Master Thesis

Software and System Engineering Programme

School of Software Engineering / FCS

Full Name: Victor Kugay

Student ID: M211МСПИН031

Decentralized Approaches for Geospatial Data Exchange using Web Technologies

NATIONAL RESEARCH
UNIVERSITY

Supervisor: Ph.D Ramon Antonio Rodriges Zalipynis

Moscow, 2023

Abstract

The approaches for exchanging spatial data in a Web has been investigated by many researchers in recent years due to the ubiquitous nature of spatial data. However, instead of centralized approaches which have been standardized by Open Geospatial Consortium (OGC) [29] and utilized in modern Earth observation platforms [62], this research examines decentralized approaches for exchanging spatial data. On the assumption that the Internet penetration rate was dramatically increased [20] and Web browsers got access to well standardized and totally interoperable Web APIs applicable for peer-to-peer real-time communications, decentralized Web protocols such as WebRTC and BitTorrent in particular are observed to investigate challenges and opportunities in cooperative access to dynamic geospatial datasets. The aim of this master thesis is to validate perspectives of the decentralized peer-to-peer approaches based on Web protocols such as WebRTC and BitTorrent for utilization in geographic information systems. To achieve this goal the research presents an overview of OGC Web services and modern decentralized Web protocols, describes benefits and issues of the cloud architecture and Earth observation platforms, examines Fog computations paradigm as a state-of-the-art decentralized approach for exchanging and processing data and formulate set of modifications to the existing centralized geographic information systems (GIS) services such as OGC Web Map Tile Service. Next, to validate and evaluate the proposed approach the research specifies design of the architecture, stack of technologies and modifications that are needed for successful integration of decentralized Web protocols to the existing GIS applications. Problems and solutions which are connected with selected technologies and identified during the implementation and deployment stage are collected in this work and mentioned in this work in order to help developers in their future experiments. Then research specifies a test plan to collect the statistics of the implemented prototype and analyze limitations and perspectives of the proposed algorithm. Finally, the research provides analysis based on collected statistics and formulates conclusions to assess applicability of the decentralized Web protocols to the existing GIS system.

Keywords: Geographic Information Systems, Spatial Data; Peer-to-peer protocols; WebRTC; BitTorrent; WebTorrent; WebSocket, Client-server architecture, Cloud architecture, Fog architecture.

Table of contents

1. Geospatial Web Services and Modern Challenges

1.1 Introduction

1.1.1 OGC Web Services as standard for exchanging spatial data

In recent years the relevance and importance of distributed processing GIS data has significantly increased due to the constant growing number of man-made satellites that are currently orbiting the earth as well as sharp growing spatial related devices that generate large amounts of GIS data. Fortunately, new technologies in distributed processing spatial data made it possible to arrange fundamentally new ways for spatial data operations. Moreover, the new generation of Web GIS platforms for real-time spatial data monitoring were presented in recent decades thanks to the Cloud and Fog computation models [31]. Considering this, distributed GIS platforms should be examined more carefully taking into account that in recent years distributed platforms have become more and more popular thanks to their high tolerance in parallel processing large amounts of data on multiple machines and introducing Real-Time communication protocol in Web.

Since the Open Geospatial Consortium [29] (OGC) has introduced the OGC Web Service standard [28] different requirements have been investigated by the researchers in order to summarize basic needs for processing geospatial data in a Web. The OGC Web Services requirements include HTTP RESTful interfaces for a specific kind of web services, which delivers knowledge on geospatial data. Geospatial services follow certain standard operations, including publication, discovery, binding, invoking and execution. In the next few years such requirements were summarized, analyzed, standardized and called Spatial Data Infrastructure (SDI). Is based on Service Oriented Architecture (SOA) the SDI provides the solutions to facilitate the challenges in the modern GIS world [3]. The solutions include frameworks, standards, technologies and human resources required for the successful collection, management, access, delivery and utilization of geospatial data in a global community. Despite that traditional SDIs have been built based on standards to represent and store spatial information in data files (e.g., GeoTIFF and Esri Shapefile) and database systems (e.g. PostGIS and Oracle Spatial), as well as to serve spatial data, metadata and processes via web services, in recent years the distributed approach for store and process data has become more popular. Traditional standards proposed by the Organization for Standardization (ISO) and the Open Geospatial Consortium are essential to enable spatial data interoperability among SDIs from worldwide agencies and institutions. For modern processing of big spatial and Earth Observation data, researchers need a next generation of SDIs. Users have to deal with hundreds (or thousands) of Earth Observation data files with varying spectral, temporal, and spatial resolutions by utilizing or writing software scripts to extract information of relevance

1.1.2 Earth observation platforms as the next generation of Spatial Data Infrastructure

Next generation of Spatial Data Infrastructure for modern processing of big spatial and Earth Observation data was based on cloud computing and utilized the main advantages of this paradigm. The dynamic allocation of computational and storage resources is emphasized in the widely studied distributed computing model known as cloud computing [2]. These resources are located in centralized data centers. Number of variables, including property costs, temperature, and the expenses of energy, occupy a significance in choosing where to put the data centers. These are just a few of the tools that cloud service companies provide. Others include developer tools for creating and launching cloud applications and complete server architectures for handling Virtual Machines (VMs) that use cloud resources. Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS) are the three main service methods for delivering cloud resources. Along with such a wide range of service types, four distinct cloud distribution models have appeared: (i) private cloud, (ii) community cloud, (iii) public cloud, and (iv) hybrid cloud. Virtualization technology is used by cloud service providers to dynamically supply vast amounts of on-demand resources on a pay-per-use basis. Figure 1.1 presents a Cloud computing overview made by Botta et al in their work Integration of cloud computing and Internet of Things: A survey [54].

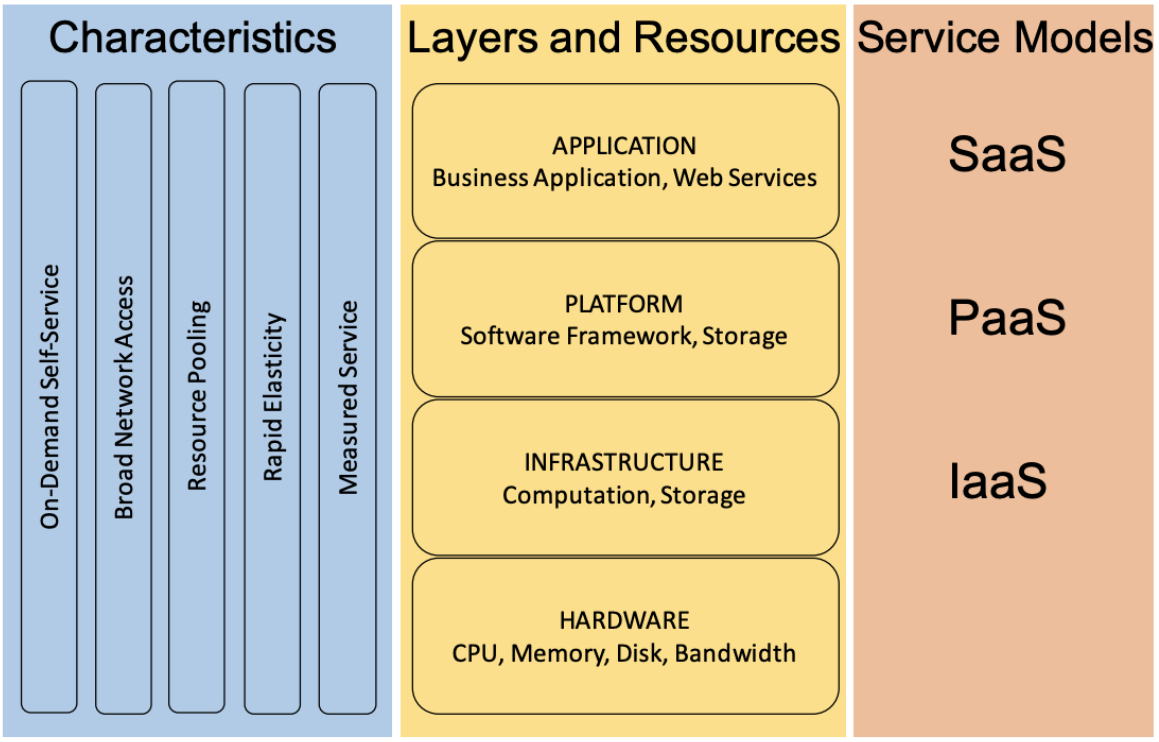

Figure 1.1 Cloud Computing Overview [52]

In 2011, the fundamental qualities of cloud computing were first included in the NIST description and continue to encompass the majority of its key features [39]. The previously stated features are explained in the paragraphs that follow.

On-Demand Self-Service As tools and services are provisioned automatically, cloud services can be obtained on demand and will be provided without requiring any human involvement.

Broad Network Access Every device linked to the Internet is able to communicate with the cloud and use the suggested services due to the global distribution of cloud computing DCs. Wearables, smart devices, and specialized computers are among the products. Many smart gadgets today use cloud services to access the Internet. Netflix, Airbnb, and Slack are just a few examples of services that can actually benefit from the dynamic resource management and the practically limitless resource capacity [6]. Given the widespread use of centralized cloud resources by service providers, this is the exact situation.

Resource Pooling Customers of cloud computing want to avoid over- and under-provisioning by acquiring changeable resources that match the present demand. When the available resource capability is greater than the level of demand at any given time, overprovisioning occurs, and resources are mismanaged. The opposite of overprovisioning is underprovisioning, which occurs when there aren't enough resources on offer to meet consumer demand. It is obvious that one would prefer to prevent both outcomes and allocate the ideal number of resources. Utilizing physical resources in accordance with consumer demand is achievable with the resource pooling notion. Traditional DCs only have static resources because private computers are unable to adapt flexibly without altering the infrastructure. On the other hand, cloud computing DC has the ability to dynamically modify the allocated resources in accordance with demand that varies over time [14]. Both types are presented on Figure 1.1. The cloud is made up of a collection of resources with the goal of creating an elastic resource ecosystem. Servers with configurable working and storage capabilities are among the tools available to meet the needs of picky clients. Resource virtualization makes it simple to specify and control the real resources. Thus, numerous clients use the physical servers concurrently, or in a multi-tenant manner. Obfuscating heterogeneous features is the purpose of having a virtual collection of resources, which makes it easier to use and compose generalized resources [27, 40].

Rapid Elasticity The rapid elasticity of the cloud is a key component of the cloud computing concept. The pool of physical resources can be expanded both horizontally and vertically while being used owing to resource virtualization technology, which requires little setup and administration work. Vertical scaling is the modification of particular capabilities of current VMs, whereas horizontal scaling is the deployment of additional VM instances to cooperatively service more clients. These dynamic scaling powers make it possible to manage client demand more effectively. As a result, it is possible to reduce energy consumption, resource waste, and total expense [14, 27].

Measured Service Whereas cloud services are pay-per-use, each and every transaction and interaction is tracked. This facilitates billing while also allowing customers and cloud service companies to evaluate the level of support. The customer and cloud supplier settle on the service's particular quality metrics, or SLAs, before buying cloud resources. If these agreements are breached, the cloud service might be required to pay a fee outlined in the SLAs in advance [27, 51].

Next generation of Spatial Data Infrastructure for modern processing of big spatial and EO data was based on cloud computing and utilized the main advantages of this paradigm. In order to retrieve information that is important, users must work with hundreds (or thousands) of EO data sets. These sets each have their spectral, temporal, and geographic resolution. In today's evolving world it is a bare necessity. Which is why, contemporary Technologies are building a sustainable future already in order to address the problem at hand. There

developed appropriate automation making use of cloud computing and, additionally, distributed networks as a main basis.

Information storage, retrieval, and management on the Internet are now generally done via cloud computing architecture. The primary objective of the next-generation network is to integrate cloud computing and web applications successfully. However, this integration faces a number of fundamental difficulties, including real-time response, service agility, and long-term connection. Fog computing has lately come to light as a more viable way to handle these issues with cloud and web applications. For content distribution and real-time data processing, the latter must provide seamless cloud and web convergence. While cloud computing focuses more on offering distributed resources over the primary network, fog computing can solve these issues by making resources and services readily available to end users at the network's edge.

1.1.3 Fog computing: a modern approach for processing and sharing spatial data

In computational models, centralized and decentralized computing strategies have alternated in recent decades. The mainframe approach of the 1970s and 1980s gave way to a wave of decentralization, which was then followed by the development of the client-server model in the 1990s. The first wave was brought on by the decreasing cost of personal computers and the growing desire to own exclusive processing capability [24]. The computational model once more changed at the start of the 2000s from a decentralized approach to a centralized one, specifically the cloud computing paradigm [1, 10]. Despite the fact that cloud computing is thriving and is unlikely to be replaced anytime soon, there is a force pushing towards a novel decentralized method to address the inherent issues with centralized systems, such as excessive latency and a lack of location awareness. The difference between this paradigm and the one that came before it is that this one will build upon rather than replace the previous one in order to enhance certain capabilities. Fog computing was created as a result of the current paradigm shift away from centralized cloud computing and toward decentralized computation [26].

In 2012, Bonomi et al. presented Fog computing as an innovative tool for the IoT [9]. The concept behind this cutting-edge distributed computing approach to the issue is to bring computational and storage resources closer to IoT devices at the network's periphery that are generating data. Near the periphery of the geospatial clients, the low-power gateway offered by Fog computing can boost speed and drastically cut on delay. As a consequence, less Cloud storage is required for geospatial big data. Likewise, a decrease in the mandatory transmission capacity leads to a total increase in efficiency.

