information

Article

File System Support for Privacy-Preserving Analysis and Forensics in Low-Bandwidth Edge Environments

Aril Bernhard Ovesen [1,*], Tor-Arne Schmidt Nordmo [1], Håvard Dagenborg Johansen [1], Michael Alexander Riegler [1,2], Pål Halvorsen [2,3] and Dag Johansen [1]

[1] Department of Computer Science, UiT The Arctic University of Norway, 9037 Tromsø, Norway; tor-arne.s.nordmo@uit.no (T.-A.S.N.); havard.johansen@uit.no (H.D.J.); michael@simula.no (M.A.R.); dag.johansen@uit.no (D.J.)
[2] Holistic Systems Department, SimulaMet, 0164 Oslo, Norway; paalh@simula.no
[3] Department of Computer Science, Oslo Metropolitan University, 0130 Oslo, Norway
* Correspondence: aril.b.ovesen@uit.no

Abstract: In this paper, we present initial results from our distributed edge systems research in the domain of sustainable harvesting of common good resources in the Arctic Ocean. Specifically, we are developing a digital platform for real-time privacy-preserving sustainability management in the domain of commercial fishery surveillance operations. This is in response to potentially privacy-infringing mandates from some governments to combat overfishing and other sustainability challenges. Our approach is to deploy sensory devices and distributed artificial intelligence algorithms on mobile, offshore fishing vessels and at mainland central control centers. To facilitate this, we need a novel data plane supporting efficient, available, secure, tamper-proof, and compliant data management in this weakly connected offshore environment. We have built our first prototype of Dorvu, a novel distributed file system in this context. Our devised architecture, the design trade-offs among conflicting properties, and our initial experiences are further detailed in this paper.

Keywords: edge computing; privacy preservation; artificial intelligence; file systems; machine learning; digital forensics

Citation: Ovesen, A.B.; Nordmo, T.-A.S.; Johansen, H.D.; Riegler, M.A.; Halvorsen, P.; Johansen, D. File System Support for Privacy-Preserving Analysis and Forensics in Low-Bandwidth Edge Environments. *Information* **2021**, *12*, 430. https://doi.org/10.3390/info12100430

Academic Editors: Lorenzo Carnevale and Massimo Villari

Received: 17 September 2021
Accepted: 13 October 2021
Published: 18 October 2021

1. Introduction

Numerous Internet of Things (IoT) devices are being deployed in geo-distributed locations far outside traditional computing facilities [1,2]. Examples include video surveillance cameras, home security devices, activity trackers, logistic tracking devices, and smart factory equipment. High velocity, high volume, and heterogeneous data are continuously produced by these devices at an unparalleled scale. A key challenge is to analyze and obtain trusted, timely insight from these data streams.

Distributed system architects must carefully consider structuring alternatives to centralized on-premise or public cloud services for data analysis. Moving computations closer to data sources is likely a better option than federating and centralizing all this data [3,4]. Hence, edge computing can supplement a two-tier centralized architecture by providing an additional computing infrastructure closer to the data sources, residing between IoT devices and centralized back-end services. Edge computed data can still be federated for further analysis and storage at the central locations, but now pre-processed, filtered, and transmitted in reduced volumes.

Edge devices can produce data that are too voluminous to transmit over edge networks, and too heterogeneous to adhere to a unified set of data management rules. For the purpose of tackling this challenge, we introduce the Dorvu file system, a storage system that can be extended with policies and fine-grained access control to solve problems of bandwidth, privacy, and compliance.

The Dorvu file system is part of the larger Dutkat [5] framework, comprising a distributed Artificial Intelligence (AI) hybrid cloud and edge system. This system is motivated by the need for sustainable harvesting of resources from the sea, including commercial fisheries in the Arctic Ocean. The world's global population depends on food obtained from the sea, and this dependence is growing [6]. This can become problematic because the global sea ecosystems have been and are currently under serious attacks by human activities and might not be able to meet this growing demand. Challenges include overfishing and depleted fish stocks, destroyed or polluted sea ecosystems, increased water temperatures, and lack of management and control regimes for sustainable fisheries. According to the United Nations Office on Drugs and Crime, fishery crimes are frequently transnational and organized in nature, and include illegal fishing, document fraud, drug trafficking, and money laundering [7]. As a result, several governments have proposed surveillance systems to track the activity of workers on fishing vessels [8–10], which has been met by some with criticism and claims of privacy intrusion and mass surveillance [11]. The Dutkat framework aims to provide some of the proposed sustainability benefits [12], while preserving the privacy of a fishing vessel crew.

In this work, we are addressing and presenting some specific parts of the envisioned Dutkat system. Specifically, the main contributions are (1) a system of how to perform privacy-preserving analysis of continuously produced data are conceptualized, incorporating challenges such as potentially poorly connected fishing vessels moving about in the Arctic Ocean, (2) we show how multi-sensory data are handled, and we showcase how to use the Dorvu file system to alleviate decision-making around issues like what type of data to store, in what format, and how to mandate access control and encryption on it. Overall, we present an alternative approach to existing surveillance programs [9,10] by replacing human inspection of video footage with a combination of automated processing of sensor data, and access control policies enforced on the storage layer at the time of data creation. We conjecture that incorporating Dorvu in this process better preserves the privacy of people working in proximity to edge sensors, through flexible, fine-grained data access, and storage policies that can retain sustainability-relevant data while discarding privacy intrusive footage, resulting in a less invasive system.

The rest of this paper presents the motivation behind our edge analysis system, the architecture of our cloud and edge hybrid storage system, and the implementation of a prototype of the Dorvu file system. Particular focus is on edge nodes and our design choices targeting a distributed file system that tolerates failures, survives adversarial attacks, and meets compliance requirements.

2. A Mobile AI Edge System at Sea

The Dorvu storage system is intended to serve as a storage layer in the larger Dutkat [5] project revolved around monitoring professional fishing activities in isolated, offshore areas, while retaining the privacy of those working in close proximity to areas subject to surveillance. Its design involves installing robust and safe monitoring devices and related software on board commercial fishing vessels with licence to fish in Norwegian parts of the Arctic Ocean. Each such vessel has been granted a specific quota from the government detailing the amount of fish allowed to catch, the fish species, fish sizes, and similar.

Video surveillance of fishing vessels to combat sustainability issues and enforce fishing quotas has been proposed by some governments [8,9] and explored by others [10]. The Dutkat system is designed to provide some of the sustainability benefits claimed by the proposed national surveillance systems, without infringing on the privacy of workers on fishing vessels.

2.1. Geo-Distribution

The overall architecture of Dutkat reflects the widely distributed and mobile nature of this application domain. We will first detail the horizontal dimension of the architecture, which reflects physical distribution among three separate components: (1) one or several

back-end control centres, (2) a collection of traditional computers on board each vessel, and (3) IoT devices primarily located outside on ship decks. We consider each of the participating large fishing vessels as individual edge nodes in a hybrid cloud computing system, where each edge node has sufficient power facilities and indoor space for deployment of a collection of computers. These computers will for security and fault-tolerance be configured to only run local Dutkat communication, storage, and analysis software parsing locally produced data from the IoT devices outside. Consider such a configuration as implementing certain features of a digital version of a local fishing inspector, which will act as an algorithmic intermediary between the IoT devices on deck and the back-end centralized control centres on mainland.

The Dutkat software must be safe-guarded and stable for 24/7 operability, while at the same time ensuring compliance and non-invasion of the daily operations of the vessel crew. Particularly, AI analysis performed at the edge must be able to detect local anomalies and activities, and consequently persist the relevant ground truth data locally while sending relevant insights to the mainland operational centres for further analysis. Data persistence and analysis being performed locally aid in preserving the privacy of the vessel crew.

A hybrid architecture is needed, with edge nodes connected to centralized structures. The problem at hand is complex and requires input by more than just insights from a single fishing vessel. For instance, one algorithmic trigger that requires input from several edge nodes is the comparison of reported catch from different fishing vessels in the same offshore proximity. Anomalies can be detected through such comparisons, one example being vessels reporting disproportionate amounts of fish caught relative to other vessels in the same area and their allocated quota.

A main problem in this domain is connectivity, since digital communication between these mobile vessels and mainland operational centres is primarily facilitated by satellites. We conjecture that, by moving computations closer to the data sources, the amount of data needed to transmit over satellite links can be reduced to a practical level. This is enabled by edge computing where local data filtering, analysis, and storage can be carried out in real-time.

Evaluating and filtering data streams close to their sources are well-known concepts for scaling distributed systems producing large quantities of data [13]. By this upstream evaluation structuring approach, algorithms can parse and analyze entire streams of data on the vessels, without adhering to the limitations of low-bandwidth satellite links.

