 2017-09-21.

[9] "Internet X.509 Public Key Infrastructure Certificate and CRL Profile." https://www.ietf.org/rfc/rfc2459.txt. Accessed: 2017-09-21.

[10] "X.509 Public Key Certificates." https://msdn.microsoft.com/en-us/library/windows/desktop/bb540819(v=vs.85).aspx. Accessed: 2017-09-21.

[11] V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, J. Gawor, S. Meder, and F. Siebenlist, "X.509 proxy certificates for dynamic delegation," 2004.

[12] "Globus Connect." https://www.globus.org/globus-connect. Accessed: 2017-09-21.

[13] "FTP Protocol." https://www.w3.org/Protocols/rfc959/. Accessed: 2017-09-21.

[14] "GridFTP Protocol." https://en.wikipedia.org/wiki/GridFTP. Accessed: 2017-09-21.

[15] "Rational Unified Process - Wikipedia." https://en.wikipedia.org/wiki/Rational_Unified_Process. Accessed: 2017-10-11.

[16] "IBM, Rational Unified Process - Capturing Architectural Requirements." https://www.ibm.com/developerworks/rational/library/4706-pdf.pdf. Accessed: 2017-10-11.

[17] "IBM - Vertical and horizontal scaling in the cloud." https://www.ibm.com/blogs/cloud-computing/2014/04/explain-vertical-horizontal-scaling-cloud/. Accessed: 2017-10-11.

[18] I. Gorton, *Essential Software Architecture*. Springer, 2 ed., 2011.

[19] "WebFTS - CERN." https://webfts.cern.ch/. Accessed: 2017-09-21.

[20] "The Transport Layer Security (TLS) Protocol." https://tools.ietf.org/html/rfc5246. Accessed: 2017-09-21.

[21] "The WebSocket Protocol." https://www.ietf.org/rfc/rfc3820.txt. Accessed: 2017-09-21.

[22] A. Dali and C. Lajtha, "Iso 31000 risk management: “the gold standard”," *EDPACS*, vol. 45, pp. 1–8, May 2012.

[23] D. W. Hubbard, "The failure of risk management: Why it's broken and how to fix it / edition 1."

[24] "Version Control - Atlassian." https://www.atlassian.com/git/tutorials/what-is-version-control. Accessed: 2017-10-21.

[25] "Git - Version Control." https://git-scm.com/. Accessed: 2017-10-21.

[26] "LMT proxy repository - GitLab." https://gitlab.cern.ch/fts/lmt. Accessed: 2017-10-21.

[27] "WebFTS repository - lmt branch - GitLab." https://gitlab.cern.ch/fts/webfts/tree/lmt. Accessed: 2017-10-21.

[28] C. Kaner, J. Bach, and B. Pettichord, *Lessons Learned in Software Testing*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[29] "Kent Beck - Wikipedia." https://en.wikipedia.org/wiki/Kent_Beck. Accessed: 2017-10-27.

[30] R. Osherove, M. Feathers, and R. C. Martin, *The Art of Unit Testing: Deutsche Ausgabe*. Mitp, 2015.

[31] D. Huizinga and A. Kolawa, *Automated Defect Prevention: Best Practices in Software Management*. Wiley-IEEE Computer Society Pr, 2007.

[32] "Continuous Integration - Wikipedia." https://en.wikipedia.org/wiki/Continuous_integration. Accessed: 2017-10-22.

[33] "GitLab CI - Runners." https://docs.gitlab.com/ee/ci/runners/README.html. Accessed: 2017-10-27.

[34] R. Pike, "Go at google," *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity - SPLASH 12*, 2012.

[35] "What are goroutines?." https://golang.org/doc/faq#goroutines. Accessed: 2017-10-31.

[36] "WebSocket API." https://html.spec.whatwg.org/multipage/web-sockets.html#network. Accessed: 2017-10-31.

[37] "Three core problems of dependency management." https://www.thoughtworks.com/mingle/scaled-agile/2015/08/06/dependency-management-problems.html. Accessed: 2017-10-31.

[38] "Go dep - Dependency management tool." https://github.com/golang/dep. Accessed: 2017-10-31.

[39] "Reproducible Builds." https://reproducible-builds.org/docs/definition/. Accessed: 2017-10-31.

[40] "Go Report Card - Static code analysis tool." https://github.com/gojp/goreportcard. Accessed: 2017-11-2.