A massive quantity of geospatial data is accumulated, saved, and examined for end consumers by systematized spatial data from numerous sources. Dozens of geospatial systems' margins are impacted by the low-power node's decreased delay at the client layer, while Fog computing has originated as a developing option that can successfully manage increased throughput. In comparison to the Cloud, FC uses less storing and transmission capacity for indelible data processing. Health care, watershed control, land use, coastal, marine, and municipal planning are a few examples of new uses that rely on CC-based models [7]. For several purposes, including over-relay analysis, statistical computing, data visualization, and query formulation, this framework is capable of analyzing and combining divergent thematic levels for geospatial data analysis and generating alternative situations. Geospatial data are handled and evaluated by the Cloud in a conventional Cloud-based GIS

framework, which needs considerable working time and large Internet capacity. This problem is meant to be solved by Fog computing with the delivery of local computation near the client's perimeter along with being reasonably close.

Fog computing is a relatively new study focus area, so there are few established approaches and practical solutions available. For the aforementioned qualities to be achievable, a lot more investigation is required. The Open Fog Consortium is a significant organization that promotes and supports the concept of cloud computing. The Open Fog Consortium's guiding fundamental concept is to maintain an open fog computing architecture to facilitate collaborative study with numerous groups. An important factor in technology advancements in cloud computing is the cooperation of numerous groups with varying specializations.

However decentralized processing and data storage introduce multiple challenges in maintenance data integrity and shared data processing. Unfortunately, replacing all functional requirements of GIS systems is impossible. Handling large amounts of spatial data requires significantly more powerful hardware. For instance, in 2019 exclusively the volume of open data produced by Landsat-7 and Landsat-8, MODIS (Terra and Aqua units), and the three first Sentinel missions (Sentinel-1, -2 and -3) reaches the amount encompassing about 5 PB [46]. As well as Google's example. The global 30m map of forest cover and change from 2000-2012 [18], which quantifies forest dynamics regarding disease, lodging, fires and tornadoes, is co-located in Cloud storage and reached from analyses of massive quantifies of Landsat data. The analysis mentioned needed more than 1M hours of CPU, which would have taken over 15 years to complete on a single computer. However, this analysis was completely processed in less than four days in case of running them in the Google Cloud infrastructure on 10 thousand CPUs concurrently. Thus, the Fog-based computations may be considered as an extra optimization layer for traditional GIS systems. However, it can not fully replace the cloud Earth observation platforms.

1.1.4 Decentralized geospatial data and Real-Time communication protocol

It is obvious that the Fog-based paradigm requires peer-to-peer communication protocols. In the context of the main topic of this paper such protocols have to be Web compatible. Fortunately, the appropriate protocol was created by Google in 2011 [49]. WebRTC, short for Web Real-Time Communication, is a set of rules, protocols and API. On the assumption that today's WebRTC is a well standardized technology with ubiquitous support by the Web browsers [13] there are dozens of enterprise companies which use WebRTC protocol in production with the purpose of peer-to-peer data access, real-time communication or video calls. One of the most well-known companies are Google, Facebook, Discord, Snapchat, Cisco and many, many more.

As it was described in Abstract, introducing Real-Time communication protocol to the Web has changed the perspectives of non-centralized processing GIS data in a Web. Peer-to-peer protocols such as BitTorrent and WebRTC make it possible to distribute processing GIS data and organize shared access to large amounts of data over the world. Based on decentralized protocols, the Fog computing paradigm, presented in the early 2010s, extends the Cloud computing paradigm. Fog computings are characterized by geographical distribution, a very large number of nodes, as a consequence of the wide geo-distribution, real-time interactions, predominance of wireless access, heterogeneity, interoperability and federation. In recent years the Fog Computing paradigm was formulated enough in order to

create the next generation of GIS applications. Meanwhile, Spatial Data Infrastructure frameworks still do not use Fog Computing paradigm and distributed protocols. Fortunately, the introduction of Real-Time communication protocol in the early 2010s triggered a significant amount of interest in distributed processing spatial data. The Fog Computing thanks to Real-Time communication protocol allows researchers all over the world to combine their machines and GIS data sets to Smart Grids [9]. Therefore it gave developers the possibility to communicate, to process, to share, to aggregate and to interoperate between multiple different Earth Observations platforms. There are dozens of Earth observation platforms and GIS standards which were established by the GIS community in recent years and in this context interoperability became more and more important for free access to GIS data.

Thankfully introducing WebRTC protocol to the modern browsers, the non-centralized peer-to-peer communications become more and more ubiquitous. For instance, it is estimated that the global Web Real-Time Communication market will reach $30.4 Billion by 2027 in comparison with $6.7 Billion in the year 2022 [17]. Presenting distributed protocols such as BitTorrent and websocket and their implementation to the web made it possible to create real-time systems for simultaneous sharing and processing data. As an example, applicable to the GIS systems it is well known in the GIS community ArcGIS GeoEvent server [21] which provides functionalities for real-time event-based processing geospatial data streams. There are dozens of enterprise GIS systems who have already integrated ArcGIS GeoEvet server to stream geospatial data. The ArcGIS GeoEvent server is developed to create real-time maps, filter data using spatial conditions, create geofences on-the-fly without disconnecting from real-time data streams and store geospatial data for later processing using traditional OGC Web services. In spite of ArcGIS GeoEvent utilizing WebSocket as a communication layer and implements traditional client-server architecture it is an important step from widespread cloud Earth observation platforms to mixed Fog-based real-time GIS systems.

1.2 Motivating scenario

The motivating scenario of this work is to provide an approach for decentralized publication of spatial data using Web technologies. It is expected that introducing peer-to-peer protocols such as WebRTC and BitTorrent make spatial data more accessible thanks to reducing network bandwidth and latency, resources consumption of the OGC Web services and utilizing users' hardware. However, modern GIS systems are fully centralized and next use case illustrates centralized approach for processing spatial:

1. The instances of the OGC Web services are deployed to the cloud infrastructure that gives them an opportunity to process large amount of spatial data;
2. The researchers download spatial data from the spatial data hubs such as Sentinel Hub [62] and process such data using Web applications and deployed Web services;
3. The processed data are saved to the data storages in the clouds and are unambiguously identified by the unique URI;
4. The researchers download published data using Web applications and unique URIs to analyze and process the data.

Described above use case presents a traditional Spatial Data Infrastructure approach when the users execute tasks in the Cloud infrastructure while Web browsers or desktop applications are used to render the computed data. In the case of Cloud-based computing, users' hardware is not utilized for processing or exchanging spatial data and cloud services are fully responsible for processing and exchanging spatial data that produce a single point of failure and make such an architecture sensitive to high volume workload and failures.

Decentralized publication of the spatial data: in contradistinction from the use case which was described above the decentralized approach for peer-to-peer publication spatial data is based on utilization of users` hardware. It is expected that such an approach will increase accessibility and reduce the workload of the existing GIS systems. Next use case illustrates desirable decentralized use case for publication spatial:

1. The instances of the OGC Web services are deployed to the cloud infrastructure and spatial data are downloaded from the spatial data hubs;
2. The researchers using Web application request processing such data using deployed Web services;
3. The researchers using Web application requests processed data and save processed data on users' device;
4. The researchers using Web application notify special registry of the decentralized spatial data;
5. The researchers using Web application download spatial data through peer-to-peer Web protocols and special registry.

The described above use case provides opportunities for reducing the workload of the centralized OGC Web services due to their partial exclusion from the communication chain while modern decentralized Web protocols guarantee peer-to-peer reliable access to the spatial data. Such a use case allows immediate sharing of processed data and decreases costs for storing spatial data using cloud-based databases.

Decentralized processing of the spatial data: in the traditional way, processing spatial data carried out using either desktop applications or a set of Web services such as OGC Web services. Fortunately, there are many open source implementations of the OGC Web services such as ArcGIS, PyWPS, etc. Introducing Web services made it possible to process petabytes of spatial data. However, millions of lightweight spatial processing tasks may be delegated to

users' devices in order to distribute workload of the centralized services. Next use case illustrates decentralized approach for processing spatial data:

1. The researchers download spatial data from the spatial data hubs and Web application estimate complexity for processing spatial data;
2. The Web application delegate computing complex tasks to the centralized Cloud-based OGC Web services;
3. The Web application process lightweight tasks using researchers' hardware and save processed data on researchers' device;
4. The researchers using Web application notify special registry of the decentralized spatial data that the data are ready;
5. The researchers using Web application download spatial data through peer-to-peer Web protocols and special registry.

Combination of both use cases for publication and processing spatial data provides the possibility for introducing Fog computations paradigm to the existing GIS systems. However, this research is concentrated only on approaches for decentralized exchanging spatial data due to complexity and voluminosity of the both topics, for instance tasks complexity estimation, real-time access to the spatial data, cooperative access to the dynamic data. Such topics are out of scope of this master thesis.

1.3 Challenges and objective

The objective of this work is to integrate decentralized approaches for exchanging spatial data using Web technologies such as WebRTC and BitTorrent to the existing approaches for exchanging spatial data such as OGC Web services. The relevance of the integration is based on Earth observation data challenges which were identified in “Big data concepts, methods and analytics” [16] research by Amir Gandomi at al. They characterized Earth Observation data as “a term that describes large volumes of high velocity, complex, and variable data that require advanced techniques and technologies to enable the capture, storage, distribution, management, and analysis of the information” [16]. At the same time Marcus Paradies at al., in the work “Large-Scale Data Management for Earth Observation Data — Challenges and Opportunities” [36] identified data storage & organization, data access and data processing pipelines as crucial challenges in management of Earth Observation data. As a solution, Open Geospatial Consortium proposed Web Map Tile Service Implementation Standard [32] whose main purpose is to provide communication protocols and ensure reliable spatial data transferring between server and clients. OGC WMTS implementation based on Service Oriented Architecture (SOA) became a standard in the geospatial community. On the assumption that the SOA implied that all users` requests are handled by the server, it became one of the most unresolvable bottlenecks in today's publication and processing of large amounts geospatial data. Despite the fact that the Cloud-based Earth observation platforms have a possibility to dynamically allocate sufficient resources to handle many users` requests there are still multiple limitations in centralized approaches. For instance, Amit Gajbhiye et al. in the work “Cloud Computing: Need, Enabling Technology, Architecture, Advantages and Challenges” [15] identified that low WAN bandwidth for transferring data across data centers and within the datacenter causes performance bottlenecks and tends high costs for I/O in cloud environments. Thus, optimization of exchanging spatial data becomes one of the crucial topics in the GIS community. On the assumption that the browser`s resources and APIs as well as Web protocols were significantly enhanced and modern approaches to store, process and share data using Web technologies were researched in recent years, reducing costs for exchange spatial data using decentralized Web protocol became more and more realistic. This research proposes decentralized Web protocols for sharing spatial data between peers as an extra approach for exchanging spatial data.

As it mentioned above, approaches for decentralized sharing data such as the Fog computing paradigm became ubiquitous and well standardized in recent years. Processing spatial data on users` devices using web applications has been the subject of several researches. However, it requires appropriate environment needs to utilize decentralized protocols in the context of GIS Web services. After throughput performance of the computer networks has dramatically increased in recent years the approaches for utilizing users` hardware for processing data become more relevant. In addition to improving network bandwidth, Internet access has become more and more accessible. According to the Internet Words Stats data [19] as of January 2023, there are more than 35 countries with a penetration rate more than 80%. Moreover, on the long distance the penetration rate is constantly increasing. The Internet Words Stats also provides statistics [20] of the global internet penetration rate from 2009 to 2022 grouped by region. According to these statistics penetration rate was doubled in Africa, tripled in Asia and Pacific, more than tripled in Arab States. Developed countries also present constant growth of the penetration rate and reached more than 80% penetration rate in the recent years. Considering this, the users have got the access to transmitting large amounts of data over the Internet in recent years. Moreover, the bandwidth continues to grow strongly. The Internet Words Stats prove this statement and

present research. According to this data the bandwidth in Europe was quadrupled from 49 Tbit/s in 2015 to 204 Tbit/s in 2021.

Considering this, constant growth in Internet penetration rate and network bandwidth as well as development of the decentralized Web protocols generate the environment for the next generation of the Spatial Data Infrastructure and reduce the costs for exchanging spatial data.

1.4 Methodology

1.4.1 Keywords and terms selection

Next keywords and expressions were used for the purpose of investigating scientific papers which were considered in preparing the master thesis. These keywords and expressions were the seeds to create the list of related works.

Keywords: Cloud architecture, Fog architecture, Open Geospatial Consortium Web services, Spatial Data Infrastructure, Earth observation platforms, peer-to-peer protocols, WebRTC, BitTorrent.

Search terms: “design Fog computing framework” or “Fog computing architecture”, “Fog-based processing spatial data” or “geospatial Fog computing”, “Cloud-based processing spatial data” or “Open Geospatial Consortium Web services based on Cloud computations”, “Open Geospatial Consortium Web services”, “peer-to-peer Web protocols”, “WebRTC overview”, “WebRTC geospatial” or “WebRTC GIS” or “WebRTC Open Geospatial Consortium Web services” or “peer-to-peer Geographic Information Systems”, “performance analysis of peer-to-peer applications” or “performance analysis of WebRTC applications” or “estimation of WebRTC applications”, “distributed processing spatial data” or “distributed processing data” or “Web-based processing spatial data”.

1.4.2 Online sources

Next online resources were investigated for the preparation purpose:

- **HAL**: hal.archives-ouvertes.fr
- **ACM Digital Library**: dl.acm.org
- **IEEExplore**: ieeexplore.ieee.org
- **Google Scholar**: scholar.google.com
- **Mendeley**: www.mendeley.com
- **CiteSeer**: citeseerx.ist.psu.edu
- **MDN Web Docs**: developer.mozilla.org
- **RTC for the Web**: webrtc.org

1.4.3 Documents selection criteria

In this section the main inclusion and exclusion criteria are presented for the purpose of providing clear statements which were used in the next research.