2.2. Vertical Distribution

The vertical dimension of the Dutkat architecture determines separation of concerns at the individual horizontal components. The relationship between computers running Dutkat software at back-end control centers and on the edge is illustrated in Figure 1. Overall, (1) a persistent storage layer is in the bottom, (2) followed by a data transfer layer, (3) a data consumption layer, and finally (4) a user interface layer. For edge deployments, the interface layer can be omitted, and IoT devices can interface with the storage system as data producers, as illustrated in Figure 1. The data storage layer will be further detailed in Section 3.

2.3. Multimedia Data Pipeline

Figure 1 shows the generic distribution of data and its relation to the software components in Dutkat. Specifically, the system is deployed to store and transmit multimedia data, like video and images. Edge nodes generate heterogeneous multimedia data, which can vary in content, type, and sensitivity level (i.e., the amount of private information contained in the data). For example, a collection of recordings from an edge video surveillance system may vary in sensitivity level if only a subset of the collection contains footage of people. Similarly, users responsible for generating data may have consented to different data sharing policies, while still contributing to the same dataset. At the same time, data consumers may have varying rights to view this data. In the scenarios presented in Section 1,

it may vary what data local law enforcement, fishing crew, and other interested parties may consume. This results in a system of several edge nodes collaborating to produce a multimedia dataset, consumed by several nodes, both in cloud and edge deployments that differ in their rights to view various parts of the whole dataset. Privileges can be enforced throughout the dataset by applying fine-grained access control mechanisms on individual files in the set.

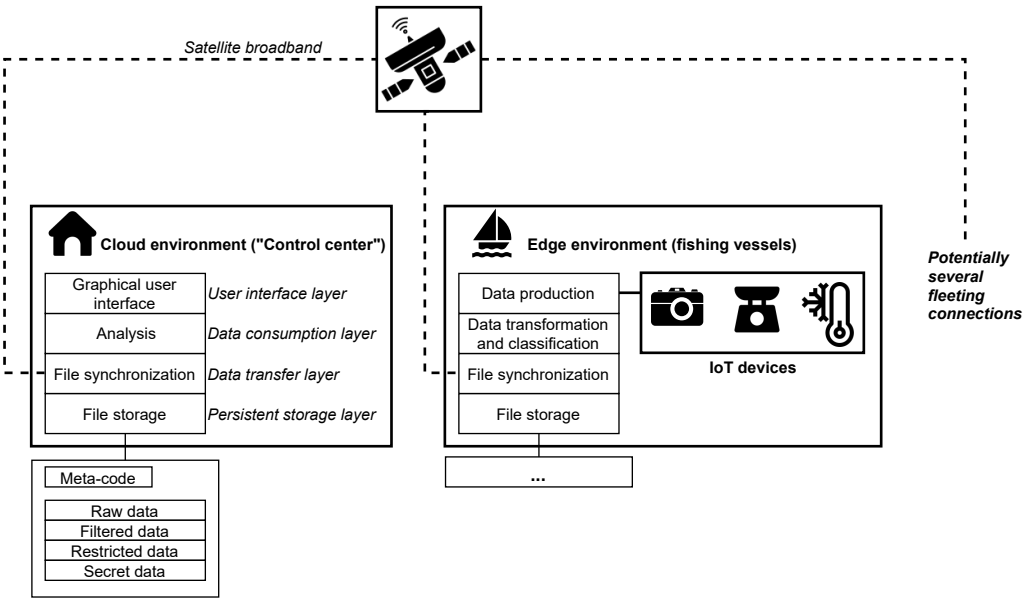

Figure 1. The horizontal and vertical distribution of data, software components, and devices in the Dutkat architecture.

Our system facilitates real-time decision-making from land observers, based on data generated from edge devices. This is a process that involves transmitting as much meaningful data as possible from edge nodes to land nodes. This process is restricted by the low bandwidth that is expected in edge environments, such as when using satellite communication. Dutkat and the Dorvu file system are designed to support a model of decision-making in which the semantic meaning extracted from multimedia recordings is of highest priority to expend bandwidth on transmitting, and the ground truth data, i.e., the full-sized media files, is of less importance and may be retrieved eventually, when the network tolerates it. An example of the extraction of meaningful data from a larger multimedia file is shown in Figure 2. Here, the smallest significant data item extracted from a video file is the acknowledgement of the existence of a file, representing an event. Extracted metadata and still images from the video provide greater detail at the expense of more bandwidth. Finally, the original video file gives an overview of the entire event but is not available to all parties.

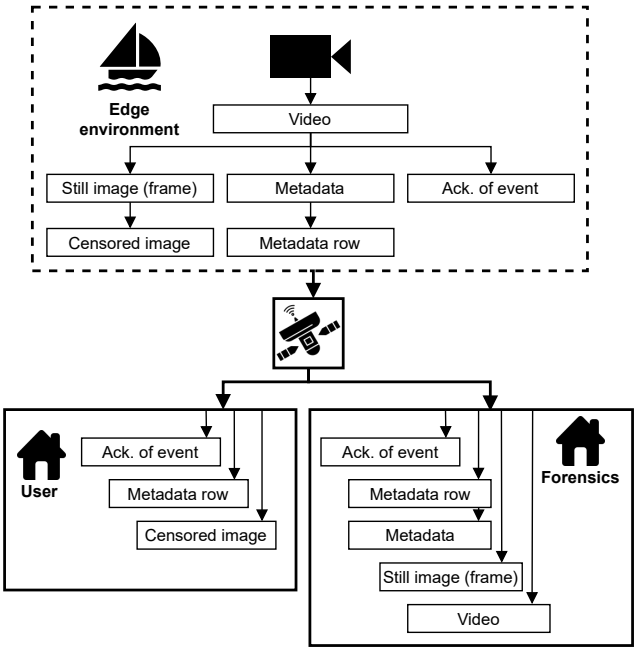

Figure 2. Example of a multimedia data pipeline transmitting information extracted from a video recording from an edge device, to two parties with differing privileges to view the data.

3. System Properties and Architecture

The overall architecture of our distributed file system is based on a central hub structure with a cluster of file servers connected with multiple mobile edge nodes, each with local file storage capacity. This resembles a client–server star network with clients on the edge perimeter and servers in the centre. Another resemblance is with the distributed file system Coda [14] with back-end server clusters supporting numerous light-weight personal computers. Of particular interest is the support Coda has for disconnected operations, a situation still plausible in an offshore environment.

The centralized server hub is resource-rich and can use existing public cloud file systems supporting efficient, reliable, and centralized storage of multimedia data, sensor data, and machine learning results from the edge nodes. The edge nodes are less resource rich and are psychically located on active fishing vessels. When along the coast and near the shore, communication options include cellular networks and radio networks, but when more distant and offshore, satellite communication is the main option. Novelty in our work is primarily related to the edge nodes, and how and what they communicate back to the mainland-located servers.

3.1. System Requirements

There are special application-specific demands that need software tailoring and customization. The file system is intended for use in an area with very limited computational and communication resources since its mobile edge clients consist of fishing vessels moving about in large ocean areas. Communication in such a widely geo-distributed mobile edge computing environment is through partially disconnected, low-bandwidth polar orbit satellite links. Since the fishing vessels in our system operate in the far north of the globe, the geo-stationary satellite solutions we previously utilized are not adequate as they do not cover the northernmost hemisphere [15]. Add to the complexity that this distributed file system must be scalable, secure, fault-tolerant, and compliant with particularly the EU General Data Protection Regulation (GDPR) privacy regulations [16].

Special properties of the select problem domain motivate the design of our system as follows. First, we need to be able to continuously capture and store video and sensor data for fish management, control, and forensics purposes. An example is a continuously captured surveillance video of equipment stored on the deck of a fishing vessel. This is

a resource demanding file storage challenge, and the file system should be used to store video sequences when specific activities or events are detected, and apply access policies based on the contents of the events. Video data are in any case high-volume data, which is challenging to reliably transmit to mainland operational surveillance and control centers from the offshore mobile fishing vessels. The bandwidth delivered by satellite-based solutions is not adequate to support live video streaming, even more so for networks based on the low-frequency L-band.

Next, we need to provide redundancy for fail-safe storage of vital data, through data copies at multiple local disks. The replicas are physically distributed on board each vessel to reduce the probability of data corruption, loss, or unavailability. Redundancy and update techniques similar to the Google file system [17] are adopted, with a master control node typically administering three replicas updated in a pipeline fashion. The master node will raise a flag, i.e., transmit a signal to the mainland operational centre, if the replication threshold is below a certain level. Additionally, since the master node might be a single point of attack or failure, we provide primary-backup replication with a hot stand-by node ready to take over. Hot stand-by implies that a sequence of the latest data written to disk is kept in main memory. Consider this as a large ring buffer whose content will be streamed to disk upon fail-detection and fail-over.

The Google file system and similar master-based distributed systems [18] do not provide such a replication due to complexity with consistency, impact on cost, and performance, the deployment in a trusted enterprise environment, and the observation that master node failures seldom happen. In our case, executing on the edge in a less trusted environment, we cannot tolerate a single node failure weakness in the critical data storage path.