[41] "Software Life Cycle Processes." https://www.iso.org/standard/43447.html. Accessed: 2017-09-21.

[42] "Analytics & Developments group - CERN." http://information-technology.web.cern.ch/about/organisation/analytics-developments. Accessed: 2017-09-21.

[43] "Program evaluation and review technique." https://en.wikipedia.org/wiki/Program_evaluation_and_review_technique. Accessed: 2017-10-11.

[44] "Gantt chart." https://en.wikipedia.org/wiki/Gantt_chart. Accessed: 2017-10-11.

[45] "Sindicato CC.OO - Tablas salariales PDI FUNCIONARIO y PDI LABORAL 2010." http://sindicatos.uca.es/ccoo/noticias/tablas-salariales-pdi-funcionario-y-pdi-laboral-2010. Accessed: 2017-11-1.

[46] "Macbook Pro Technical Specifications." https://www.apple.com/macbook-pro/specs-2015/. Accessed: 2017-10-27.

[47] "WebDAV - Wikipedia." https://en.wikipedia.org/wiki/WebDAV. Accessed: 2017-10-31.

[48] "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)." https://tools.ietf.org/html/rfc4918. Accessed: 2017-10-31.

[49] "Go Documentation - WebDAV package." https://godoc.org/golang.org/x/net/webdav. Accessed: 2017-10-27.

[50] W. Kennedy, B. Ketelsen, and E. S. Martin, *Go in Action*. Manning Publications, 2015.

[51] A. A. A. Donovan and B. W. Kernighan, *The Go Programming Language (Addison-Wesley Professional Computing Series)*. Addison-Wesley Professional, 2015.

[52] J. Newmarch, *Network Programming with Go: Essential Skills for Using and Securing Networks*. Apress, 2017.

[53] J. Turnbull, *The Docker Book*. 2014.

[54] J. V. Calvellido, "Dpmbox: An interactive user-friendly web interface for a disk-based grid storage system," 2015.

[55] "Google Chrome - SSL settings." http://googlechrometutorial.com/google-chrome-advanced-settings/Google-chrome-ssl-settings.html. Accessed: 2017-10-11.

[56] "Mozilla Firefox - Exporting a p12 certificate." https://support.mozilla.org/en-US/questions/842394. Accessed: 2017-10-11.

[57] "HashiCorp - Vagrant." https://www.vagrantup.com/. Accessed: 2017-10-11.

[58] "Vagrant - Downloads page." https://www.vagrantup.com/downloads.html. Accessed: 2017-10-11.

[59] "Getting Docker." https://www.docker.com/docker-centos-distribution. Accessed: 2017-10-11.

Appendix A

User manual

Last Mile Transfer in its essence is a proxy service that is accessible via WebFTS. In this chapter we will go through all the necessary steps to get a file uploaded from a user's local computer to a remote storage on the Grid.

A.1 Delegating credentials

1. The authentication with the Grid Storage Elements (SE) is done based on certificates.
2. A pop-up window appears when you choose the "My jobs" or "Submit a transfer" tabs (it can be closed by clicking outside the pop-up window).
3. There, you will need to paste the private RSA key of your certificate.
4. The private key will not be transmitted anywhere. It is only used locally (within the user's browser) to generate the proxies needed to have access to the FTS services.
5. You need to export your certificate from your the browser in P12 format:
   - Instructions for Chrome [55]
   - Instructions for Firefox [56]
6. To obtain the private key, you can run in the console:

   Listing A.1: Extracting a private key from a .p12 certificate using OpenSSL

```
openssl pkcs12 -in yourCert.p12 -nocerts \
    -nodes | openssl rsa
```

7. If you need a delegation with VOMS credentials (required to access some types of SE endpoints), you will need to introduce the name of the virtual organization (VO) you belong to. Please note that the VO which are currently suppoted are: atlas, cms, dteam, lhcb, ops, biomed.

8. From the right button where the remaining time for the current delegation is shown, you can remove the current delegation and delegate again (the delegation window will appear).