Main Inclusion Criteria:

- Papers dealing with Open Geospatial Consortium web services as traditional Spatial Data Infrastructure pattern;
- Papers dealing with Cloud-based deployment of the existing Earth Observation systems such as Sentinel Hub, Google Earth Engine, SEPAL, etc.;
- Papers dealing with Fog architecture as the next generation of the Cloud architecture. The problems and solutions in Fog architecture;
- Papers dealing with peer-to-peer web protocols such as Bittorrent and WebRTC. Applicability to the Fog architecture;

- Papers dealing with processing spatial data in a Web browser. The limitations, benefits and perspectives.

Main Exclusion Criteria:

- Papers that do not deal with distributed processing or access to the data;
- Papers dealing with peer-to-peer architecture but not applied to the web;
- Papers dealing with distributed processing or access to the data but not applied to the web;
- Papers dealing with distributed processing data but not applied to client-server, cloud or fog architectures.

1.4.4 Design and evaluation logic

The end value of the research is a system (prototype) for decentralized transmitting spatial data using peer-to-peer Web technologies. Considering this, the research is based on the existing systems analysis to formulate a set of requirements, limitations and constraints of the modern decentralized systems for transmitting data. The existing systems analysis describes challenges and perspectives of the analyzed solutions to filter inefficient hypotheses during the analysis stage. The requirements for the prototype are based on Earth observation platform requirements proposed by Gilberto Camara et al., in the work “Big Earth Observation Data Analytics: Matching Requirements to System Architecture” [8]. In particular, the system should be extensible and scalable, follow software reuse approach and collaborative work approach. The design architecture is dedicated to the technical aspects and includes description of the systems, algorithms and models of the proposed GIS system. For the description purpose the UML description of the components and algorithms are used. The prototype for decentralized transmitting spatial data is based on the designed architecture and includes frontend and backend applications. Frontend application includes algorithms for the decentralized transmitting spatial data using WebRTC and Bittorrent protocols as well as Graphical User Interface (GUI) to illustrate meta information about system statistics. The backend application should include torrent and signaling servers. They can be combined both together. The signaling server is responsible for establishing WebRTC communication between peers. The torrent server is responsible for providing meta information which is needed for establishing BitTorrent communication. The prototype is deployed and used in the evaluation phase. Evaluation phase is based on the IEEE Standard for Software Test Documentation [61]. The test steps should include functional testing and performance testing. The prototype should be deployed using Docker and Docker Compose to guarantee constant resource limitations such as RAM, CPU and network bandwidth. Functional testing should be performed using manual testing.

2. Web Technologies for the Proposed Approach

2.1 Open Geospatial Consortium (OGC) Web services

2.1.1 OGC Web services

The term “geospatial web service” may be defined as follows. It may be classified as a modular application that offers geospatial data services and relevant information [35]. Geospatial web services, similarly to other services, implement a series of common actions throughout the course of their lifecycle, including publishing, discovery, binding, invoking, and execution. Regardless of their obvious parallels, OGC web services differ from those based on W3C and OASIS standards for that geospatial web service standards were created concurrently with those issued by W3C and OASIS. The major reason why two different service categories differ from one another is the consequence of the mentioned factor. A decent illustration would be WSDL, UDDI, and SOAP technologies that remain fundamental once again to regular web services yet seem to be no more than optional for geospatial web services.

For the purpose of publishing geospatial data, OGC has created several standards across a wide range of sectors. OGC Specifications outline the interface methods that web service implementations must offer. In particular, Web Coverage Service (WCS) and Web Feature Service (WFS) represent two standards. Yet they all lack techniques for visualizing data. Or, they provide algorithms that produce raw data. Another web service can subsequently collect this data, convert it into images, and publish it using an OGC interface like Web Map Service (WMS). WMS “produces maps of spatially referenced data dynamically from geographic information”, which is clear from its definition [23]. Each and every WMS implementation must agree with this feature. Utilizing the Web Map Service may convert data from certain sources including database management systems, WFS, and WCS. OGC registered Web Processing Service (WPS), an interface that “facilitates the publishing of geospatial processes, and the discovery of and binding to those processes by clients” [43].

Much like the W3C works, the guidelines and methods that the OGC employs can be categorized as a stack for its geospatial online services (see Table 2.1). As was stated earlier, HTTP is the primary communication layer, which also applies to standard online applications built on the REST architecture. By adopting the GeoServices REST API, OGC hopes to further incorporate its geospatial online services with REST [11]. Universal Resource Locators (URLs) could be used by API clients to access materials stored on the server. The client (web application or browser) receives map pictures, text position input, or other data as a response. Geospatial web applications are distinguished by their data component. Geospatial online services can output data in binary or text formats, with text data typically being an XML document with the proper grammar. Geography Markup Language (GML), one of these grammars, is specified by standard as an “XML encoding (...) for the transport and storage of geographic information modeled in accordance with the conceptual modeling framework (...) and including both the spatial and non-spatial properties of geographic features.” [12]. Therefore, it can be used as a protocol for sharing location data via the

internet and additionally as a tool for describing geographic systems. Geometric elements like points, lines, and polygons are described in GML. It is simple to clarify physical things like highways, boundaries, rivers, and others using those three items. Since GML 3.0, raster data, such as numerous sensor data and images, has been added. The OGC specification also includes dozens of additional XML-based languages. For instance, the SensorML specification can be used to define devices and measurement data. A system called CityGML is used to exchange and store virtual 3D city models serves as another example.

Process	Query	ISO 19125-1
	Integrative	**WPS**
Metadata	Service description	WSDL, **ISO 19119**, ISO 19109
	Data description	ISO 19115:2003, ISO/TS 19139
	Cataloging	**CSW**, UDDI, WSIL, **ebRIM**, **ISO metadata**
Messaging	Application interfaces	**WCS**, **WMS**, **WFS**, WICS, WCTS, **SOS**
	User interface	WMC
	Message	**HTML GET / POST**, SOAP
Type	Schema	XSD, DTD, OWL
Data	Vector	**OGC-GML**, **SensorML**, OGC-WKT/WKB
	Data file	SDTS, VPF, DIGEST, HDFEOS
	Data	**XML**, ASCII, **JSON**, **GeoJSON**
Communications		**HTTP**, SSL, SMTP, FTP, IIOP

Table 2.1 Geospatial web service standard stack [35, 53]

Due to its minimal data overhead, JSON has gained more popularity for transmitting geospatial data, which is particularly so with the new GeoServices REST API. Another format called GeoJSON is built primarily on JSON. This open standard allows for the encoding of a point, line, polygon, and other spatial data kinds. The project's objective is to use the JSON structure to encrypt location data. Since the launch in 2008, it has gained a lot of popularity. For instance, Twitter sends geotagging information using GeoJSON. Furthermore, it is simple in implementation. Any JSON instrument can be used to handle GeoJSON data because GeoJSON documents adhere to the JSON format. Despite its obvious benefits, GeoJSON has been left out of the GeoServices REST API plan.

2.1.2 OGC Web Map Tile Service (WMTS)

This section provides an overview of OGC Web Map Tile Service (WMTS). OGC WMTS is analyzed for the purpose of decentralized spatial data exchange and was selected for the modification due to flexibility of the standard, close connection to the spatial data exchange topic and wide distribution in the geospatial community. Considering this, next research provides OGC WMTS definition, describes WMTS interfaces, highlights potential bottlenecks and optimizations perspectives. OGC Web Map Tile Service Implementation Standard [32] was developed by Open Geospatial Consortium and, it is important to note, was inspired by OSGeo Tile Map Service Specification [34]. OGC Web Map Tile Service has become a response to the geospatial community needs for serving map tiles. On the assumption that the OGC standard guarantees interoperability between OGC WMTS and OSGeo, OGC WMTS implementation provides both resource and procedure oriented approaches, in particular OGC WMTS utilizes both RESTful and KVP/SOAP encoding as a communication layer.

OGC WMTS standard is an important part of the OGC set of standards and is embedded in the OGC services chain, for instance close collaborators of the OGC WMTS is OGC Web Map Service (WMS). The main purpose of the OGC WMTS is serving static spatial data. In the hierarchy of OGC Web services, WMS is responsible for rendering custom maps and collaborating with OGC Style Layer Descriptor (SLD) which generates static styled map tiles. Such map tiles include predefined content, extent, and resolution based on dynamic spatial data. At this time, OGC Web Map Tile Service doesn't generate any data and implements an interface for serving static data where the bounding box and scales have been constrained to discrete tiles. Considering this, OGC Web services architecture approach distributes responsibilities to render, styling and serving spatial data. It gives an opportunity to dynamically scale Web Map Tile Service instances and guarantees stress resilience of the architecture. Figure 2.2 schematically illustrates the components diagram for the described above set of OGC Web services.

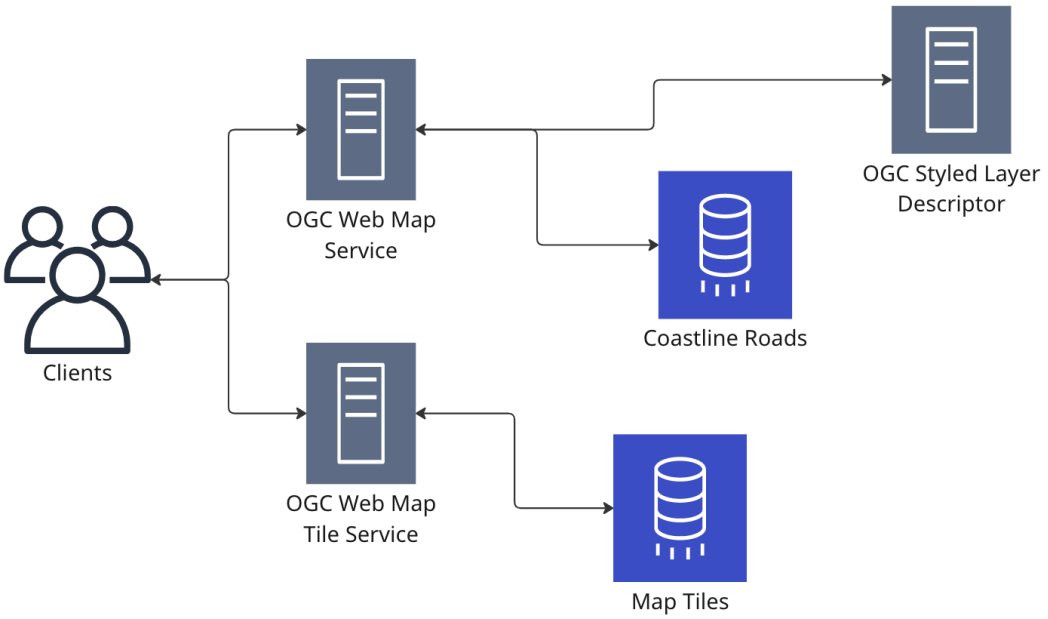

Figure 2.2 Example of OGC Web services

The WMTS interface provides three types of resources: FeatureInfo, Tile and ServiceMetadata. A tile is a fragment of the map layer. Usually these are sliced raster images which represent regions of the map. As a service description OGC proposed a ServiceMetadata document which is common for all OGC Web services. The ServiceMetadata document should describe all tiles which are available in each layer as well as all communication protocols and encodings. In particular, ServiceMetadata documents should describe each format, type, scale, coordinate reference system over each geographic fragment. FeatureInfo interface should provide information about the features located at a particular pixel of a tile map. Most of the details of the WMTS interface are described in OGC Web Services Common Implementation Specification [33], for instance OGC WMTS interface FeatureInfo the same as OGC WMS GetFeatureInfo operation. RESTful API of the OGC Web Map Tile Service includes getCapabilities, getTile and getFeatureInfo endpoints. The response of the getCapabilities operation is an XML document which provides information about ServiceMetadata. According to the standard such description should include ServiceIdentification, OperationsMetadata, ServiceProvider, Contents and Themes properties. There are other available properties for the getCapabilities implementation. However, for the purpose of this research the most important property is Contents and in particular nested to the Contents field Layer property which includes the ResourceURL information. The ResourceURL includes format, resourcesType and template for the unambiguous identification of all available tiles in the system. Figure 2.3 in the appendices section presents the Layer UML model and illustrates the position of ResourceURL in the WMTSCapabilites.xml. The algorithm for the communication with getCapabilities and getTile is as follows. After getCapabilities information was fetched once and parsed by the WMTS client the getTile operation is invoked many times to fetch fragments of the map. On the assumption that the ResourceURL provides a template for the getTile operation, such templates as TileMatrixSet, TileMatrix, TileCol and TileRow are transformed to the unique URIs for each tile according to the selected map region, layers and scale. The figure 2.4 presents the example of the ResourceURL. Then the OGC WMTS client creates HTTP connection to the OGC Web Map Tile Service and fetches concurrently the raster data.

```
<ResourceURL
  resourceType="tile"
  template="https://cartoweb.wmts.ngi.be/1.0.0/crossborder/default/{TileMatrixSet}/{TileMatrix}/{TileRow}/{TileCol}.png"
  format="image/png"
/>
```

Figure 2.4 Example of ResourceURL from the OGC WMTS server

As it mentioned above, OGC Web services and OGC WMTS service in particular are based on the client-server architecture. Figure 2.5 schematically illustrates a use case which demonstrates a centralized approach for the tiles downloading process using OGC Web Map Tile Service. Despite the fact that the OGC community intentionally created read and write services to distribute the workload for serving static geospatial data and generating styled static map tiles, there are still many challenges in serving static spatial data using OGC WMTS. High-concurrency Web systems are widely used nowadays. Considering this higher scalability and more availability are one of the most important issues for OGC WMTS. In order to guarantee these characteristics the maintainers of GIS systems have to increase resources of such services or upgrade existing components architecture. There are no standardized components diagrams of OGC Web services deployment. However usually such deployment includes reverse-proxy, instances of the OGC Web services and instances of the

NoSQL document-oriented databases such as MongoDB as a database layer. According to the service-based architecture both horizontal and vertical scaling are available for scaling the system but unfortunately both algorithms increase costs for maintaining the system and do not guarantee high availability in case of oversaturation when faced with thousands of concurrent user requests. Centralized architecture implies that such a vulnerability affects all clients and becomes a single failure point for the whole system. Fortunately modern decentralized Web protocols allow to distribute the workload and partially redirect requests from the OGC WMTS directly to the clients.