The hot stand-by keeps a large enough sliding window of data to be backed up in case of failure so that a primary node failure will be transparently handled. Notice that inconsistency problems among the data storage replicas or master replicas are to a large extent avoided since data to be permanently stored on the edge nodes are tagged with its timestamp and is immutable. This way no read-write conflict will appear.

3.2. Data Classification

Fail-safe storage implies that there are very strict access policies affiliated with some of this data. We must therefore distinguish between and classify data according to different compliance, safety, and liveness properties. That is, we provide different guarantees with regard to data reliability, availability, privacy, and confidentiality based on how the data are classified. Data must be classified as either `RAW`, `FILTERED`, `RESTRICTED`, or `SECRET`. As will become apparent, this classification differs from traditional security classification schemes, as it supports implementation of non-functional aspects other than security. This classification and its various properties are explained in the following paragraphs.

Data tagged `RAW` contains the continuous stream of data produced by video cameras and sensor devices. Select crew members on board the vessel where the data are produced can gain access to this data in real-time. This can be through real-time streaming to display monitors, or it might be made available as a configuration option if some of these data are persisted on local disks. No encryption of the data is mandated, and only privacy-preserving aspects must be handled. This might involve the fact that vessel crew members grant consent to store and access this data, or that software applies masking of any personally identifiable characteristics.

`FILTERED` data consist of processed select `RAW` data that can be persisted to disk and/or used as input to local analysis applications. Such `FILTERED` data can for instance contain a video sequence with human activities, or sequences of video captured upon other sensory input. Depending on its content, it can be enforced that these data are modified before persisting to disk, in order to be used in analysis applications.

Data tagged `RESTRICTED` is not accessible by any of the crew on board the vessel and is intended for surveillance operations. These data contain specific results obtained from

local edge analysis software processing either `RAW` or `FILTERED` data. This classification also indicates that stricter access policies need to be applied, since data might be annotated with additional information from analysis software and because it is expected to be transmitted over network to a central control center at some point. Examples of this type of data include output from edge located machine learning applications analyzing activities at the fishing vessels, select I-frames from specific surveillance videos, and other sensor data detecting for instance amount of fish caught, fish types, their average size, and relative distribution among species.

The purpose of this data classification is to provide context for central control centres, and extract semantic meaning from a larger data set generated at edge nodes. This serves two purposes: (1) reduce the amount of data transmitted, to support lower bandwidths, and (2) transmit as little data as required to provide meaning and perform forensics. This is to apply a principle of minimal privilege of access to parts of a live surveillance feed, as only the data deemed necessary to provide ground truth to some observation or detected event will be transmitted from the edge node.

The goal of providing these data overlaps with a goal of the overarching Dutkat system [5], which is to provide a probabilistic and evidence-based approach to inspection of fishery activities, and to shield crew members from being subjected to continuously transmitted surveillance.

`SECRET` data are a complete log of RAW data that is encrypted upon storage and persisted in a highly fault-tolerant manner. These data must be stored locally on the edge nodes due to its sheer volume, and access to it is mandated by very restricted access policies. Notably, `SECRET` implies that nobody on board the fishing vessel might access it, and the data are immutable and cannot be altered or deleted. These data can optionally be pre-processed upon storage, blurring out personal identification characteristics. The data can only be accessed and decrypted by a trusted third-party with legal, explicit authorization to do so. This can be a fishery inspector or other forensic parties inspecting a vessel with access permissions according to existing laws and regulations.

In general, we build this distributed file system adhering to the proportionality principle in a legal context striking a balance between human privacy rights and the claimed sustainability benefits of video and sensory surveillance of fishing waters [12]. Invading surveillance on a physically limited area as a trawler deck impossible for the vessel crew to avoid might violate privacy principles well grounded in constitutional, national, and international laws. One example of this is that people in a video sequence can be personally identified while working.

4. Implementation Details

The data plane described in Section 2.2 is implemented by the Dorvu filesystem, to achieve support of heterogeneous data formats and pre-existing tools for analysis and surveillance, through POSIX-compatibility. In this section, we detail some of the implementation details of our prototype.

Customization and adaptability are core aspects of the architecture, where the software aims to provide a basic layer of traditional data storage, with the possibility to interposition and add extra functionality modules between applications and the file system storage. We refer to the modules attached to a file as its *meta-code*, similar to the work done in [19]. This provides a means for transparently adding custom software modules in the critical path of data storage.

The version of Dorvu implemented for this paper includes (1) encryption as a module between the user and disk, (2) file versioning based on user access rights, and (3) an interface for a user to configure the encryption module and access control. Additionally, we investigate the performance overhead of this functionality and the userspace file system platform.

Interfacing with the Dorvu file system can be listed in three steps, beyond reading and writing as if to a local and un-encrypted file system: (1) Users can define access rights for

their own files, indicating identities with public key signatures; (2) When a user creates a file, a corresponding configuration file is created by Dorvu. This defines the available versions of a file to the members of listed access groups; (3) When writing to a file, its file extension, referenced in its corresponding configuration file, decides what access control and encryption the file system will apply to the newly written data.

4.1. FUSE

Dorvu is implemented as a File System in Userspace (FUSE) application [20]. FUSE is a library and Linux kernel module that enables user-level programs to function as mountable file systems, by calling the FUSE kernel module via the FUSE userspace library. In short, a FUSE daemon can service Linux Virtual File System (VFS) calls despite running with userspace privileges. Dorvu is implemented with the Rust programming language, using the `Fuser` library (https://crates.io/crates/fuser, accessed on 12 October 2021) to interface with the FUSE kernel module. `Fuser` provides a userspace library that is implemented separately from the FUSE reference library `libfuse`.

Dorvu implements storage by mirroring the contents of a directory on a local file system. By using Dorvu while it is mounted to a local folder, the mirrored folder will be populated with internal files and encrypted data files. These files can only be read through Dorvu, or by means of manual decryption.

4.2. File Definitions

Dorvu handles three different types of files internally. Interfacing with Dorvu as a regular file system is done by creating, writing, and reading files. These files of arbitrary content are referred to as *data files* in the context of this implementation. Creating a data file automatically creates an auxiliary *configuration file*. A data file must always have a configuration file in order to be visible in a Dorvu directory listing. This file contains a JSON specification of the different available versions of a file (referred to as *layers* in the file), in addition to a path to the access group definition to use for the corresponding data file. The *group definition file* is the second type of auxiliary file used in Dorvu. Access group definitions in these files are listed in JSON and require a name and a list of public key SHA-256 signatures. Examples of these files are shown in Figure 3.

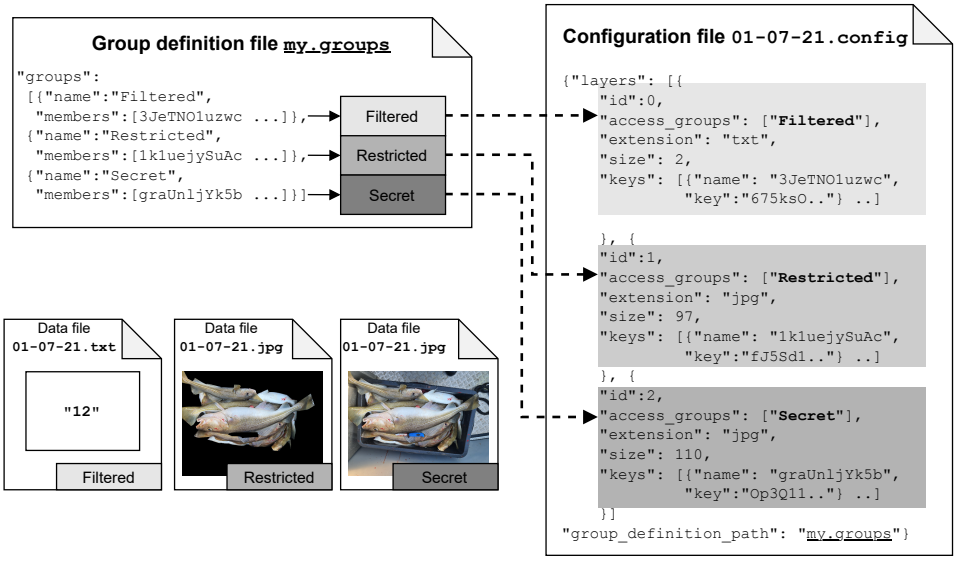

Figure 3. Definition of a configuration file, which defines access rights for a single data file, with its corresponding group definition file. The illustrated data file exists in three versions, one text file and two image files.