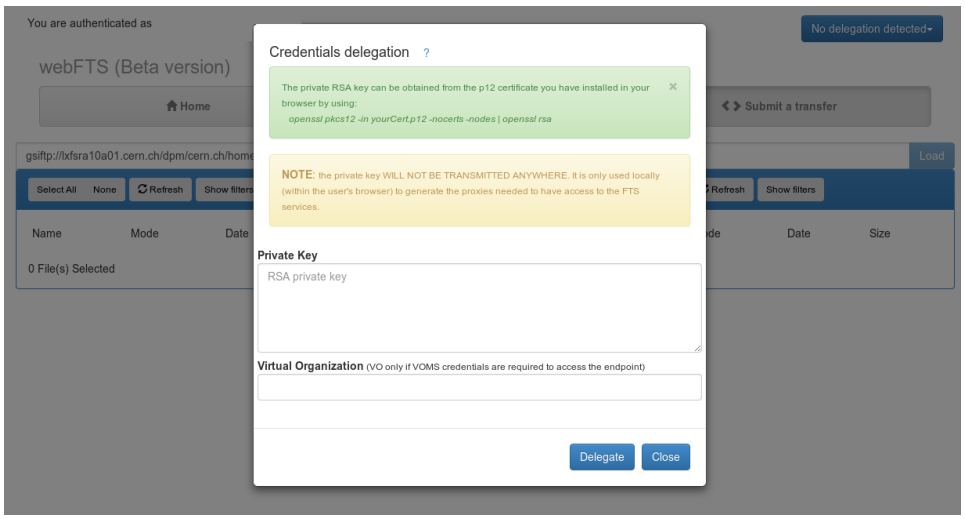


Figure A.1: WebFTS - Credentials delegation

A.2 Submitting a transfer

1. Open the "Submit a transfer" tab.

2. From the drop-down menu on the left, choose "Local", as shown in figure A.2.

3. If one endpoint url is not known, there is an autocomplete option just after 3 characters have been typed. Alternatively, a list of endpoints can be obtained from the infosys (described later). You should also specify the protocol, the address of the endpoint and the path. (Ex: gsiftp://lxfsra10a01.cern.ch/dpm/cern.ch/home/). You need to have the appropriate privileges to access the storage

4. Browse the content and select all the files you want to transfer. CTRL and SHIFT keys can be use for selecting. This is shown in figure A.3