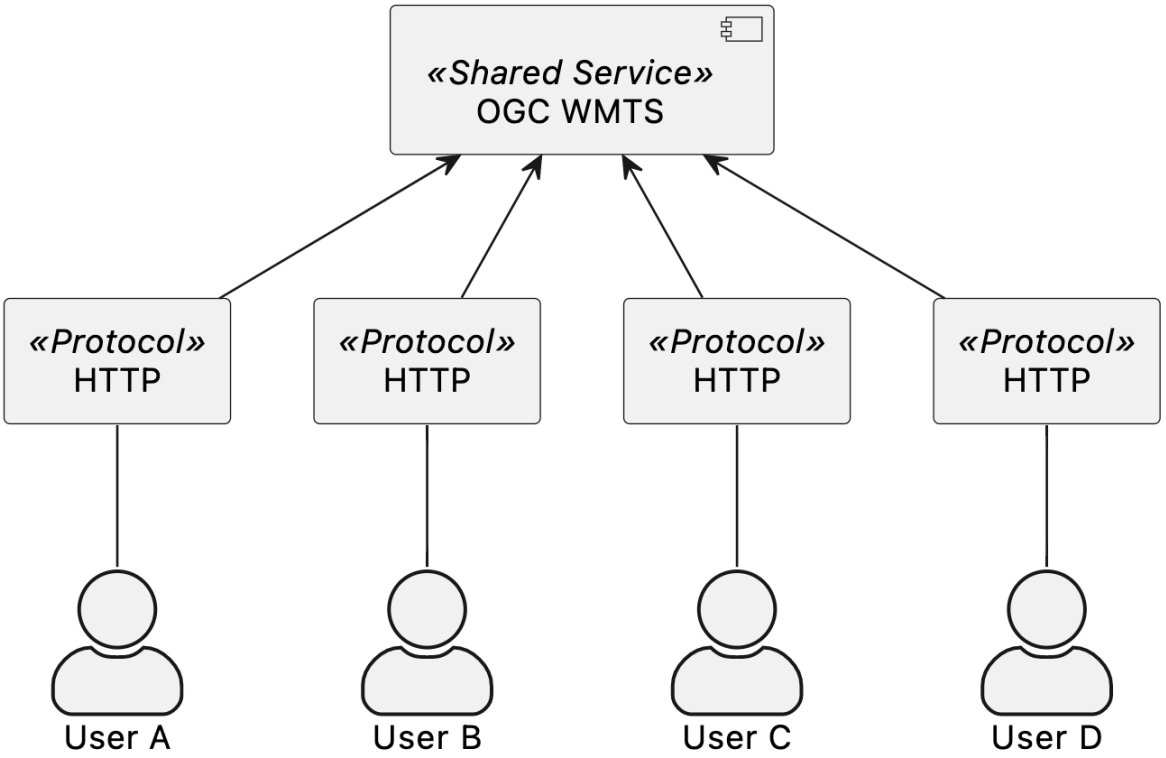

Figure 2.5 Top level UML diagram for OGC Web Map Tile Service

2.2 Existing Decentralized Web protocols

2.2.1 Real-Time communication protocol for the Web

The WebRTC was developed with the purpose of making possible audio and video communications, screen sharing and data transferring using native browsers API, HTML and JavaScript features. The developers by Google define interoperability as one of the main features of the WebRTC when presented the protocol in 2011. In order to reach such the goals Google developers combined existing technologies e.g. protocols, data types, audio and video codes, and created the protocol which does not need to install any plugins. On the assumption that the interoperability between different clients was ensured, the introduction of the well standardized WebRTC protocol to the most popular browsers destroyed the barriers between the clients and presented possibilities to create fog computing systems.

The communications in WebRTC is divided into two planes: signaling and media planes. Figure 2.6 illustrates simplified schema of the WebRTC planes. On the assumption that the signaling plane is not standardized, the developers select appropriate solutions to exchange signaling messages. Usually the web servers based on WebSocket or HTTP protocols are used for this purpose. Considering this there is no interoperability in the signaling plane between different realizations. The second media plane is well standardized and implemented to most Web browsers.

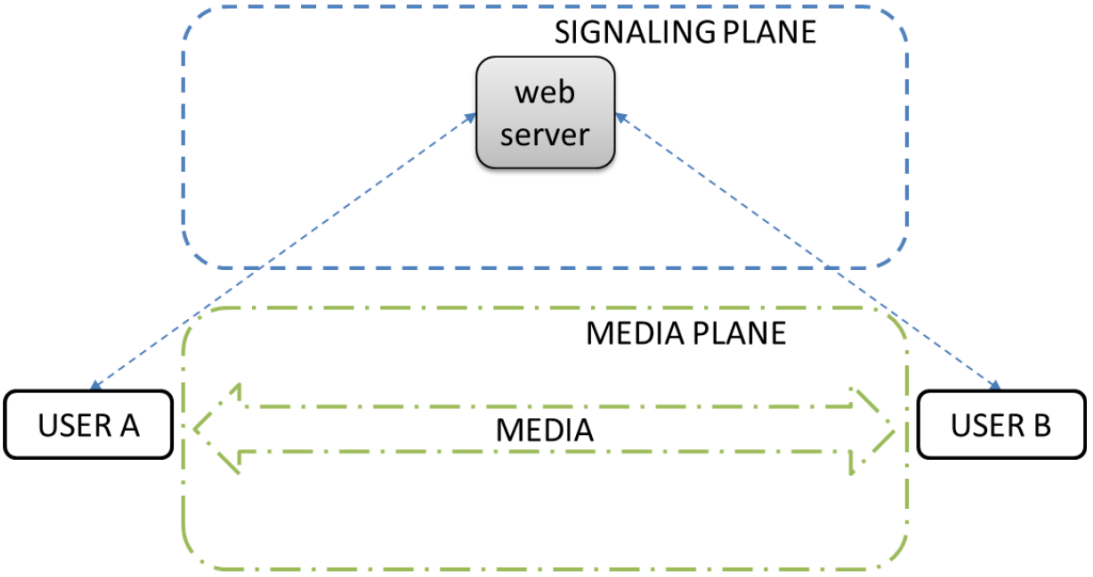

Figure 2.6 Simplified WebRTC planes [49]

In real-time communication systems media traffic is time sensitive so one of the main challenges in development WebRTC was latency optimization. The two main reducing latency strategies were presented by the Google developers. The first strategy is video and audio codecs development for processing streams. The second strategy is supporting both TCP and UDP protocols. The data exchange is not guaranteed in real-time communication due to the rapid state changes, so UDP potentially provides better performance but the research [4, 47] illustrates that the packet loss leads to an increasing utilization of the data channel and finally the performance of the UDP even worse than TCP. Some networks don't

allow UDP traffic at all, or maybe they don't allow TCP so both protocols are supported by WebRTC.

Ideally, the communication in WebRTC should be done in peer-to-peer mode using a direct IPs between peers. However in most cases it is not possible due to the topology of the peer-to-peer graph and Network Address Translation (NAT). The first limitation is connected with network bandwidth and requirements to transmit media streams from every peer to all other peers that lead to reducing the bandwidth of the channel. It has been solved by establishing the dynamic limitations on the number of concurrent peer-to-peer sessions. The second limitation requires introducing Session Traversal Utilities for NAT (STUN) and Traversal Using Relay for NAT (TURN) servers that break the peer-to-peer architecture fundamentals. Fortunately, NAT is not required in local networks, so establishing peer-to-peer communication in local networks is possible.

Thus, there is a need to provide the method to create connection when the direct media path is not possible. The ICE (Interactive Connectivity Establishment) was introduced with the purpose of enabling direct communication between WebRTC users. It gathers possible IP addresses and ports in order to test their connectivity. Two users exchange the signaling messages indirectly, by using a Web server. Both users describe all their possible addresses, i.e. ICE candidates that will be used for communication. These possible ICE candidates are presented on the figure 2.7.

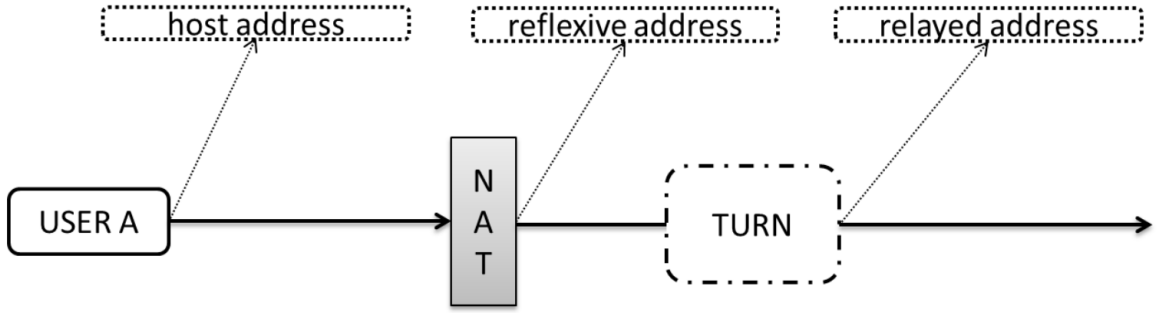

Figure 2.7 ICE candidates types [49]

Particular description of the ICE candidates is out of the scope of this master thesis. However it is critically important for understanding STUN and TURN modules highlight that ICE candidates includes:

- **Host address** - address allowing a peer-to-peer communication;
- **Reflexive address** - a public address allocated to a device behind a NAT. Reflexive address provided by the STUN server placed in a public network;
- **Relayed address** - an address provided by a TURN server that acts as a media relay and is placed in a public network.

STUN (Session Traversal Utilities for NAT) is a protocol, used as a tool by the other protocols for identifying real addresses in case of NAT. The purpose of the STUN is to handle users' requests and responses with a public IP address and port allocated by a NAT. It is relevant in case of binding and keeping alive the port on users' hardware. The user sends a request (Binding Request) to the STUN server. STUN server identifies users' address behind the NAT and sends a response (Binding Success Response) with attribute XOR-MAPPED-ADDRESS within the body of the STUN message.

TURN (Traversal Using Relays around NAT) is a tool used when peer-to-peer communication between clients is not possible for the NAT reasons. If the users' hardware supports only Symmetric NAT the combination of one internal IP address plus a destination IP address and port is mapped to a single unique external source IP address. Considering that the identification of the public users' address using STUN in case of Symmetric NAT is not possible the concept of TURN comes into play. The simplified TURN communication is presented on figure 2.8.

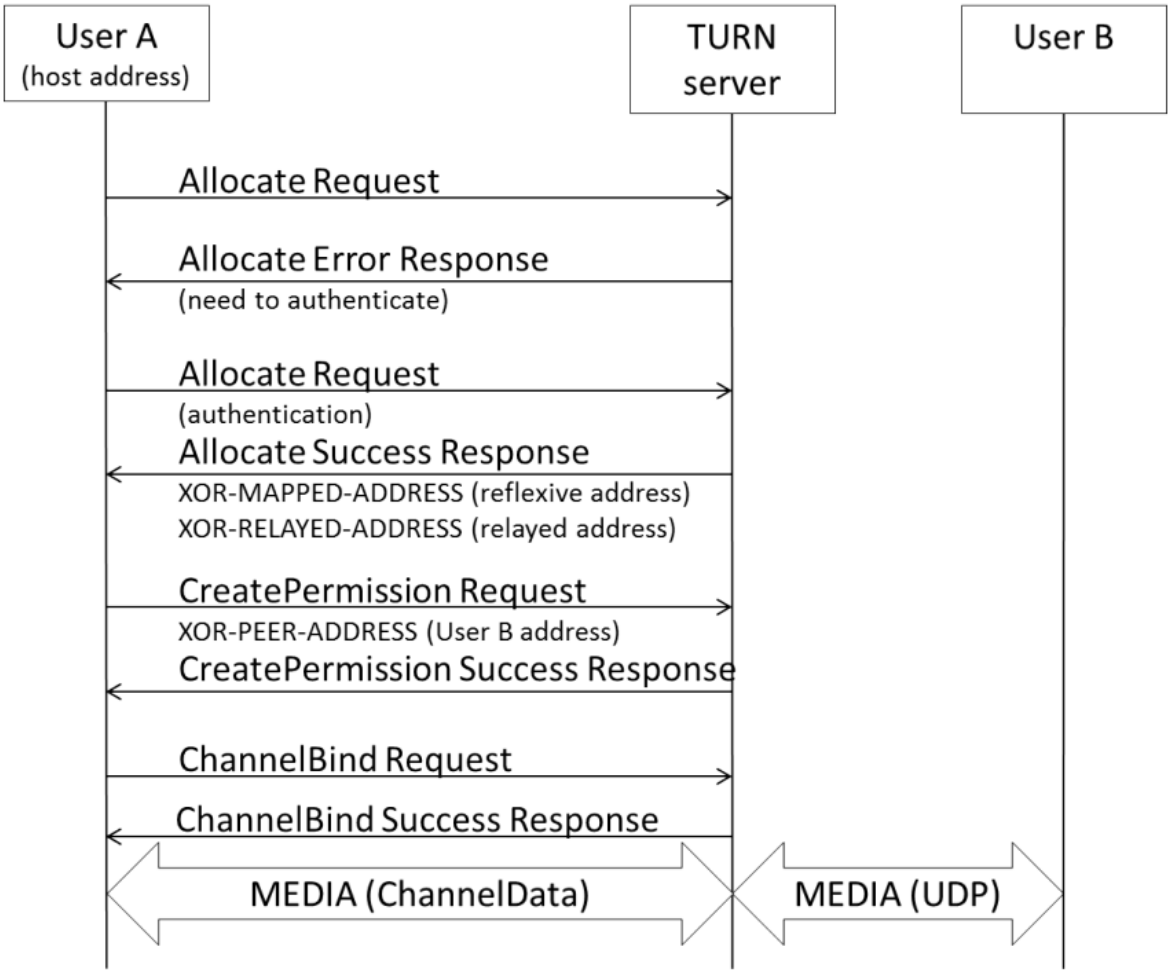

Figure 2.8 Simplified TURN flow diagram [49]

The user allocates a connection to the TURN server and keeps it alive until the end of the communication. The audio and video streams as well as data channel is proxying through the TURN server. Hence, peer-to-peer communication is not possible in this case and TURN servers rolling us back to traditional client-server architecture. Of course there are benefits in case of WebRTC and TURN such as data analytics, streams optimization, and streams collapsing. However, implementing client-server architecture makes WebRTC more unreliable due to extra costs for maintaining TURN servers and risks for transmission media streams through centralized TURN servers much higher than peer-to-peer communication between clients.