Data files are matched to their configuration files by their file stems (i.e., their file names without any directories or extensions). Files are matched to a version defined in this configuration by its extension, which is expected to be an integer matching the *ID* of a layer. When listing the contents of a directory, every configuration file at the mirrored folder corresponding to the working directory will be parsed in order to determine accessibility and file extension. Every file version whose access group includes a given user's identity will be visible in the directory for this user. When multiple versions of the same file have the same file extension, the version with the lowest identity is deemed redundant. If a user has access to several versions of the same file with different file types, both will be listed, as illustrated with a text file (`.txt`) and image file (`.jpg`) in Figure 4. File versions enforced by access control and encryption are a generic implementation of the data classification scheme overviewed in the system requirements in Section 3.2. With adequate meta-code modules and appropriate access group management, the required data classification scheme for Dutkat can be implemented in Dorvu.

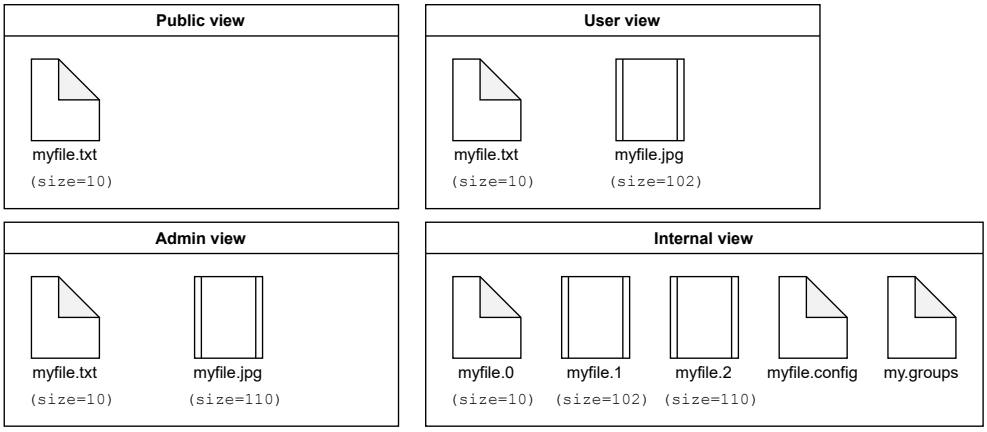

Figure 4. Multiple views of the same directory based on several users' differing access rights, with a scheme similar to Figure 3.

As manually maintaining configurations for every newly created file can be time consuming and increase the chance of human error, Dorvu includes the concept of *configuration templates*. A configuration template is a directory-wide configuration file that is applied to every newly created file. This feature is introduced for ease-of-use; the expected usage pattern of Dorvu is to set up file configurations before any automated data production begins. In that case, the expected access control will be applied to newly created files without the need for manual maintenance.

4.3. Encryption

With the encryption module implemented in Dorvu, files classified as `SECRET` are stored encrypted by default, using the OpenSSL implementation (https://crates.io/crates/openssl, accessed on 12 October 2021) of 128 bit AES-CBC. The AES key for a given file is encrypted with RSA once for each user with access to that file, using their 2048 bit public key. A base 64-encoded version of this encrypted AES key is stored in the file's config, as shown in Figure 3.

5. Experiments and Results

5.1. I/O Speed and Overhead Measurement

We want to gain insight into the potential overhead cost by adding Dorvu as an extra layer of indirection in the critical data path for disk access. This experiment is performed by measuring time taken for read and write operations on various storage back-ends. The test environments are chosen to provide information about the expected sources of performance overhead: the cost of encryption, the cost of file versioning and access control, and the cost of utilizing a FUSE-based file system rather than a kernel-integrated file system.

To observe the impact of encryption on read/write throughput, we deployed two configurations of Dorvu, one with encryption enabled and one with all encryption features disabled. To observe the costs associated with deploying a FUSE file system, we implemented a simple FUSE application that forwards all operations to an ext4 file system, labeled *FUSE passthrough* in our experiment. Our assumption is that the maximum possible throughput of Dorvu will be that of the FUSE passthrough application, and that throughput loss between the FUSE implementation and the ext4 storage back-ends will be outside of the control and scope of our implementation. The difference between the throughput of the encrypted and decrypted Dorvu configurations will indicate the cost of encryption, and the difference between the decrypted Dorvu configuration and the FUSE passthrough application will indicate the cost of file versioning and other metadata operations.

5.1.1. Experimental Setup

This experiment was performed on a desktop workstation with an AMD Ryzen 5 3600 6-Core processor running at 3.60 GHz, and a Kingston UV400 solid state drive storage device, running Ubuntu 18.04. 8 randomly generated files of varying sizes were read and written 10 times per storage environment. These file sizes are chosen due to our use-case of storing media files suited for low-bandwidth network transfer, while we acknowledge that system overhead and inefficiencies are more easily observed during longer operations with larger files. Because of this, later experiments described in Sections 5.2 and 5.3 utilize smaller files.

5.1.2. Results

The results from this experiment are shown in Figure 5. Results show that, for reading our largest files of 64 MB and 128 MB, encryption was the biggest source of overhead. For every other test case, however, the difference between the FUSE passthrough performance and unencrypted Dorvu performance indicate that file versioning and metadata operations are the biggest software bottlenecks in Dorvu. We theorize that this is particularly prevalent during file writes because these operations are split into smaller operations of individual page sizes of 4096 bytes on our test system, resulting in worse performance during writes than reads, relative to the baseline ext4 environment. We further hypothesize that, while performance penalties associated with encryption are expected, file versioning and metadata overhead observed in both read and write operations can be investigated through software profiling, and reduced by further optimization of the file system.

We observe that costs associated with utilizing a FUSE implementation are negligible in many of our observations, particularly during reads because the majority of overhead relative to the ext4 environment is visible in the Dorvu environment, and not in the FUSE passthrough application. It is worth noting that the measurements for both the FUSE and ext4 storage deviate by up to 30% between minimum and maximum observations, but their averages are within each other's standard deviation for every file size in the experiment.

The FUSE passthrough comparison measurement was also performed in [19], but was re-implemented for this experiment due to our adoption of the Rust programming language and its `Fuser` library. A more thorough examination of the performance implications of utilizing user-space file systems is given by Vangoor et al. [21]. They show that the throughput penalty of using FUSE over ext4 can be as low as 5%, but that certain workload characteristics can severely negatively impact this performance.

We conjecture that, for our use-case of optimizing for low-bandwidth transfer, utilizing a FUSE implementation does not significantly negatively impact the performance of Dorvu, while recognizing that a future use-case involving larger file sizes or different workload characteristics may change this outlook.

5.2. Satellite Latency and Bandwidth

Network communication between edge components in Dorvu and the Dutkat system will be provided by satellite broadband. Observations of the capabilities of the available

satellite network are key to designing communication models and delegating tasks to edge and land components in the system. In addition to measuring the sustained average bandwidth provided by the network, we measure the transfer speed of individual files of specific sizes. This is both to emulate file system usage on the network, and to give an indication of the impact of network latency when transferring small amounts of data.

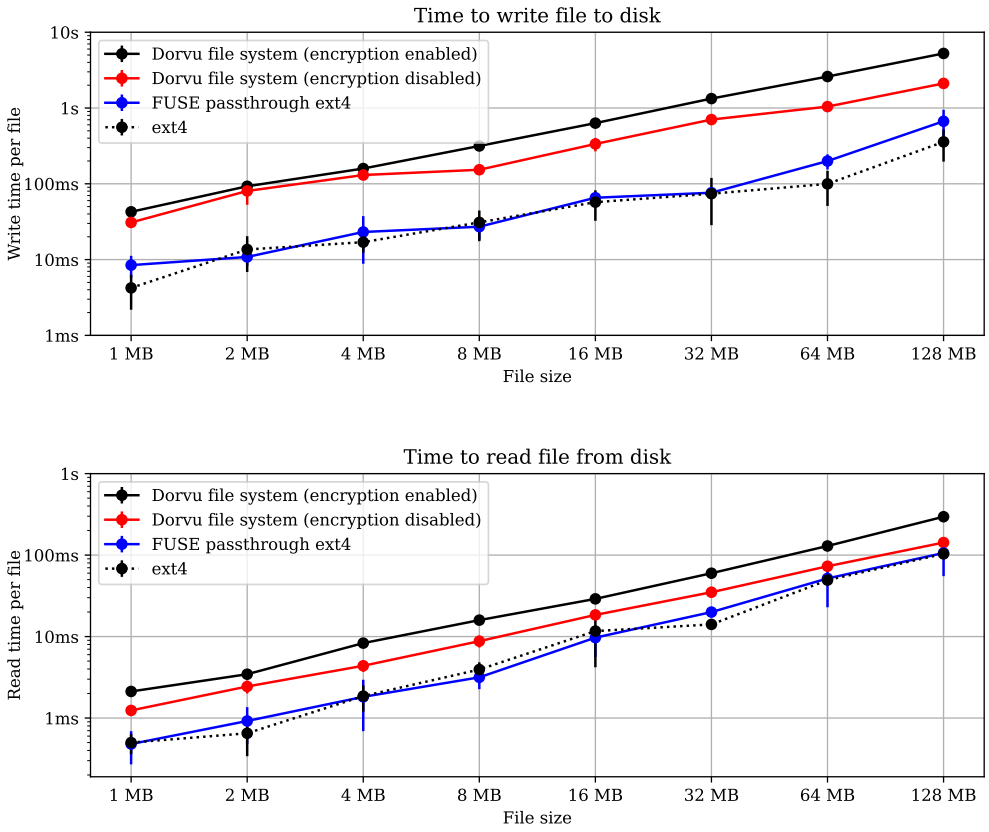

Figure 5. Time measured for read and write operations on varying file sizes and file system configurations. The top graph indicates measurements of the write operation, while the bottom graph indicates the read operation. The *y*-axis represents time spent on an operation, measured in seconds, plotted on a logarithmic scale. The various files used for this experiment are listed on the *x*-axis, represented by their file sizes. Plots include error bars, indicating standard deviation on the *y*-axis.

