# 1. PUBLISHABLE SUMMARY

The ultimate goal of the ATOMICDFS project was to investigate the existence of **efficient**, **strongly consistent** distributed file systems in **asynchronous environments, where fail-prone processes communicate via exchanging messages**. The desired strong consistency is **atomicity**, which provides the illusion of a single sequential storage space, when in reality multiple operations may access the storage space concurrently.

The project focused in three main objectives: (1) Development of efficient algorithms for **atomic file objects**, (2) File manipulation techniques to **minimize communication overhead**, and (3) Implementation of the proposed solutions in **simulated and real-life environments**. To meet the project’s goals, the fellow performed a detailed review of the literature, developed a number of algorithms, established new distributed object definitions, proved theoretical results, and implemented some of the proposed algorithms in simulation and real-life environments.

To **address objective (1)** we first identified the weaknesses of current atomic consistency algorithms to support file objects. Solutions failed to provide clear guarantees on the evolution of the file contents, allowing operations to overwrite recent changes on a single file object. To specify the exact guarantees that an atomic **versioned-object** (such as a file object) must posses when operations modify or retrieve its value concurrently, we defined a new property for concurrent objects, called **cover-ability**. Coverability specifies precisely the evolution tree of a versioned object, leading to the creation of a single path from the initial version of the object to any written version of that object.

In our quest to define coverability, we determined that the performance of eixsting atomic algorithms could be improved. In particular, we proved that read operations, as proposed by the most efficient atomic algorithms, are **computationally hard**. During the project we managed to introduce a **new algorithm** that **reduces computation demands** while preserving consistency. Since the proposed algorithm imposes some limitations on the participation of the service, we further developed our ideas and introduced **four (4) more algorithms** that relaxed those limitations, with a minimal impact on operation latency. As we proved that some system **conditions impose specific restrictions on operation latency**, we claimed the **optimality of our algorithms** in terms of communication demands.

To **address objective (2)**, we considered two file fragmentation techniques to reduce the communication costs: (i) **simple division into blocks**, and (ii) **division into encoded blocks** using erasure coding approaches. When considering (i), a file object can be defined as a chain (list) of block objects. An **atomic file object** is the **composition** of a list of **atomic block objects**. Therefore, we use atomic block object implementations to introduce an atomic file object. A file object supports two main operations: (a) a retrieve operation, and (b) a modify operation. A modification on a file attempts to atomically **write()** any block chain that has been created due to the change. A retrieve on the other hand, atomically reads the blocks that the file consists of.

Block fragmentation offers communication benefits but it does not offer storage efficiency: every storage host needs to store a copy of all the blocks for a single file, and thus the whole file object. To improve storage utilization, we examined the use of erasure codes. Erasure codes allow the division of a file object into *n* different encoded blocks (eblocks), out of which only *k<n* eblocks are needed to reconstruct the file. This allows *n* storage hosts to maintain a single eblock for each file, and *k* hosts can be used to reconstruct the file. We discovered (through literature review and own investigation), that atomicity can be achieved using erasure codes under **restrictions either on the concurrency of operations, the number of storage host failures, or when allowing unbounded storage** to be used. In addition, erasure codes require an **additional computation overhead** for the encoding and decoding of the file. We established that further examination on the construction of the coded elements must be performed, to allow efficient manipulation of particular eblocks, and hence benefit atomic implementations.

As a last strategy, we examined the use of journaling (as used in journaling file systems) to perform read and write operations in a single file object. Here, we do not transfer the file data but we keep a **record of the changes** made on a file. Hence, we use a circular log, we refer to as **journal**, that is consolidated periodically to produce the file contents. The journal is treated as a single atomic object, whose small size allows operations to inflict a small footprint on the communication and the storage cost.

To complement the theoretical results and **address objective (3**), we implemented the proposed algorithms in **NS3, a network simulator**, and in the real-life environment **EC2, Amazon’s elastic cloud**. In particular, 6 algorithms were developed for NS3 and 2 algorithms were developed in EC2. Experimental results showed that the proposed approaches outperform previous solutions. We also developed a preliminary working prototype of the journaling distributed file system. The prototype, is based on an architecture of a distributed journaling system we developed during the project.

The socio-economic value and the scale of information increases day by day, and it is becoming imperative to ensure not only that digitally stored data endures device failures, but also that it is readily available, reliable and, above all, consistent. The algorithms developed by ATOMICDFS provide the means of **minimizing the cost (both communication and computation) for atomic object implementations, demonstrating that consistent storage systems can be practical.** In addition, the project proposes solutions to allow the **manipulation of large shared objects (such as files).** The outcomes of ATOMICDFS contribute towards the production of practical file systems on top of cheap commodity hardware, providing strict, provable consistency guarantees. Therefore, the project makes a big step towards **highly reliable, highly consistent, highly collaborative, practical, and global, distributed storage systems**.

The results presented in this work pave the way and place Europe in the frontend of a new era of collaborative applications, which may take advantage of a highly reliable and consistent shared storage to carry out complex tasks. Thus, we can also claim that the project makes a small but decided step towards a future global computing platform. ATOMICDFS places Europe amongst the worldwide leaders in this research area.

The outcomes of ATOMICDFS, along with publications, and source code of the implementations can be found in the website: <https://atomicdfs.networks.imdea.org/>.