Considering this, the WebRTC protocol supports three types of ICE candidates: host address, reflexive address received by STUN or relayed address received by TURN. After ICE candidates have been collected by both clients the connection probe is checked. First, the

WebRTC protocol sorts collected ICE candidates from the highest to lowest priority. Then the offer to establish the communication is sent to the remote client using SDP (Session Description Protocol). When the remote endpoint receives the offer, it answers with its own ICE candidates. At this moment both users have the complete list of ICE candidates and one client is set up as controlling agent. Next, the ICE candidates are paired up by each endpoint so that different combinations can be tested. The ICE candidates checking stage is called Connectivity Checks and starts from high priority candidates. Connectivity Checks implements STUN Binding Request/Response. In the end the controlling agent selects the ICE candidates pair with valid connectivity checks. The hosts addresses are more preferred, than reflexive addresses and last relayed addresses. In this way the peer-to-peer communication using hosts addresses or reflexive addresses are more preferable than client-server connection in case of TURN. Unfortunately, the signaling communication and connectivity checks take some time depending on network latency, signaling server bandwidth, etc. There are extensions to begin connectivity check while all ICE candidates are not collected. Thus, there are optimizations of ICE candidates connectivity checks bottleneck [5]. In summary, there are three possible approaches for establishing connection in WebRTC: (1) direct connection using host addresses, (2) direct connection using reflexive addresses, (3) proxying media and data streams through TURN server. The simplified schema is presented on the figure 2.9.

As a communication layer WebRTC uses RTP (Real-time Transport Protocol), notably the Secure RTP (SRTP) [37]. RTP provides mechanisms to ensure peer-to-peer transport of real-time data e.g. audio and video. RTP does not provide any resource reservation or quality of service. However, RTP Control Protocol (RTCP) provides information about the quality while RTP provides information about sequence numbering, timestamps and monitoring of devilry. SRTP is a secure extension of RTP and provides encryption and message authentication of RTP and RTCP. It is important to highlight that SRTP does not provide any guarantees or reliability. However the protocol provides information about the number of lost packets and last received sequence number. Working together with the controlled by WebRTC protocols allows dynamically adapting media to the network variations and there are ongoing efforts on providing congestion control algorithms. Data Channel is used to transmit non-media data. Both TCP or UDP protocols may be used as transport layers and existing ICE candidates pairs are used. The Data Channel API is based on Stream Control Transmission Protocol (SCTP), encapsulated in Datagram Transport Layer Security (DTLS) protocol. DTLS protocol ensures secure communication between peers and is simply TLS over UDP while SCTP is used as a controlling mechanism in transmitting data packets and increasing resilience and reliability. There are no limitations about data types for Data Channel in WebRTC.

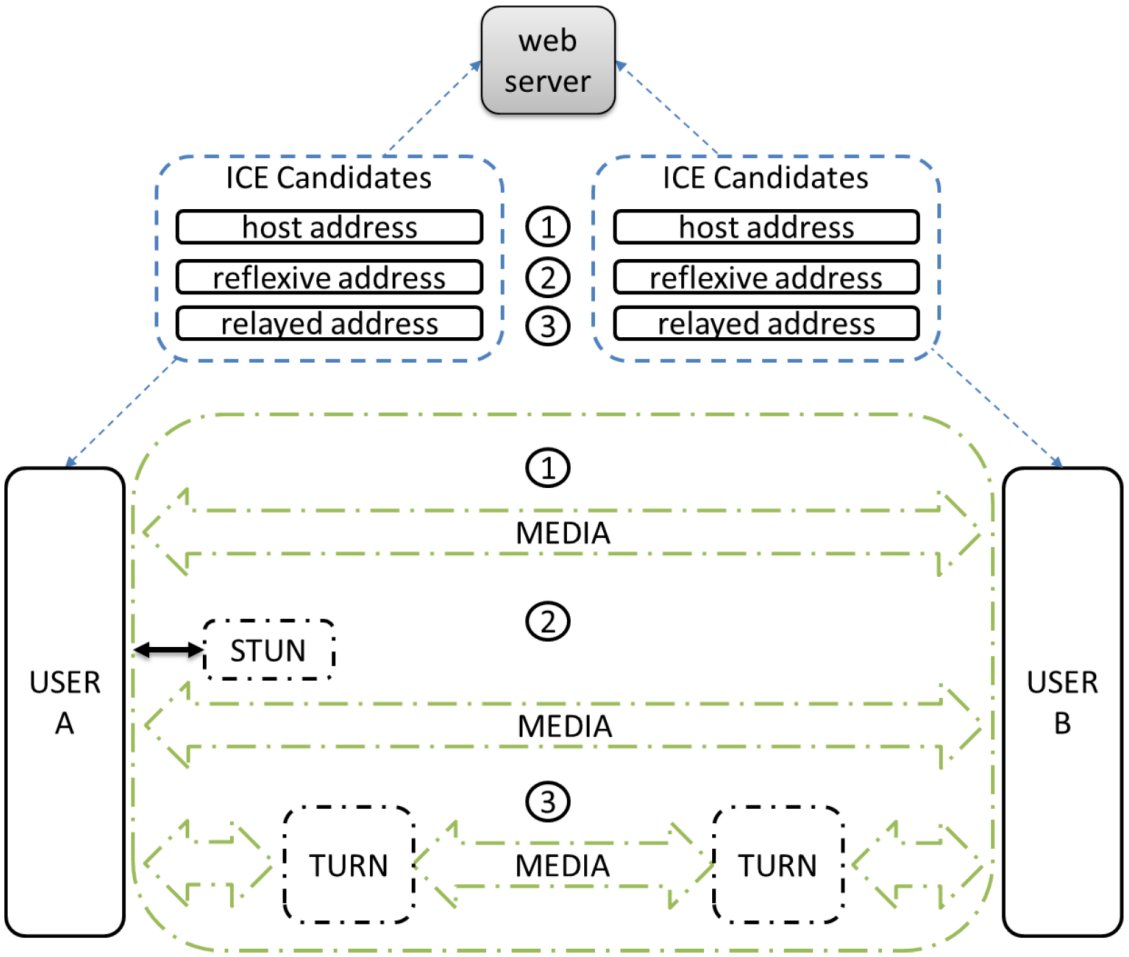

Figure 2.9 Simplified connection approaches in WebRTC. (1) direct connection using host addresses, (2) direct connection using reflexive addresses, (3) proxying media and data streams through TURN server.

Hence, the WebRTC utilizes both TCP and UDP protocols as transport layers in relation to environment constraints. DTLS and SRTP as a securing layer and RTP and SCTP as a communication with peers layer. SDP is used to describe the candidates during signaling phase and ICE candidates are used to establish the communication between peers. Finally, the STUN and TURN servers are used to relay data over NAT and all protocols, servers and tools are orchestrated by WebRTC browser API. The WebRTC protocol stack is presented on figure 2.10. On the assumption that the WebRTC combine and control already implemented to the browsers protocols which included a wide range of quality controlling, buffering and transmission mechanisms, the developers task is simplified.

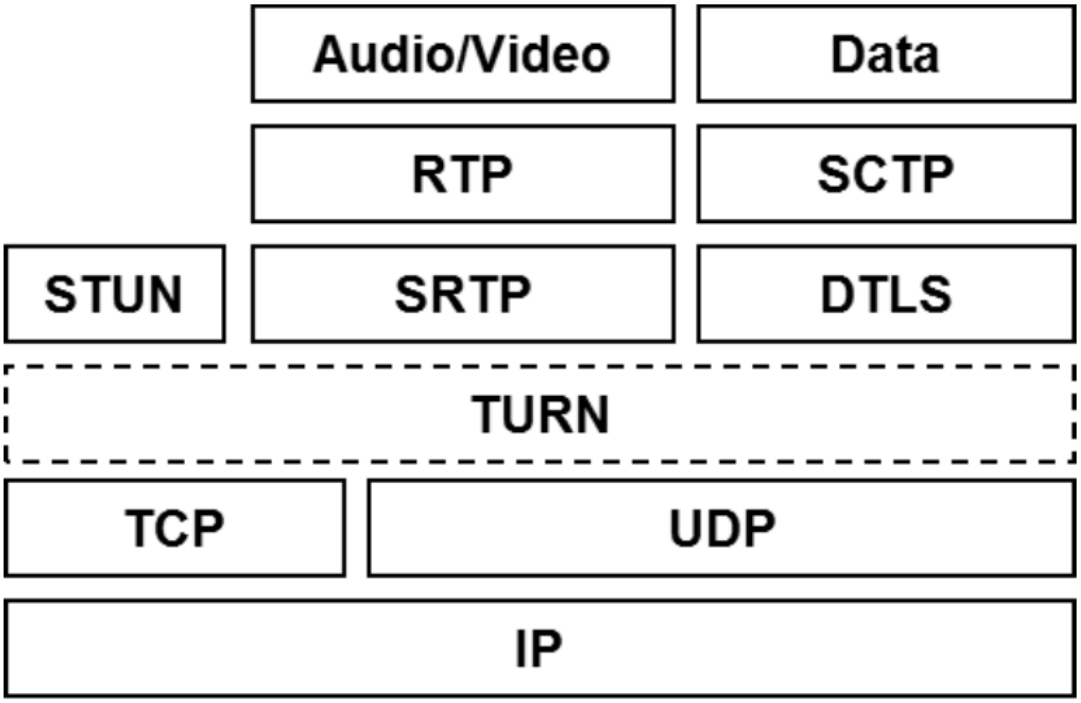

Figure 2.10 WebRTC protocol stack

2.2.2 BitTorrent protocol and its utilization in a Web environment

BitTorrent is a peer-to-peer protocol. It allows to utilize users' resources for the purpose of optimizing bandwidth consumption for the publisher. In contrast to client-server architecture where all clients communicate with the server to download data, BitTorrent protocol proposes an algorithm to split the file by multiple chunks and send it to dozens or thousands clients. Then clients communicate with each other and fetch missing files' chunks directly from the different clients. As a result both download time and load on the server is reduced due to excluding the communication with centralized servers and possibility to download the file from thousands of different hosts. Such groups of hosts are called "peers". They are orchestrated by a central component. Central component is called a "tracker". This combination of peers and tracker is called a "swarm". The tracker controls the resources transfer between peers. There are two types of peers: seeds and leechers. Seeds are the peers that have the complete file and share it. Leechers have no complete file but they want to have it and communicate with seeds to download it. Leechers download the chunks from the multiple seeds and immediately share the chunks with other peers. Considering this, when all file fragments are downloaded the leecher becomes a seed. Set of all seeds and leechers (swarm) controlled by the tracker or multiple trackers. Such trackers know particular seeds who have the complete file or particular file fragments.

Client-server approach implies that the file is completely downloaded from the server while peer-to-peer protocols such as KaZaA, eDonkey, DirectConnectimply implies that the file is completely downloaded from the particular client bypassing the server. However BitTorrent protocol implies that the file is downloaded from multiple different clients. On the one hand it gives the opportunity to distribute the workload between clients in order to reduce bandwidth and significantly improve reliability. On the other hand it introduces new

challenges to know which clients have the particular pieces or identification most effective pieces downloading strategy. In order to guarantee reliability and effectiveness of the data exchanging BitTorrent provides policies for data transferring. According to Johnsen et. al [25] such rules describe patterns of the sub-pieces downloading. Every piece is broken into sub-pieces and transferred between clients. Strict Policy implies that the sub-pieces for that particular piece are requested before sub-pieces from any other piece. Rarest first Policy means that when a peer selects the next piece to download, it selects the piece which the fewest of their peers have. Random First Piece Policy means that once you start downloading, you don't have anything to upload. The policy is then to select the first piece randomly. When the first piece is complete, we change to "rarest first". Finally, Endgame mode Policy checks the transfer rate and selects the client with the best transfer rate to guarantee selection of the fastest channel for the communication. These policies prevent peers overload and ensure reliability of the decentralized peer-to-peer system.

To start the downloading process the torrent client should have the special file with meta-data of the torrent. Such meta-data may be downloaded from indexing service or may be generated on the fly directly on the client if all file parameters are available. According to Cohen (2013) a torrent meta-data includes information about files, checksums and tracker info. Tracker info is called announce URL and is needed to clarify which trackers have to be used to download the file because BitTorrent supports downloading from the multiple trackers. Checksums describe length of each chunk of file as well as complete file length. Files information includes files' paths and sizes. Usually the files are cut into small chunks 256KB or 512KB and hash codes of each files' pieces are included to the torrent file. Such hash codes are used to verify downloaded chunks and guarantee completeness and correctness of the downloading. Unfortunately, BitTorrent V1 utilizes the SHA-1 algorithm for generating the hashes. Such an algorithm is unsafe and provides many collisions. For instance there are many cases when users` devices were infected by the attackers using substituted files content. Fortunately BitTorrent V2 uses the SHA-2 and SHA-256 in particular to guarantee less collisions and better safety. However the BitTorrent protocol still does not guarantee safety and assumes scanning downloaded data for the protection users' devices. Moreover hashing each chunk is a very intensive operation for the CPU and I/O so several BitTorrent clients implement storing information about the chunk directly in the torrent file.