5.2.1. Experimental Setup

The experiment was performed by using the Linux `curl` command to download files from a test GitHub repository with files generated for the purpose of this experiment. The experiment is performed on the Iridium Certus 200 broadband satellite service, an L-band non-geostationary satellite network claiming global satellite coverage and download and upload speeds of up to 176 kilobits per second [22].

We connect to this network through a Thales Avionics VesseLink 200 broadband terminal, consisting of an antenna and router for maritime use [23]. The experiment was ran on an HP EliteDesk 800G6 workstation with the VesseLink providing its only connected network. The equipment is stationed at the University of Tromsø, Norway and was tested during cloudy weather conditions with drizzle, and each download was repeated five times. This number of downloads was chosen to adhere to restrictions on network resources.

5.2.2. Results

The purpose of this experiment is to observe the capabilities and limitations of the hardware and network available to our system when deployed in the targeted environment and weather conditions. The experiment is not intended as an exhaustive evaluation of

the feasibility of satellite broadband, or the performance of this particular satellite service in broadly defined use-cases, but to explore various network conditions that our system must handle gracefully. These results will also be applied in the experiment described in Section 5.3.

The resulting measurements from this experiment are shown in Figures 6 and 7. Note that, in both of these figures, the *y*-axis follows a logarithmic scale, while the *x*-axis follows no particular scale, rather representing a set of files. The standard deviation on the *y*-axis is plotted as error bars.

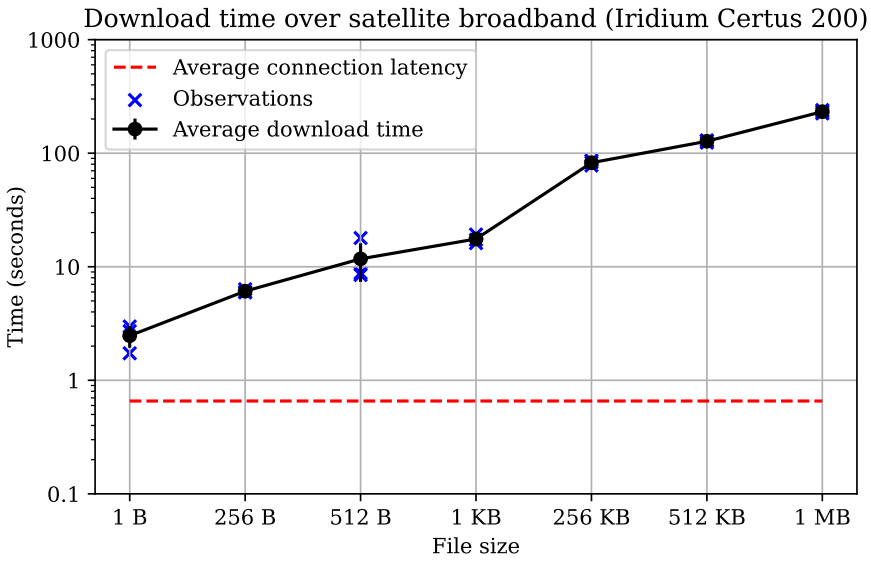

Figure 6. Observed time to download files at various sizes over a satellite broadband connection. Time is represented in a logarithmic scale on the *y*-axis, to show average download times of the various files, represented on the *x*-axis by their sizes. The average connection latency observed throughout every established connection during this experiment is also shown.

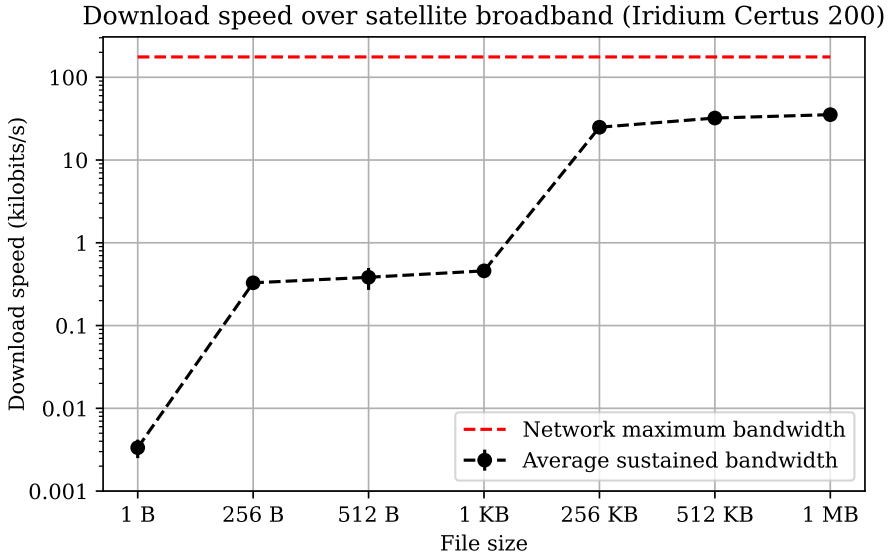

Figure 7. Observed download speed when downloading files of various sizes over a satellite broadband connection. The *y*-axis represents download speed, measured in kilobits per second, of the files represented on the *x*-axis. Additionally, the theoretical maximum network bandwidth as defined by the system provider [22,23] is shown.

Results shown in both Figures 6 and 7 indicate the same expected observation that the average download speed throughout the download process is lower when transmitting smaller files. We assume this is a result of the time of initiating the file transfer connection and cost of transferring metadata are proportionally more significant the smaller the payload. Some storage systems are designed to handle and distribute large amounts of small files specifically, to alleviate weaknesses of existing protocols in this use-case [24,25].

It is observed in Figure 7 that the achieved download speed is considerably lower than the network maximum, which can be influenced by several factors, such as weather conditions, relative satellite location, and network traffic [26].

5.3. Machine Learning Workloads on the Edge vs. a Centralized Hub

We argue that analysis should be performed on the edge to preserve privacy. Additionally, based on our end-to-end satellite communication experiments in Section 5.2, we conjecture that transferring raw video data from the edge nodes to a centralized mainland hub has its performance limitations. We would therefore like to evaluate such a centralized system to see if it is feasible. The typical workload in our fishery use-case is activity recognition based on video data to determine whether e.g., discard of fish has occurred. Therefore, we will test if a centralized system could work based on an activity recognition workload.

In order to evaluate the throughput of a centralized system, we focus on the bitrate required to run inference on videos in real time and compare against the average bandwidth measured for the satellite connection. We send different levels of compressed video data over our satellite connection and identify the top-1 video-level accuracy of models trained on this data. We compress the video by reducing the resolution and/or reducing the frame rate. We aim to see how the compression affects the accuracy of the machine learning model chosen. If the accuracy decreases significantly from the base case (112×112, 30 fps), then it is not feasible to send data over the satellite connection and the inference should be performed locally.

If we use the best average results for bandwidth from Figure 7, we get an average bandwidth of ≈35 `kbps` over the satellite connection. Given that the video files, on average, are much smaller than 1 MB, this is a conservative estimate. The optimal bandwidth is taken from documentation sheets for the satellite router [22,23]. The bandwidth required to send data over the satellite connection should be lower than the average bandwidth measured.

5.3.1. Experimental Setup

In our experiments, we utilize a 18-layer R(2+1)D network, introduced by Tran et al. [27], to perform action recognition. The network was pretrained on the Kinetics-400 dataset [28] and then fine-tuned on the HMDB51 dataset [29]. The video data's resolution and frame rate are reduced to various degrees, while measuring required bandwidth. The network is fine-tuned over 50 epochs and the weights from the epoch which gave the best validation accuracy are kept. This network is then run on a test set giving the final accuracy in Figure 8.

We train on 16-frame clips, as was done during pre-training on the Kinetics-400 dataset. If the frame rate is too low, we repeat the last frame until we fill the tensor. The frames are consecutive and we apply temporal jittering while training. The video-level accuracy is calculated by taking the average prediction of 20 different clips from the same video, and then we choose the top-1 result. The bandwidth required for the different levels of compressed video data was calculated by taking the average bitrate of all compressed videos in the HMDB51 dataset.