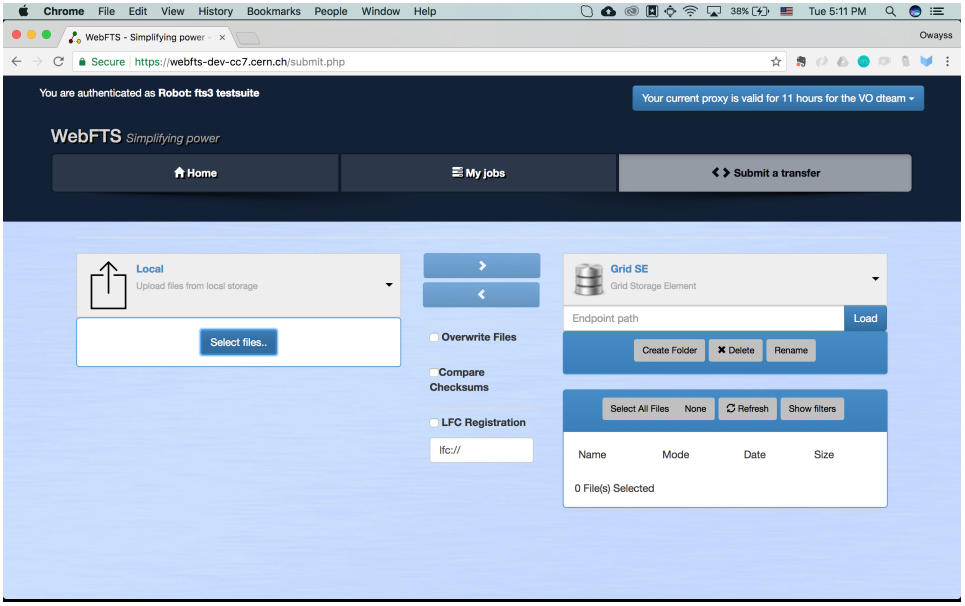


Figure A.2: WebFTS - Upload from local

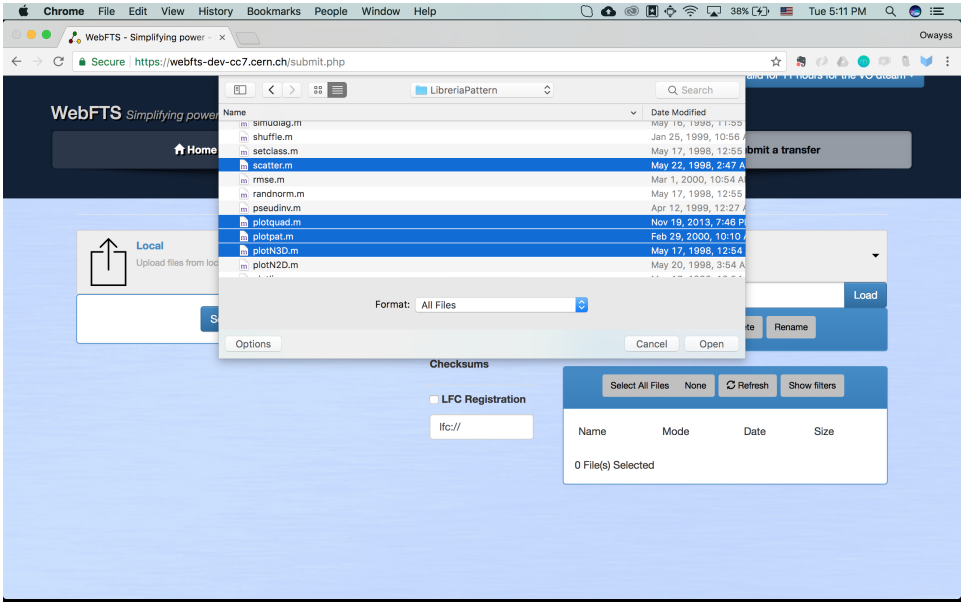


Figure A.3: WebFTS - Choose files to upload from local storage

5. If you need to filter the list of files and folders, you can use the available filters: name, size and date. For the ones allowing ranges, it is not mandatory to specify smallest and largest values. If one is not specified, it will take the minimum (if the smallest value is not speci-

fied) or maximum value (if the largest value is not specified). You can also hide the folders and show only the existing files ("Hide folders" checkbox). You can apply several filters to obtain the final set of files to transfer. The current selection is lost if a different folder is opened.

6. Once you have loaded both endpoints and selected the files, the transfer buttons will be enabled.

7. Click the button with the correct direction. In the case of a local upload, it is the left-to-right transfer button.

8. A success or an error message will appear above the endpoints containers.

9. You can now check the status of you transfer in the "My jobs" tab.

A.3 Listing your transfer jobs

1. Open the "My jobs" tab.

2. If you have done any transfers recently, they will appear there. Their colour will depend on the state. If you click on an individual transfer, you can see its state and any errors.

3. You can resubmit transfer jobs, but you would need to delegate your credentials if you did not do it before (see section 3). You can do this at any state by clicking the "Resubmit" button.

4. The transfer states are: YELLOW if still running, GREEN if successfully completed and RED if something was wrong.

5. The transfers that are not completed have a "Cancel" button on the left of the "Resubmit" button. Clicking this button will cancel that transfer.

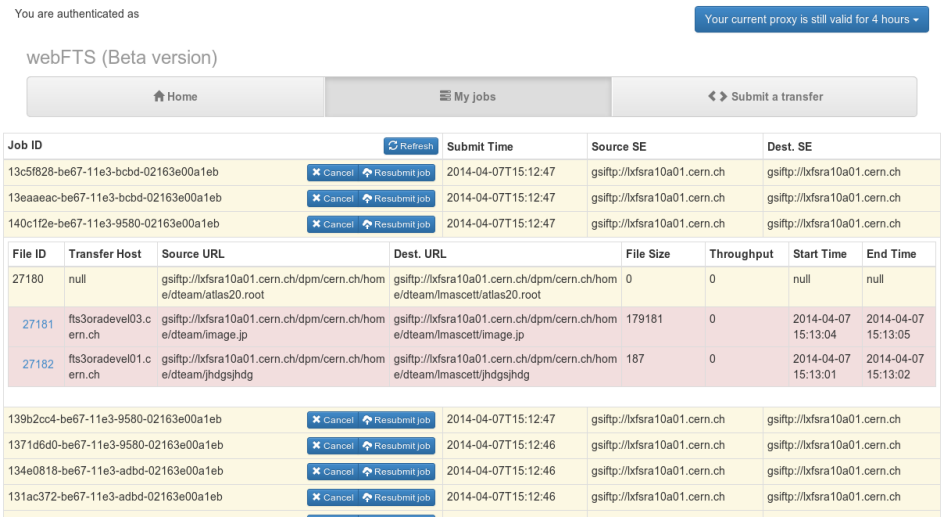


Figure A.4: WebFTS - Listing your transfer jobs

Appendix B

Developer manual

A development environment has been set up using Vagrant [57]. Vagrant is a tool for building and managing virtual machine environments in a single workflow. With an easy-to-use workflow and focus on automation, Vagrant lowers development environment setup time, increases production parity, and makes the "works on my machine" excuse a relic of the past. It contains all the dependencies and configuration files necessary to work on the development of the project.