BitTorrent peer-to-peer communications are based on TCP-based protocol also known as peer wire protocol which operates with bi-directional streams of length-prefixed protocol messages. On the assumption that the BitTorrent session management is based on TCP sessions there are no tearing down methods beyond the TCP teaser down itself. Peer-to-tracker communication usually is based on HTTP protocol. Mandatory parameters of the peer-to-tracker communication are info_hash, peer_id, port, uploaded, downloaded and left parameters which are sent using a Query [48] fragment in the URI [48]. Despite the fact that the BitTorrent specification provides recommendations for the encoding parameters for the peer-to-tracker communications there are many implementations of the torrent trackers which in general inherit BitTorrent specification with minor changes. Most important parameters of the peer-to-tracker communication are info_hash and peer_id parameters because info_hash includes SHA-1 hash of the info parameter of the torrent file and peer_id is a unique identifier of the torrent consisting of a 20 bytes string. Both info_hash and peer_id allow uniquely identifying torrent peers and torrent files between trackers. According to the Quental at al. [41] the messages chain in the BitTorrent protocol includes next stages. Handshake is an initial message for the start of communication between peers, bitfield is a bit field representation of successfully downloaded pieces. Interested action informs a peer that a

peer is interested in its chunks. Request action requests chunk of the file. Piece includes file chunk and corresponding offset and chunk index. Have action informs that the peer has the complete chunk. Choke and unchoke informs requesters that the peer can or can not send chunks.

In a Web environment the BitTorrent protocol is presented by the webtorrent [50] NPM package which provides functionality for the decentralized files sharing using the Web browser. The webtorrent is based on the simple-peer [45] NPM package which utilizes native browser WebRTC API to establish peer-to-peer communication between torrent clients and transmit the files. The webtorrent provides a set of services and packages to create a complete Web-based torrent infrastructure e.g. torrent server and torrent client. The webtorrent server supports IPv4 and IPv6 and is based on WebSocket protocol whose main purpose is to share the information about pieces of the file and transmit signaling data for establishing WebRTC connection between torrent clients. There are possibilities to communicate over HTTP and UDP however the Web browser implementation of the webtorrent client supports only WebSocket and HTTP protocols. Considering this the webtorrent client utilizes WebSocket protocol to handle the tracker requests and send signals for the WebRTC communication. The WebRTC implementation is based on the simple-peer NPM package as a top level abstraction over browsers' WebRTC API. The webtorrent client supports simultaneous downloading of multiple torrents, streaming files, discovering of the peer via DHT [44] and ut_pex, streaming media content and is supported by the most popular Web browser such as Chrome, Firefox, Opera and Safari. One of the key advantages of the webtorrent is supporting BitTorrent V2 which as it mentioned above improves stability and security thanks to utilization SHA-2 algorithm for hashing meta information about torrents, and utilization WebRTC data channels for the transmitting pieces of the files. It gives the possibility to transmit large amounts of data directly from the clients and provides a delivery guarantee which is based on the WebRTC protocol stack as well as excellent support by the users' devices. Both webtorrent server and WebTorrent client are implemented in JavaScript, the language of the Web, and webtorrent client supports both browser and node.js environments with small adaptation changes for the server-side utilization.

Despite the fact that the total BitTorrent traffic significantly decreased from the early 2000s when peer-to-peer protocols produced about 50% of the global Internet traffic, decentralized approaches for transmitting the data are currently still relevant. According to the Global Application Traffic Share report made by Sandvine [42] BitTorrent occupied 2.91% of global Internet traffic and 5.64% of mobile devices Internet traffic in 2022. Better rate on mobile devices is connected with distributing updates to Facebook and Twitter as well as transferring large files like video and music clips.

2.3 Comparison of OGC Web Services, Cloud and Fog based architectures

In this section, OGC Web Services, Cloud-based and Fog-based architectures are analyzed applicable to the development and maintenance of the geographic information systems. The aim of this section is to identify potential advantages, risks and bottlenecks. The research is based on classification of the existing systems provided by the previous research. The study implements the mixed methodology of comparative analysis [38]. The analysis rests on case-oriented implementation of the comparative strategy in qualitative methodology of comparative analysis and provides a coherent comparison of the architectures.

2.3.1 Limitations

Next limitations were researched applicable to the server-oriented architecture, cloud-based and fog-based architectures applicable to the geographic information systems.

Features	Web services	Cloud computing	Fog computations
Management	Distributed	Centralized	Mixed
Computation device	Locally deployed servers	Powerful server system	Any device with computation power
Nature of failure	Highly Diverse	Predictable	Highly Diverse
Distance from user	Close	Far	Close
Network latency	High	High	Low
Node Mobility	Middle	Very low	High
Intermediate hops	One/Few	Multiple	One/Few
Real-time application handling	Difficult	Difficult	Achievable
Participating nodes	Static	Variable	Constantly dynamic
Storage capacity	High	High	Low

Table 2.11 Limitations of WS, CC and FC.

Management As it mentioned above the approaches for processing geospatial data have overcome two main changes. From distributed processing on the local devices or locally deployed OGC web services to centralized processing using Earth observation platforms resources. Fog-based computations implement a mixed type of architecture. Tasks are partially handled on the local devices. It implies that particular types of tasks can be handled only on the local devices. In this way, fog-based computations combine the best characteristics of both architectures.

Computation device The limitations in locally deployed GIS web services are obvious. It is service agility and service scalability. Based on pay-per-use plans cloud-based solutions make centralized Earth observation platforms dramatically more effective than self-hosted GIS web services. However, in the context of processing geospatial data, a centralized approach faces several challenges. Due to constant growth of the spatial data sets and expandable functional requirements centralized systems become bottleneck in processing spatial data in a web. In contrast to the cloud-based architecture, fog-based architecture proposes to execute tasks on the local users machines. fog-based computations mitigate single point of failure risk and reduce resource consumption on the central server.

Nature of failure However fog-based computations as well as self-hosted OGC web services tend to be affected by much more issues than centralized solutions. Possible sources

of the issues connected with fog-based computations are network bandwidth, shared resources, technical condition, environmental issues, etc. As a result the maintenance of cloud-based solutions is more predictable. In this context, the main characteristics of cloud-based architecture which were described in Chapter 1 become the main advantages of cloud solutions.

Distance from user Close distance to the user in traditional GIS web services and fog computations generates the main advantage of these two systems. The costs for transmitting spatial data are minimal. It made it possible to handle large amounts of spatial data with minimal network delays. In contradistinction from traditional GIS web services, fog-based computations are distributed to multiple close devices. This key advantage of the fog computations allows to create locally based GIS networks for real-time processing and shared access to the geospatial data.

Network latency As it mentioned above the close distance from the user in fog-based computation model allows the users to have shared access to the spatial data. Despite that fog-based computations include multiple challenges such as selection and balancing edges it still proposes significantly better network latency and does not require constant Internet access.

Node Mobility Standard OGC tools implies self-hosted servers on the locally based hardware. It leads to multiple issues connected to the maintainability, accessibility and scalability of such services. Unfortunately, high cohesion of the OGC web services makes it difficult to migrate such systems. On the assumption that the OGC web services include a static number of nodes it needs to cooperate with existing databases or hardware to transfer the system. At the same time cloud architecture does not imply physical transfer of the hardware. According to the fog computations paradigm the special devices produce multiple types of data and store it to the local or users devices. It makes it simple to interchange the data through the users and create shared datasets and geospatial processing systems all over the world.

Intermediate hops As it mentioned in Chapter 1 fog-based computations implies utilization cloud solutions. However it is not required for the fog. According to the paradigm cloud solutions may be used as last chance servers. Only unresolvable on the local devices tasks push their data to the clouds and wait for the responses. Moreover the edges of the fog may create the graphs and delegate tasks to one another for cooperative work. According to the Tahira Sarwar Mir et al. research introducing fog significantly decreases workload of the cloud nodes. Delegating working tasks to the edges allowed maintain nodes available over whole highload testing.

Real-time application handling Introducing real-time protocols to the web browsers made it possible to create distributed systems for shared access and handling spatial data over the Internet. Centralized solutions for real-time communications face maintenance challenges. Real-time systems require low network latency and high productivity in parallel managing and processing shared data. Development of the real-time systems over the OGC web services is absolutely not possible due to the architectural constraints of the web services. The chain of web services implies that requests are handled by web services one by one and users wait for the results of computations. Cloud-based solutions provide the same architectural plan in context of the chain of the services but in cloud infrastructure with cloud benefits. Fog-based computations provide a fundamentally different approach and make it available to reduce network latency and distribute the load to implement real-time access to the geospatial data.

Participating nodes On the assumption that the principals of the spatial data infrastructure and OGC web services were formed at the beginning of the Internet era the amount of the nodes for standard OGC web services are static**.** It leads to multiple challenges such as load balancing, scalability and stability. As it mentioned in characteristics of the cloud architecture section cloud-based computations resolve issues of the standard web services and provide tools to manage containerized applications. However the disadvantages of the centralized approach are still here. As opposed to classical web applications or containerized applications fog-based applications include constantly dynamic participants. In this way it includes costs to manage the graph of the fog's edges, excludes single points of failure and provides dramatically more methods to interoperate geospatial data.

Storage capacity Undoubtedly, due to constant growth of geospatial data sets and continuously increasing georeferenced devices classical OGC web services can no longer process all needed GIS data. Modern Earth observation platforms were developed to solve this task and provide scalable tools for processing spatial data. Unfortunately, in recent years the necessity for processing spatial data has dramatically increased. Introducing IoT made it impossible or too expensive to process large amounts of spatial data in the clouds. In this context fog computation paradigm provides simple and understandable tools to utilize users devices, their storages, CPU and memory resources.

2.3.2 Summary and conclusion

On the one hand, fog-based computations introduce completely new challenges such as managing resources, data integrity, designing peer-to-peer resilient architecture and peer selection algorithms. On the other hand, research illustrates that fog-based computations take advantage of most useful features of the service-based and cloud-based architectures. Table 2.12 presents SWOT analysis of the fog-based computations architecture in GIS systems.

	Strength	Weaknesses
Internal	● Utilization of users hardware; ● Distributed data storage; ● Cooperative tasks handling; ● Close distance from the users.	● Challenges in maintenance data integrity; ● Challenges in shared processing tasks.
	Opportunities	Threats
Externa l	● No single failure point; ● High mobility of the system; ● Low network latency; ● Real-time communication.	● Need Internet access to send the complicated task to the Cloud.

Table 2.12 Advantages and disadvantages of the fog-based computations GIS systems

Most valuable strengths of the fog-based architecture are utilization of users hardware, distributed data storage and cooperative tasks handling. All strengths generate one of the most important advantages of the fog-based systems. There is no single failure point. The

load balancing between edges in the fog and failed nodes can be easily replaced. High mobility of the system also based on all system strengths. According to the fog computations paradigm there can be masters and slaves nodes but masters nodes can also be easily replaced. Low network latency and possible real-time communications are based on close distance from the users. It allows the use only of local networks and exploits their throughput to capacity. However distributed processing and data storage introduce multiple challenges in maintenance data integrity and shared data processing. Such challenges are partially explained in the design architecture chapter of this thesis. Unfortunately, replacing all functional requirements of GIS systems is impossible and handling large amounts of spatial data requires significantly more powerful hardware.

3. Decentralized Approaches for Geospatial Data Exchange using Web Technologies

As it mentioned above the OGC WMT services are used to exchange map tiles and are based on the client-server architecture. Considering this, combining Fog computing paradigm e.g. decentralized data transmitting lead to significantly reducing OGC WMT services workload, and consequently decreasing costs for exchanging spatial data. The approach is to combine decentralized Web protocols such as WebRTC and BitTorrent, and OGC WMT services which are responsible for exchanging map tiles. According to the Fog computing paradigm both centralized and decentralized approaches for processing and exchanging data are utilized to combine benefits which were described above and mitigate disadvantages of the centralized Cloud-based architecture. Considering this, the following functional requirements for the decentralized spatial data exchange are combined with the previously analyzed OGC Web services requirements.

3.1 Bird's Eye View

The OGC WMTS client should load OGC WMT service metadata using getCapabilities endpoint provided by OGC WMT service. The OGC WMTS client should parse such metadata and create URLs for the requesting map tiles. The OGC WMTS client should initialize requests for every particular tile. In contradiction to standardized OGC WMTS flow the OGC WMTS client first requests tiles from the Torrent server using WebSocket. If the Torrent server responds with status code OK, it means that there is a seed, the OGC WMTS client will wait for initialization of the WebRTC connection and wait when appropriate tile will be transmitted through the WebRTC DataChannel. The figure 3.1 illustrates the algorithm for decentralized tile downloading. If the Torrent server responds with status code ERROR, there is no seed, the OGC WMTS client will immediately download the tile from the OGC WMT service. The figure 3.2 illustrates the use case when the torrent has no seed for requested tile. In this case, the only data source for the tile downloading is OGC WMT service.