We implemented the experiment using PyTorch [30], and the model was trained on an Nvidia RTX 2080 Ti. The model was imported from the `torchvision` module. The frames are extracted from the videos and are resized and combined into a tensor in the batch generator. The frames are augmented randomly using horizontal flips and affine translations before they are normalized according to the means and standard deviations of the Kinetics-400 dataset [31].

5.3.2. Results

We hypothesized that compressing data, in order to adhere to bandwidth restrictions, would lead to lower accuracy for the action recognition model. As we can observe in Figure 8, reducing the resolution results in a dramatic decrease in model performance. Reducing the frame rate also decreases the accuracy, but not to the same degree. The highest possible accuracy we achieve that requires a bandwidth lower than the average bandwidth is at 43.81%, which is much lower than our highest accuracy at 69.3%. Assuming we want a high-performing model, with results as close to state-of-the-art as possible (see Figure 8), we conclude that performing inference on a centralized hub is infeasible. The reason for the discrepancy in our highest accuracy and the accuracy documented in [27] might be due to numerous factors, such as training time, different augmentation schemes, learning rate scheduling, etc.

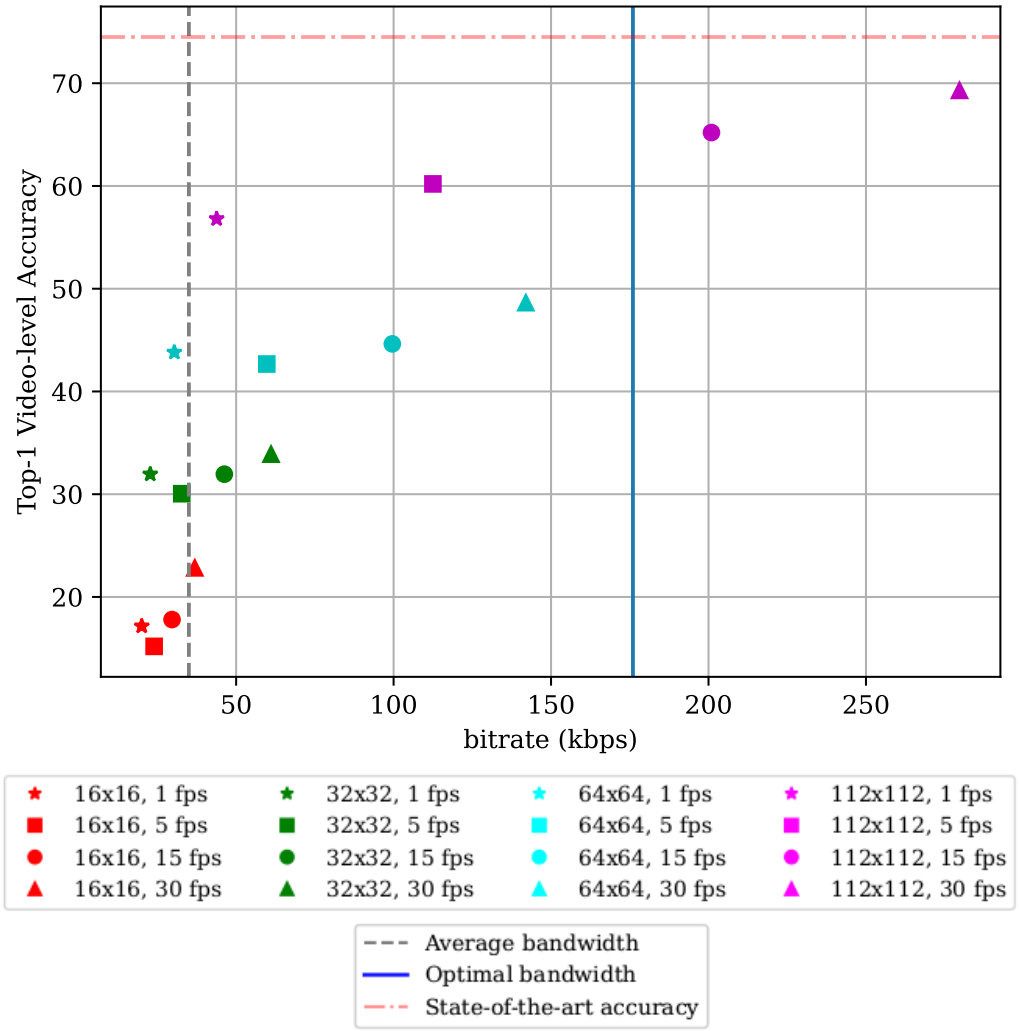

Figure 8. Plot over top-1 video-level accuracy (y-axis) vs. bitrate required to perform real-time inference (x-axis), including vertical lines indicating average and optimal bandwidth possible on our satellite connection. The different colors represent different resolutions of the video data used in fine-tuning the machine learning model, and the shapes are different frame rates. The horizontal line represents the top-1 video-level accuracy achieved by fine-tuning on HMDB51 in [27].

Based on our experiments, upstream evaluation is a more realizable design option for our application scenario [13]. As our system should support real-time monitoring, we will choose the evaluation scheme and inference location based on the capabilities of transferring results from edge nodes in real time. Hence, data should be analyzed

close to its source with inference on video data performed on the edge, on board the fishing vessel where the video camera is located. This design choice also complies with the privacy-preserving design of the system, with the edge nodes performing privacy-critical operations.

6. Related Work

6.1. Privacy-Preserving Surveillance

A surveillance system with built-in video processing and access control based on video analysis was presented in IBM's PrivacyCam [32]. This surveillance system provides cameras with the capability to re-render an input video stream with features such as persons or objects removed. Unedited output streams can be provided for authorized users. Various techniques were applied in similar surveillance systems, for instance by removing distinctive facial features [33], encrypting faces [34], or obscuring people based on specific visual markers [35].

6.2. File Systems

Numerous systems provide encryption as a transparent file system feature or as software on top of a traditional file system. Cryptfs [36] and its successor eCryptfs [37] are cryptographic file systems included in the Linux kernel that provide encryption to files located in local or remote file systems, by storing cryptographic metadata in headers of individual files. Similarly, software like TrueCrypt [38], VeraCrypt [38] and Apple FileVault provide a decrypted view of an encrypted directory mounted elsewhere in the file system.

Several projects provide cryptographic file systems implemented in FUSE. EncFS [39] runs in userspace and mounts to a directory in a local file system and encrypts all data written, storing encrypted versions of these files in a separate location. Gocryptfs [40] is a similar project implemented in the Go programming language with the Go-FUSE kernel bindings library. SecureFS [41] is a C++ FUSE project that aims to provide similar features as EncFs and Gocryptfs to multiple operating systems. Common for these FUSE file systems and Dorvu is that they are implemented as an overlay file system, providing a layer of indirection before writing to a separate local or remote file system. The design of generalized layered file systems and the technology that enables them on various platforms are reviewed and discussed by Zadok et al. [42].

6.3. Extensibility

Architecting extensible software in the offshore domain resembles how we structured our StormCast system [15,43], which further motivated the early mobile agent system TACOMA [44] built for shipping code and state around in a network for remote installation. Our current work utilizes the meta-code concept [19] for extending and customizing remote nodes where remote software can be configured with mobile code.

Our previous Balava file system [45] was built with FUSE and meta-code for managing computations that coupled multiple public clouds together transparently, and involved data with confidentiality constraints. Meta-code as a structuring toolkit is used as in Dorvu, but not in a weakly connected, mobile edge environment. Meta-code is used in Balava for transparently gluing together a hybrid cloud system that interconnects private environments with public clouds such as Microsoft Azure and Amazon Web Services.

6.4. Data Transmission

To avoid transmitting irrelevant and redundant data over the bandwidth-limited links from the remote edge devices to the central cloud-based servers, we aim to apply several data reduction mechanisms. By performing most of the analysis locally, transmission of large amounts of data can be reduced. This is especially important for bandwidth-hungry data types like images and videos. Multiple approaches have been explored for reducing the amount of data generated and for reducing data transmission. For instance, Gurrin et al. [46] propose a system that detects action in images and keeps only images

where action is detected. Ji et al. [47] extract features from both the spatial and the temporal dimensions by performing 3D convolutions, thereby capturing the motion information encoded in multiple adjacent frames. Such approaches are used to reduce data, both for storage, transmission, and later analysis. Further reductions can be achieved reducing image or frame dimensions and sizes without losing important information [48], and analyzing the trade-offs between better quantization and reducing the frame rate [49].