B.1 Installing Vagrant

Installing Vagrant is extremely easy. Head over to the Vagrant downloads page [58] and get the appropriate installer or package for your platform. Install the package using standard procedures for your operating system.

The installer will automatically add vagrant to your system path so that it is available in terminals. If it is not found, please try logging out and logging back in to your system (this is particularly necessary sometimes for Windows).

B.2 Running the Vagrant box

Once you have `Vagrant` installed, running the vagrant box for the development of the `LMT` project is pretty straightforward:

Listing B.1: Running the vagrant development box

```
# Clone the LMT repo
git clone https://gitlab.cern.ch/fts/lmt.git && cd lmt
# Run the vagrant box
vagrant up && vagrant ssh
```

Appendix C

Guide for system administrators at CERN

C.1 LMT proxy

C.1.1 Standalone deployment

The Last Mile Transfer (LMT) proxy server is written entirely in Go. This means that the server can be compiled to be run as a standalone and a statically linked binary.
To statically build the LMT binary to run on an `amd64` architecture with a linux distribution, use the following command:

Listing C.1: Building a static binary of the LMT proxy

```
CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build -o lmt
    .-ldflags "-w␣-s"
```

C.1.2 Deploying the proxy inside a Docker container

The only requisite for this option is that the host OS must have the Docker daemon installed. If you do not have Docker installed, see [59] for instructions on how you might get Docker running on your machine.

Building the Docker image

Listing C.2: Building a Docker image for the LMT proxy

```
    docker build -t lmt .
```

Running the Docker container

Listing C.3: Running the LMT proxy inside a Docker container

```
docker run -v /host/path/to/certs:/etc/grid-
    security -it lmt
```

C.1.3 Usage

Striaght from the output of the `help` command available in the LMT binary:

Listing C.4: Usage of the LMT binary

```
Usage of ./lmt:
-cert string
      path to the server's certificate in PEM
         format (default "/etc/grid-security/
         hostcert.pem")
-key string
      path to the server's private key in PEM
         format (default "/etc/grid-security/
         hostkey.pem")
-port string
      port to listen on (default "8080")
```

Therefore, to run the proxy server to listen on port 8082 for example, and use the default paths for the TLS certificates, use:

Listing C.5: Launching the LMT proxy server

```
./lmt -port=8082
```

C.2 WebFTS

For the Last Mile Transfer (LMT) solution to function correctly, WebFTS needs to know the address for the LMT proxy to contact. This is done by providing the `lmtWebsocketEndpoint` and `lmtHealthCheckEndpoint` parameters in the `config.xml` file located in the root directory of the WebFTS repository.
Below is an example configuration file:

Listing C.6: Example config.xml file for WebFTS

```
    <config>
    <ftsAddress>https://webfts.cern.ch:8443</
        ftsAddress>
    <lmtWebsocketEndpoint>wss://lmt.cern.ch:8080/
        socket</lmtWebsocketEndpoint>
    <lmtHealthCheckEndpoint>https://lmt.cern.ch:8080/
        health-check</lmtHealthCheckEndpoint>
    <jobToList>100</jobToList>
    <endpointListUrl>https://webfts.cern.ch/
        endpointList</endpointListUrl>
    <proxyCertHours>12</proxyCertHours>
    <cernboxBaseUrl>root://eosdevbox.cern.ch/eos/
        devbox/user/</cernboxBaseUrl>
    <VOs>
        <VO>atlas</VO>
        <VO>auger</VO>
        <VO>biomed</VO>
        <VO>bitface</VO>
        <VO>cms</VO>
        <VO>dteam</VO>
        <VO>lhcb</VO>
        <VO>ops</VO>
    </VOs>
</config>
```

Appendix D

GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML,

PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to

the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant

Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `https://www.gnu.org/licenses/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the

Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.