The diagrams mentioned above describe two algorithms. First algorithm is a centralized algorithm. Such an algorithm introduces only one extra participant in comparison with the OGC WMTS standard. Second algorithm mixes centralized and decentralized approaches. Second algorithm implies that the tile should be initially downloaded from the WMT service and subsequently seeded using Torrent server. Next, tiles will be downloaded bypassing the OGC WMT service. In this way, the algorithm is divided into two steps. First is fetching the tile from the OGC WMT service. Second is leeching the tile. Fetching step includes both OGC WMT service and Torrent server participants while the leeching step includes only communications with Torrent server and peer-to-peer communications using WebRTC and BitTorrent protocols. The leeching stage requires a signaling step for establishing WebRTC communication and data transmitting step to send binary data through the WebRTC DataChannel. In this way, OGC WMT service is fully excluded from the communication and Torrent server is responsible for lightweight signaling messaging that allows to control Torrent server workload and easily apply vertical and horizontal scaling in Cloud infrastructure.

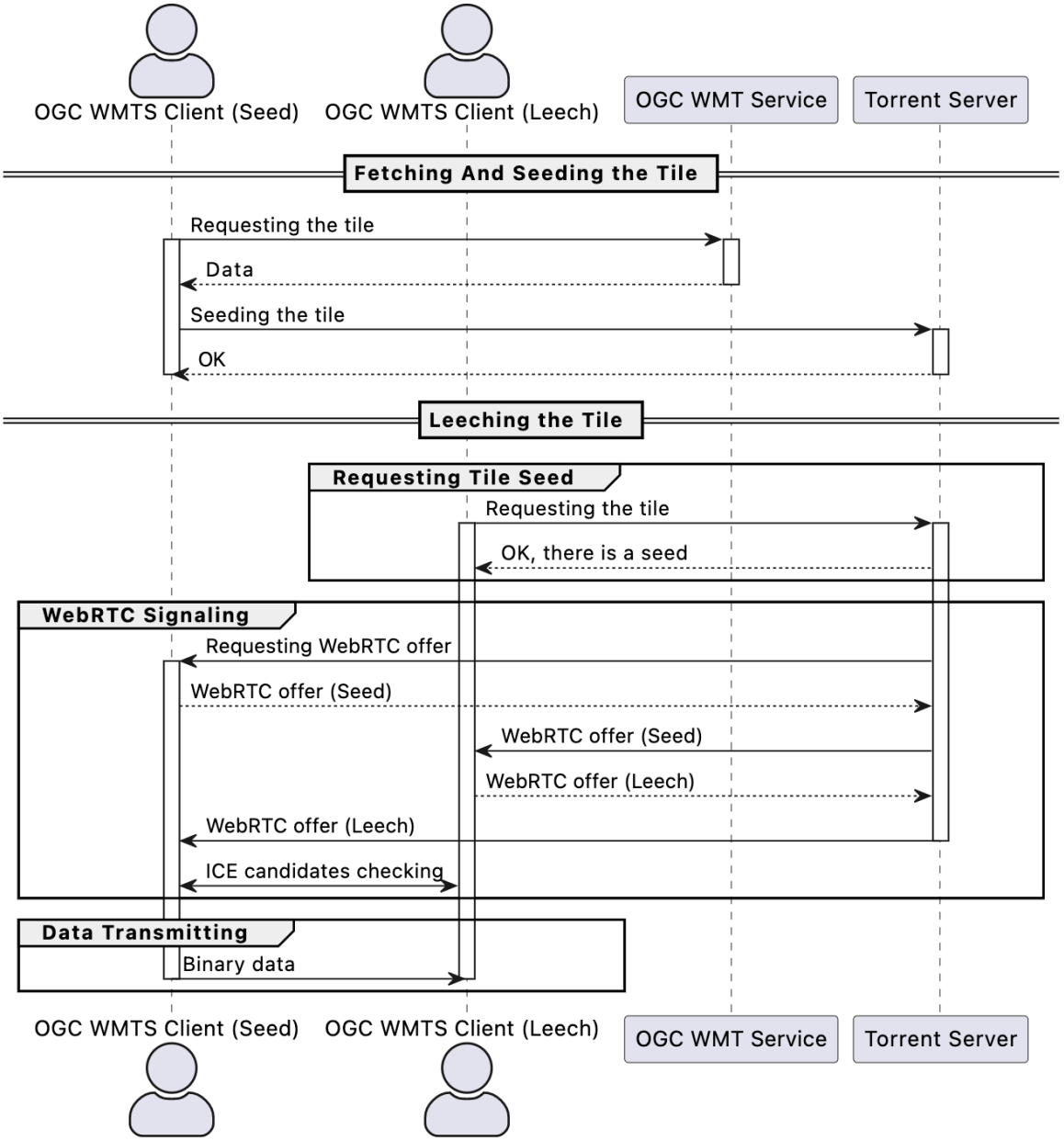

Figure 3.1 Decentralized tiles exchange algorithm

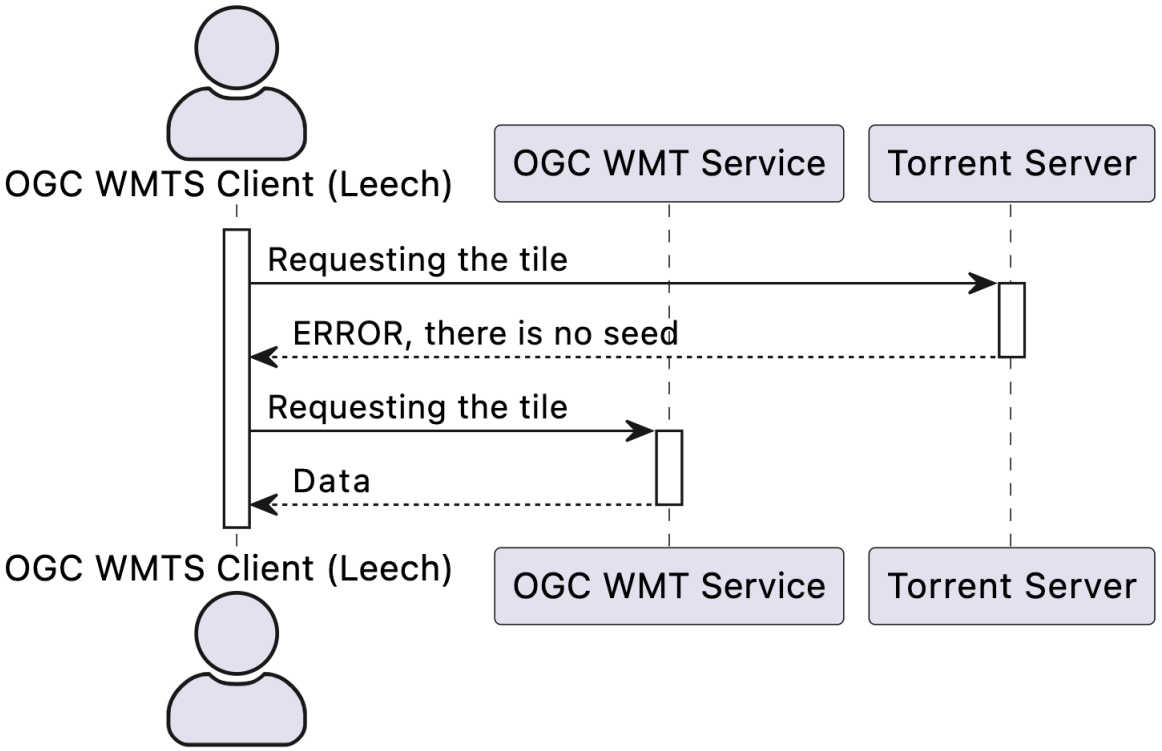

Figure 3.2 Centralized tiles exchange algorithm (unavailable peers)

3.2 Components and stack

Proposed solution implies that the architecture includes four main components. First component is the OGC WMTS server. Such a server should implement the OGC Web Map Tile Service Implementation Standard. For the implementation stage of this research the OGC WMTS service was provided by the supervisor.

Second component is the torrent client. For the research implementation stage JavaScript WebTorrent [50] library was selected. As it mentioned above in section 2.2 the webtorrent provides a well tested and well standardized solution for the utilization of BitTorrent protocol in a Web environment with more than 10 years old maintenance history and more than 400 versions. The analysis of the activity illustrates that the status of the package is Healthy which means that the Web community continues to develop the ecosystem of the webtorrent and propose modifications for better compatibility and stability of the Web-based BitTorrent communications. Figure 3.3 illustrates the activity of the maintenance of the webtorrent made between April 2022 and April 2023.

Third component is the torrent tracker (WMTS tracker). The WMTS tracker should provide a WebSocket gateway for the purpose of WebRTC signaling and sharing information about torrent swarms as well as balancing swarms and handling the information about active and inactive torrent clients. The bittorrent-tracker [55] was selected as a torrent tracker for the implementation stage. This package meets the described above requirements and is included in the webtorrent ecosystem and provides compatibility with webtorrent. The bittorrent-tracker provides an API for the modification of standardized contracts and

implementation of custom functionality for the purpose of extending basic BitTorrent protocol.

Fourth component is the OGC WMTS client. The OGC WMTS client component includes the largest number of changes. However, OGC standards commonly describe protocols for the communication and support different implementations. Considering this, it allows handling the spatial data using different methods. As an OGC WMTS client the Open Layers [30] was selected due to the compatibility with OGC standards and flexibility to customization. As a basic map data provider the Open Street Map was selected due to well compatibility with the Open Layers.

Summing up, the proposed solution stack includes the OGC WMTS service as tiles provider, the bittorrent-tracker as a torrent swarm manager, the webtorrent as a Web-based torrent client, the Open Layers as OGC WMTS client and the Open Street Map as a basic map data provider.

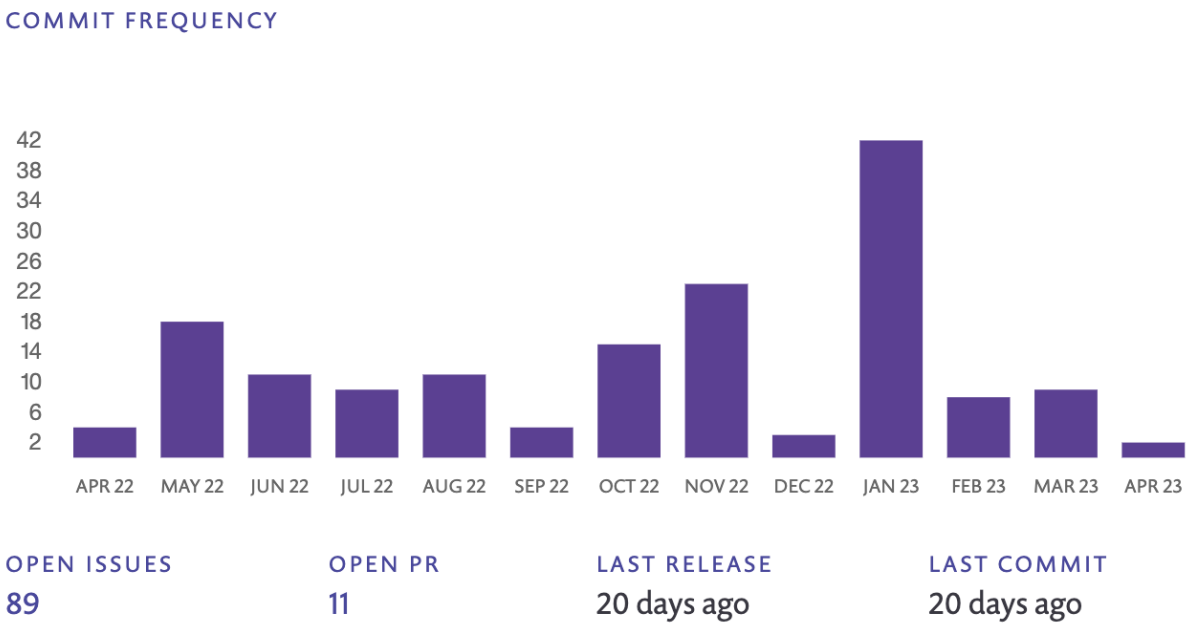

Figure 3.3 Maintenance activity of the webtorrent [54]

3.3 Changes and modifications

The only change in the OGC WMTS server is supporting a new type of ResourcesURL provided by the getCapabilities request which was described above. The ResourcesURL should include “format” property with “application/x-bittorrent” content and “resourcesType” property with “tile” content. The extension of the ResourcesURL should be “.torrent”. The contract of the “template” property should inherit the OGC Web Map Tile Service Implementation Standard. Figure 3.4 illustrates an example of such ResourcesURL. Such a modification implies that the existing WMTSCapabilities.xml will be changed. Proposed solution guarantees backward compatibility and does not require the abandonment of previous versions of the OGC WMTS clients. The OGC WMTS clients which do not support torrent-based communication still can use standardized raster tiles for the rendering

fragments of the map. The master thesis supervisor provided the instance of the OGC WMTS server with the only change — WMTSCapabilities.xml modification.

```
<ResourceURL
  resourceType="tile"
  template="https://cartoweb.wmts.ngi.be/1.0.0/crossborder/default/{TileMatrixSet}/{TileMatrix}/{TileRow}/{TileCol}.torrent"
  format="application/x-bittorrent"
/>
```

Figure 3.4 Example of modified ResourceURL for the torrent-based exchanging map tiles