Compression of video data using machine learning is also something we investigated and compared against in Section 5.3. Related approaches include Nvidia Maxine [50,51], a recently developed tool for massive compression of video data for video conferences. This application domain involves videos of faces with typically static backgrounds. It is challenging to apply this approach to our application domain, with video data depicting general activities because it requires large amounts of data to train an equivalent generative adversarial network. Similar video analysis must be performed for privacy, e.g., avoiding to show faces or objects that should for some reason be protected. For example, Fitwi et al. [52] describe a system for masking private information in video frames from surveillance cameras by doing detection and filtering on the edge. Moreover, D'souza et al. [53] describe a similar system that uses object detection for surveillance camera video streams, and whitelists classes of objects that should *not* be censored. Thus, such approaches will both reduce bandwidth, but also provide support for privacy preservation. Neto et al. [54] describe an edge-based system for smart city applications. They describe a system for real-time processing of data that preserves privacy that also utilizes a workload balancer to balance tasks across multiple edge nodes. However, this workload distribution is not applicable for our application, since edge nodes are expected to be physically distant from each other.

6.5. Centralized Data Analysis

We have proposed that the desired rate of data production in our system is larger than the targeted satellite communication link can transfer in real time. Despite bandwidth restrictions, it can still be advantageous to analyze data from multiple nodes and sensors, and potentially in combination with additional data collected from third-parties, such as sales notes and weather data.

Multimodal analysis of data are usually leading to better and more accurate results as recent work shows but comes with additional costs regarding the hardware needed [55,56]. Especially, ensembles of experts models work well with multimodal data streams and complex task analysis [57,58] which makes them a good alternative for the presented use case. For future work, we can use pre-analyzed streams of data that will act as input to an expert ensemble model in which each of the expert sub-networks will focus on learning the specific patterns of that particular data stream.

7. Conclusions

We are developing a geo-distributed, loosely coupled AI system for surveillance of fishing activities in the Arctic Ocean. The development and deployment of this system come with several challenges, due to the nature of the data produced and the targeted edge environment. For example, continuous production of multimedia data requires privacy compliance and fault-tolerance, while the bandwidth of edge networks hinders data transmission and real-time monitoring from non-edge components in the system. We observe that our available satellite broadband networks are not suitable for real-time video transmission for activity recognition, and we propose a system for analysis and data storage on the edge to facilitate this.

We have presented details of a prototype of Dorvu, a geo-distributed file system with support for fine-grained access control policies and software modules. Our prototype demonstrates an implementation of encryption and file versioning based on access rights, and we outline the expected I/O overhead of encryption and metadata operations associated with these capabilities. The deployed version of this system will be spanning edge

nodes on fishing vessels out at sea connected with mainland centralized file servers, in order to utilize a combination of data filtering, analysis, and access control, to serve as a privacy-preserving alternative to manual video surveillance.

Author Contributions: Conceptualization, D.J.; methodology, A.B.O. and D.J.; software, A.B.O., T.-A.S.N. and M.A.R.; validation, A.B.O. and T.-A.S.N., investigation, A.B.O. and T.-A.S.N.; data curation, T.-A.S.N.; writing—original draft preparation, A.B.O., T.-A.S.N., M.A.R., P.H. and D.J.; writing—review and editing, A.B.O., T.-A.S.N., H.D.J., M.A.R., P.H. and D.J.; visualization, A.B.O. and T.-A.S.N.; supervision, H.D.J. and D.J.; project administration, H.D.J., P.H. and D.J.; funding acquisition, H.D.J. and D.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially funded by the Research Council of Norway project numbers 274451 and 263248, and Lab Nord-Norge ("Samfunnsløftet").

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not Applicable, the study does not report any data.

Acknowledgments: We particularly acknowledge contributions from Kim H. Andreassen, Arne E. Karlsen, Nandor Knust, Jon F. Mikalsen, Magnar Pedersen, Jon P. Rui, and Kjetil Robertsen.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Satyanarayanan, M.; Simoens, P.; Xiao, Y.; Pillai, P.; Chen, Z.; Ha, K.; Hu, W.; Amos, B. Edge analytics in the internet of things. *IEEE Pervasive Comput.* **2015**, *14*, 24–31. [CrossRef]
2. Wang, S.; Zhang, X.; Zhang, Y.; Wang, L.; Yang, J.; Wang, W. A survey on mobile edge networks: Convergence of computing, caching and communications. *IEEE Access* **2017**, *5*, 6757–6779. [CrossRef]
3. Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konečnỳ, J.; Mazzocchi, S.; McMahan, H.B.; et al. Towards federated learning at scale: System design. *arXiv* **2019**, arXiv:1902.01046.
4. Zhang, Y.; McQuillan, F.; Jayaram, N.; Kak, N.; Khanna, E.; Kislal, O.; Valdano, D.; Kumar, A. Distributed Deep Learning on Data Systems: A Comparative Analysis of Approaches. *Proc. VLDB Endow.* **2021**, *14*. Available online: http://www.vldb.org/pvldb/vol14/p1769-zhang.pdf (accessed on 12 October 2021).
5. Nordmo, T.A.S.; Ovesen, A.B.; Johansen, H.D.; Riegler, M.A.; Halvorsen, P.; Johansen, D. Dutkat: A Multimedia System for Catching Illegal Catchers in a Privacy-Preserving Manner. In Proceedings of the 2021 Workshop on Intelligent Cross-Data Analysis and Retrieval, Taipei, Taiwan, 12–15 July 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 57–61.
6. Costello, C.; Cao, L.; Gelcich, S.; Cisneros-Mata, M.Á.; Free, C.M.; Froehlich, H.E.; Golden, C.D.; Ishimura, G.; Maier, J.; Macadam-Somer, I.; et al. The future of food from the sea. *Nature* **2020**, *588*, 95–100. [CrossRef]
7. UNODC. *Fisheries Crime: Transnational Organized Criminal Activities in the Context of the Fisheries Sector*; 2016. Available online: https://www.unodc.org/documents/about-unodc/Campaigns/Fisheries/focus_sheet_PRINT.pdf (accessed on 12 October 2021).
8. Ingilæ, Ø. Fiskere settes under overvåkning. In *Kyst og Fjord*; 2020. Available online: https://www.kystogfjord.no/nyheter/forsiden/Fiskere-settes-under-overvaakning (accessed on 12 October 2021).
9. Bizzotto, M. Fishing rules: Compulsory CCTV for certain vessels to counter infractions. In *European Parliament Press Release*; 2021. Available online: https://www.europarl.europa.eu/news/en/press-room/20210304IPR99227/fishing-rules-compulsory-cctv-for-certain-vessels-to-counter-infractions (accessed on 12 October 2021).
10. Ministry of Trade, Industry and Fisheries. Framtidens Fiskerikontroll. NOU 19:21. 2019. Available online: https://www.advokatforeningen.no/aktuelt/horingsuttalelser/2020/mars/horing---nou-201921-framtidens-fiskerikontroll/ (accessed on 12 October 2021).
11. Martinussen, T.M. Danske fiskere samler seg mot kamera-overvåkning i fiskeriene. *Fiskeribladet*. Available online: https://www.fiskeribladet.no/nyheter/danske-fiskere-samler-seg-mot-kamera-overvakning-i-fiskeriene/2-1-839478 (accessed on 12 October 2021).
12. Van Helmond, A.T.; Mortensen, L.O.; Plet-Hansen, K.S.; Ulrich, C.; Needle, C.L.; Oesterwind, D.; Kindt-Larsen, L.; Catchpole, T.; Mangi, S.; Zimmermann, C. Electronic monitoring in fisheries: Lessons from global experiences and future opportunities. *Fish Fish.* **2020**, *21*, 162–189. [CrossRef]
13. Carzaniga, A.; Rosenblum, D.S.; Wolf, A.L. Design and evaluation of a wide-area event notification service. *ACM Trans. Comput. Syst. (TOCS)* **2001**, *19*, 332–383. [CrossRef]

14. Satyanarayanan, M.; Kistler, J.J.; Kumar, P.; Okasaki, M.E.; Siegel, E.H.; Steere, D.C. Coda: A highly available file system for a distributed workstation environment. *IEEE Trans. Comput.* **1990**, *39*, 447–459. [CrossRef]
15. Johansen, D. StormCast: However, another exercise in distributed computing. *Distrib. Open Syst. Perspect.* **1993**. Available online: https://munin.uit.no/bitstream/handle/10037/392/report.pdf?sequence=1&isAllowed=y (accessed on 12 October 2021).
16. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Off. J. Eur. Union* **2016**. Available online: https://eur-lex.europa.eu/eli/reg/2016/679/oj (accessed on 12 October 2021).
17. Ghemawat, S.; Gobioff, H.; Leung, S.T. The Google file system. In Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, Bolton Landing, NY, USA, 19–22 October 2003; pp. 29–43.
18. Dean, J.; Ghemawat, S. MapReduce: Simplified data processing on large clusters. *Commun. ACM* **2004**, *51*, 107–113. [CrossRef]
19. Johansen, H.D.; Birrell, E.; Van Renesse, R.; Schneider, F.B.; Stenhaug, M.; Johansen, D. Enforcing privacy policies with meta-code. In Proceedings of the 6th Asia-Pacific Workshop on Systems, Tokyo, Japan, 27–28 July 2015; pp. 1–7.
20. FUSE—The Linux Kernel Documentation. Available online: https://www.kernel.org/doc/html/latest/filesystems/fuse.html (accessed on 30 August 2021).
21. Vangoor, B.K.R.; Tarasov, V.; Zadok, E. To FUSE or not to FUSE: Performance of user-space file systems. In Proceedings of the 15th USENIX Conference on File and Storage Technologies, Santa Clara, CA, USA, 27 February–2 March 2017; pp. 59–72.
22. Iridium Certus 200. Available online: https://www.iridium.com/services/iridium-certus-200/ (accessed on 8 September 2021).
23. Thales VesseLINK 200. Available online: https://www.thalesgroup.com/sites/default/files/database/document/2021-02/2807_V1_VesseLINK200_012021.pdf (accessed on 8 September 2021).
24. Yu, L.; Chen, G.; Wang, W.; Dong, J. Msfss: A storage system for mass small files. In Proceedings of the 11th International Conference on Computer Supported Cooperative Work in Design, Melbourne, Australia, 26–28 April 2007; pp. 1087–1092.
25. Thain, D.; Moretti, C. Efficient access to many small files in a filesystem for grid computing. In Proceedings of the 8th IEEE/ACM International Conference on Grid Computing, Austin, TX, USA, 19–21 September 2007; pp. 243–250.
26. Gerard, M.; Bousquet, M. *Satellite Communications Systems*; Teubner. 1993. Available online: https://www.amazon.com/Satellite-Communications-Systems-Communication-Distributed/dp/0471971669 (accessed on 12 October 2021).
27. Tran, D.; Wang, H.; Torresani, L.; Ray, J.; LeCun, Y.; Paluri, M. A closer look at spatiotemporal convolutions for action recognition. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6450–6459.
28. Kay, W.; Carreira, J.; Simonyan, K.; Zhang, B.; Hillier, C.; Vijayanarasimhan, S.; Viola, F.; Green, T.; Back, T.; Natsev, P.; et al. The Kinetics Human Action Video Dataset. 2017. Available online: http://xxx.lanl.gov/abs/1705.06950 (accessed on 28 July 2021).
29. Kuehne, H.; Jhuang, H.; Garrote, E.; Poggio, T.; Serre, T. HMDB: A large video database for human motion recognition. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2556–2563.
30. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Curran Associates, Inc. 2019; pp. 8024–8035. Available online: https://papers.nips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf (accessed on 12 October 2021).
31. Ashley, K. *Applied Machine Learning for Health and Fitness*; Springer: Berlin/Heidelberg, Germany, 2020.
32. Senior, A.; Pankanti, S.; Hampapur, A.; Brown, L.; Tian, Y.L.; Ekin, A.; Connell, J.; Shu, C.F.; Lu, M. Enabling video privacy through computer vision. *IEEE Secur. Priv.* **2005**, *3*, 50–57. [CrossRef]
33. Newton, E.M.; Sweeney, L.; Malin, B. Preserving privacy by de-identifying face images. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 232–243. [CrossRef]
34. Boult, T.E. PICO: Privacy through invertible cryptographic obscuration. In Proceedings of the Computer Vision for Interactive and Intelligent Environment (CVIIE), Lexington, KY, USA, 17–18 November 2005; pp. 27–38.
35. Schiff, J.; Meingast, M.; Mulligan, D.K.; Sastry, S.; Goldberg, K. Respectful cameras: Detecting visual markers in real-time to address privacy concerns. In *Protecting Privacy in Video Surveillance*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 65–89.
36. Zadok, E.; Badulescu, I.; Shender, A. Cryptfs: A stackable vnode level encryption file system. In *Technical Report, Technical Report CUCS-021-98*; Computer Science Department, Columbia University. 1998. Available online: https://academiccommons.columbia.edu/doi/10.7916/D82N5935 (accessed on 12 October 2021).
37. Halcrow, M.A. eCryptfs: An enterprise-class encrypted filesystem for linux. In Proceedings of the 2005 Linux Symposium, Ottawa, ON, Canada, 20–23 July 2005; Volume 1, pp. 201–218.
38. VeraCrypt—Free Open Source Disk Encryption Software. Available online: https://veracrypt.fr/ (accessed on 15 August 2021).
39. EncFS—An Encrypted Filesystem. Available online: https://vgough.github.io/encfs/ (accessed on 15 August 2021).
40. Gocryptfs—Simple. Secure. Fast. Available online: https://nuetzlich.net/gocryptfs/ (accessed on 15 August 2021).
41. Filesystem in Userspace (FUSE) with Transparent Authenticated Encryption. Available online: https://github.com/netheril96/securefs/ (accessed on 15 August 2021).
42. Zadok, E.; Iyer, R.; Joukov, N.; Sivathanu, G.; Wright, C.P. On incremental file system development. *ACM Trans. Storage (TOS)* **2006**, *2*, 161–196. [CrossRef]

43. Hartvigsen, G.; Johansen, D. Co-operation in a distributed artificial intelligence environment—The stormcast application. *Eng. Appl. Artif. Intell.* **1990**, *3*, 229–237. [CrossRef]
44. Johansen, D.; Van Renesse, R.; Schneider, F.B. Operating system support for mobile agents. In Proceedings of the 5th Workshop on Hot Topics in Operating Systems (HotOS-V), Orcas Island, WA, USA, 4–5 May 1995; pp. 42–45.
45. Nordal, A.; Kvalnes, Å.; Hurley, J.; Johansen, D. Balava: Federating private and public clouds. In Proceedings of the 2011 IEEE World Congress on Services, Washington, DC, USA, 4–9 July 2011; pp. 569–577.
46. Gurrin, C.; Aarflot, T.; Johansen, D. GARDI: A Self-Regulating Framework for Digital Libraries. In Proceedings of the IEEE International Conference on Computer and Information Technology, Xiamen, China, 11–14 October 2009; pp. 305–310.
47. Ji, S.; Xu, W.; Yang, M.; Yu, K. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 221–231. [CrossRef]
48. Ng, S. Principal component analysis to reduce dimension on digital image. *Procedia Comput. Sci.* **2017**, *111*, 113–119. [CrossRef]
49. McCarthy, J.D.; Sasse, M.A.; Miras, D. Sharp or Smooth? Comparing the Effects of Quantization vs. Frame Rate for Streamed Video. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Vienna, Austria, 24–29 April 2004; pp. 535–542.
50. AI Can See Clearly Now: GANs Take the Jitters Out of Video Calls. Available online: https://blogs.nvidia.com/blog/2020/10/05/gan-video-conferencing-maxine/ (accessed on 8 September 2021).
51. Wang, T.C.; Mallya, A.; Liu, M.Y. One-shot free-view neural talking-head synthesis for video conferencing. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 19–25 June 2021; pp. 10039–10049.
52. Fitwi, A.; Chen, Y.; Zhu, S.; Blasch, E.; Chen, G. Privacy-Preserving Surveillance as an Edge Service Based on Lightweight Video Protection Schemes Using Face De-Identification and Window Masking. *Electronics* **2021**, *10*, 236. [CrossRef]
53. Dsouza, S.; Bahl, V.; Ao, L.; Cox, L.P. Amadeus: Scalable, Privacy-Preserving Live Video Analytics. *arXiv* **2020**, arXiv:2011.05163.
54. Rocha Neto, A.; Silva, T.P.; Batista, T.; Delicato, F.C.; Pires, P.F.; Lopes, F. Leveraging Edge Intelligence for Video Analytics in Smart City Applications. *Information* **2021**, *12*, 14.
55. Hosseini, M.P.; Tran, T.X.; Pompili, D.; Elisevich, K.; Soltanian-Zadeh, H. Multimodal data analysis of epileptic EEG and rs-fMRI via deep learning and edge computing. *Artif. Intell. Med.* **2020**, *104*, 101813. [CrossRef]
56. Lu, R.; Cai, Y.; Zhu, J.; Nie, F.; Yang, H. Dimension reduction of multimodal data by auto-weighted local discriminant analysis. *Neurocomputing* **2021**, *461*, 27–40. [CrossRef]
57. Zhai, Y.; Ye, Q.; Lu, S.; Jia, M.; Ji, R.; Tian, Y. Multiple expert brainstorming for domain adaptive person re-identification. In Proceedings of the 16th European Conference on Computer Vision, Part VII, Glasgow, UK, 23–28 August 2020; pp. 594–611.
58. Zhang, W.; Yang, D.; Zhang, S.; Ablanedo-Rosas, J.H.; Wu, X.; Lou, Y. A novel multi-stage ensemble model with enhanced outlier adaptation for credit scoring. *Expert Syst. Appl.* **2021**, *165*, 113872. [CrossRef]