The WMTS tracker should support caching and storing information about the map tiles using their URIs in contrast to traditional torrent trackers which operate info hashes which were described above. Moreover, the torrent tracker implementation can not be based on standardized torrent magentURIs because the OGC WMTS standard is highly concentrated on using URIs to uniquely identify fragments of the map and guarantee unambiguous identification of the requested tile. Considering this, there is a need to integrate ResourcesURL which interface was described in section 2.1.2. However the webtorrent as well as bittorrent-tracker due to their inheritance of BitTorrent protocol do not allow complete migration from info hashes to the ResourcesURLs. Such changes require significant package modification or implementation of a completely new JavaScript library without webtorrent compatibility. In this way, the required changes were implemented as an extension of the bittorrent-tracker library. Such changes included integration middleware for replacing info hashes with ResourcesURL to provide functionalities for ResourcesURL-based seeding and leeching. The modification algorithm is presented schematically in Figure 3.5. The modification implies introducing Map of the ResourceURLs and InfoHashes accordingly to adapt ResourceURL-based OGC WMTS communication to InfoHash-based communication in BitTorrent protocol.

```
procedure BitTorrentResourceUrlAdapter
    TorrentResourceUrlMap ← Map
    if TorrentTrackerMessage = StartSeedingResourceUrlMessage then
      addResourceUrlToTorrentResourceUrlMap;
    else if TorrentTrackerMessage = StopSeedingResourceUrlMessage then
      removeResourceUrlFromTorrentResourceUrlMap;
    else if TorrentTrackerMessage = StartLeechingResourceUrlMessage then
      findTorrentSeedsByResourceUrlInTorrentResourceUrlMap;
      return torrentSeedsByResourceUrl;
```

Figure 3.5 Schematic representation of the ResourcesURL adapter for the bittorrent-tracker

The main changes were introduced to the OGC WMTS client. Fortunately, the selected OGC WMTS client provided multiple ways to integrate different algorithms for downloading

fragments of the map. In this way, the Open Layers Web Map Tile Service adapter was used to parse and render tiles on the top level of the proposed application. The Open Layers WMTS adapter allows overriding the method for fetching the tiles so the standard client-server approach for fetching tiles was replaced with the decentralized torrent based spatial data exchange approach. In particular when the Open Layers WMTS client tries to fetch the tile data the Open Layers WMTS torrent adapter captures the request and checks the existence of this tile in the torrent swarm. If the tile exists in the torrent swarm, the Open Layers WMTS torrent adapter downloads the tile directly from the appropriate seeder. If not, the tile is downloaded from the OGC WMTS service. Such an approach saves the drawback compatibility with OGC WMTS only clients and partially reduces the workload of the OGC WMTS services. Next figure 3.6 presents the base algorithm for the OGC WMTS client modification.

```
procedure DOWNLOADINGMAPTILES
    if TorrentTrackerHasResourceUrl then
     startDownloadingTileFromTheAvailablePeer;
    else
     startDownloadingTileFromTheOpenGisWmts;
```

Figure 3.6 Schematic representation of the OGC WMTS client for the integration decentralized spatial data exchange approach

3.4 Design and architecture

The proposed stack of technologies require a combination of Cloud-based and Fog-based architecture. In particular initial communication with the OGC WMT service provides map tiles which subsequently are shared using peer-to-peer Web protocols. Such a communication flow satisfies Fog computations paradigm where requests are handled by the users devices and peer-to-peer communications are an important part of the architecture. Figure 3.7 presents top level representation of the components and illustrates the basic participants of the system. In particular the system includes the Fog of peers which are connected with each other using WebRTC and BitTorrent protocols. System includes Cloud-based Web Map Torrent Service and Web Map Tile Service. First manages the peers, second is a source for map tiles. Communication with Web Map Torrent Service carried out using WebSocket while Web Map Tile Service messaging carried out using HTTP.

In detail, the proposed solution requires a Cloud environment. Cloud environment is based on the container orchestration systems such as Google Kubernetes [64] which perfectly cover requirements of the modern Web infrastructure. In this way, multiple instances of the OCG WMT service as well as Web Map Tile Torrent instances are running in the Cloud. When the OGC WMT service has no responsibilities to synchronize state between instances it is an important need for the Web Map Tile Torrent service which requires database connection to manage torrent swarms as well as synchronize state for the WebSocket communication. Besides dependencies needed for the OGC WMT service such as services instances and the database instance it introduces a message broker layer for the synchronization Web Map Tile Torrent. In the modern Cloud infrastructure such a layer includes Redis instance or his alternatives such as KeyDB. When the synchronization layer is

described the load balancing layer should be described. For instance, in the Google Kubernetes such a layer is presented by the Istio [65]. Istio is a proxy server for the routing requests and workload balancing.

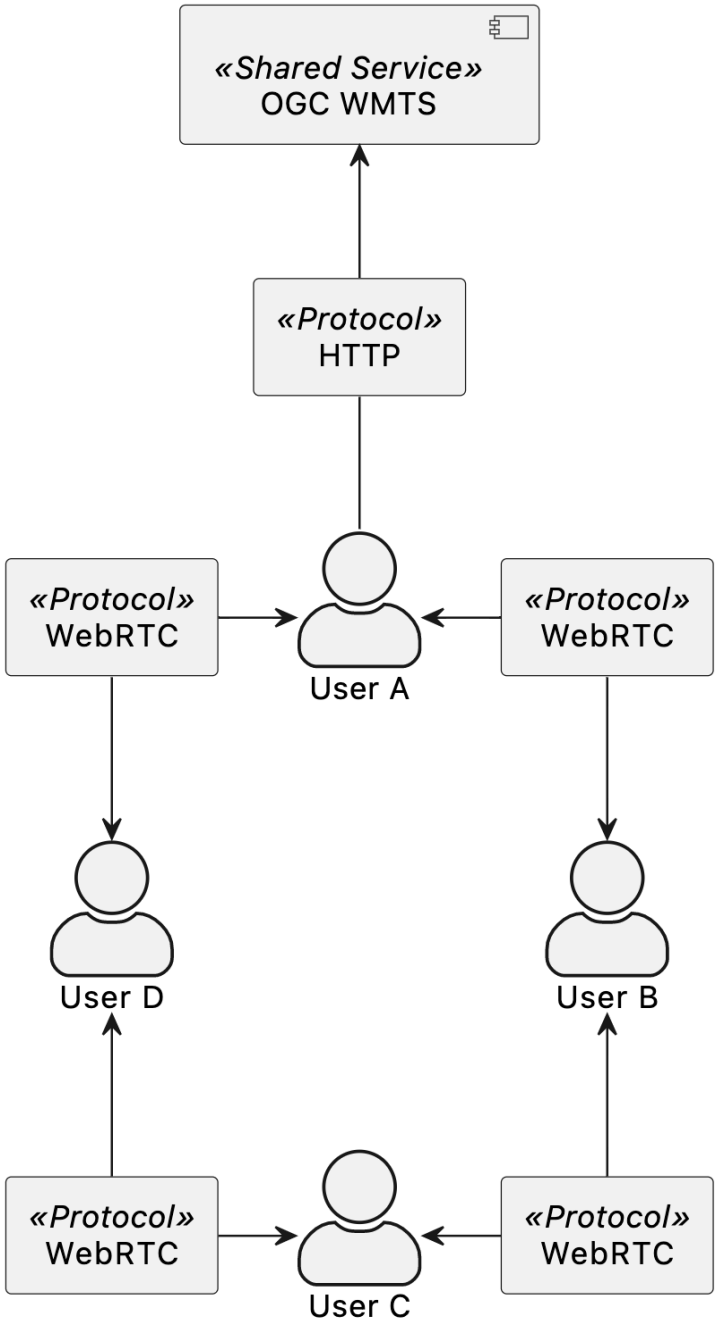

Figure 3.7 Top level components diagram of the proposed system

The Cloud orchestration system provides an API to configure the max workload and manage vertical and horizontal scalability. Such a proxy server allows to distribute the workload between instances and guarantee limited resources utilization by the servers. Next, the peer-to-peer architecture requires introducing STUN and TURN servers for resolving NAT traversal. Unfortunately, introducing the TURN server is rolling us back to the client-server architecture when the user establishes the connection with a server and the

communication is carried out exclusively using the server. However, as it mentioned above the TURN servers are needed only in cases of incompatible NAT such as symmetric NAT. Considering this, the decentralized approach is relevant only for the users which have the compatible NAT. Summing up, the infrastructure for the WebRTC communication should include both STUN and TURN servers as well as signaling server. Signaling and Web torrent server may be combined with each other due to close responsibilities in Web-based BitTorrent infrastructure. Figure 3.8 illustrates the UML components diagram of the proposed solution. The diagram presents two types of packages. First presents a centralized element of the system and is based on the existing OGC infrastructure while second is responsible for decentralized communication between clients.

Hence, infrastructure dependencies are grouped by four functional layers. First layer is the client layer which configures requests and handles HTTP, WebSocket and WebRTC connections to download and share map tiles. Second layer is a gateway layer whose main purpose is to balance workload and route requests within the cluster. Third layer is a service layer. Service layer handles users requests and is responsible for the map tiles downloading as well as managing BitTorrent communications. Fourth layer is a database layer. The databases are represented by a message broker database for the WebRTC signaling and BitTorrent servers and OGC WMT service database for storing map tiles. Centralized approach implies that the whole layers involved to the communication while decentralized allow partially exclude second, third and fourth layer from the communication chain. Considering this, it allows to partially reduce workload of the centralized geographical information systems and easy to introduce such an architecture due to modularity of the proposed decentralized approach.

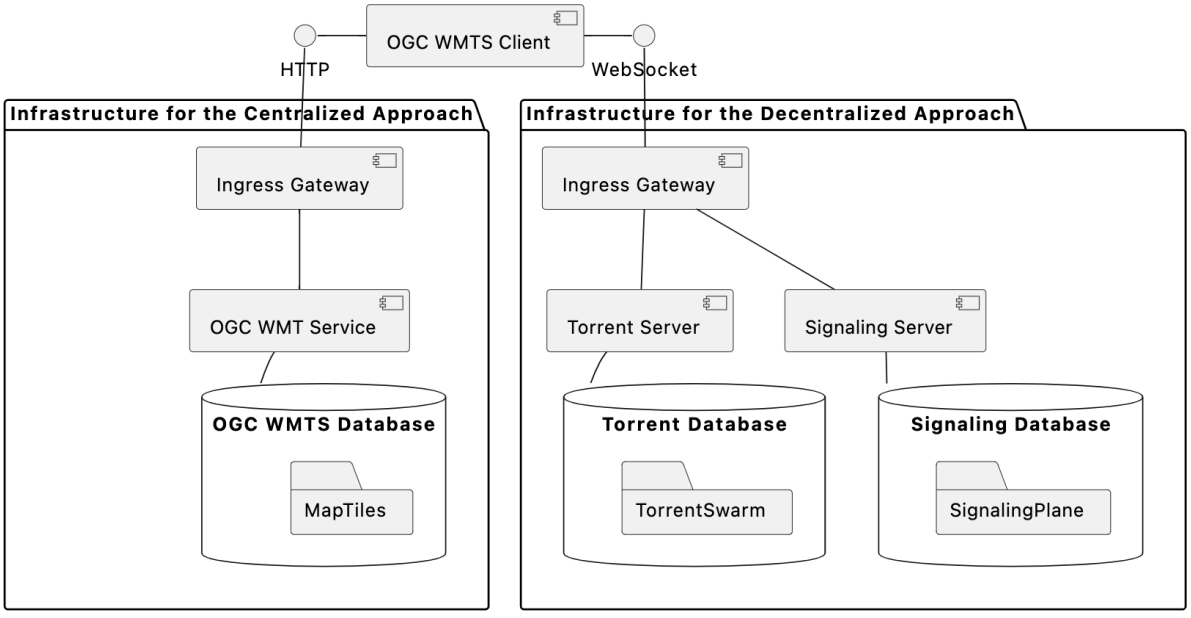

Figure 3.8 UML components diagram of the proposed architecture

3.5 Resolved challenges

The prototype was successfully deployed. However, next challenges were collected during the testing of the functionality step:

- Crypto.subtle module which is used by the webtorrent library to hashing file content is not available in the http browser context. It is available only in the https browser context;
- Webtorrent library uses streams API, globalThis object and path module which are available only in Node.js context;
- Open Layers optionsFromCapabilities function does not support torrent ResourceURL that means layers with torrent sources is not available in native parsing capabilities algorithm provided by Open Layers;
- Delay of 5 seconds during initialization of many PeerConnections for the tiles downloading using BitTorrent in Google Chrome.

First issue is connected with restrictions of the browser API. In particular, Crypto.subtle module is available only in the https:// browser context [57]. Considering this, the system should be deployed using https:// and signed using an SSL certificate. For the tests a local certificate was used.

Next issue is connected with common modules of the webtorrent library. Webtorrent supports both browser and server platforms. However the browser implementation requires polyfilling Node.js methods. For instance, globalThis object, path module, streams API have to be pollyfilled. Such polyfills were easily searched using stackoverflow.com but should be mentioned in the challenges section due to their importance for the application bootstrap.

Next issue is connected with the Open Layers API which provides methods for fetching and parsing getCapabilities endpoint response. The response of the getCapabilities is an XML document. The optionsFromCapabilities generates options for the Open Layers WMTS layer. Obviously, the BitTorrent-based spatial data exchange is not supported out of the box by the Open Layers and optionsFromCapabilities does not support torrent layers. Considering this, the source code of the optionsFromCapabilities was copied to the work directory and necessary changes were implemented. Subsequently, optionsFromCapabilities was substituted in the Open Layers source code.

The last issue is connected with delay in Google Chrome browser. During the tests it was noticed that many PeerConnections are opened with a delay of 5 seconds when webtorrent opens many connections to fetch tiles. The issue was investigated and the issue 825576 [56] was found. According to this issue Chrome is destroying many connections to optimize tabs workload. In particular, the “Cannot create so many PeerConnections” exceptions will be thrown. Webtorrent developers knowing this restricted opening multiple PeerConnections Google Chrome and added Pool to control the connections queue. It leads to significant delay during opening multiple PeerConnections. Hopefully, this bug of the Google Chrome should be resolved and synthetic restrictions will be canceled.

4. Evaluation of the Proposed Approaches

4.1 Evaluation

The evaluation methodology is based on IEEE Standard for Software Test Documentation [61]. The IEEE Standard for Software Test Documentation was adapted to the master thesis needs because the proposed approach does not cover whole requirements mentioned in the IEEE standard. Considering this, next sections were taken from the standard: features to be tested, features not to be tested, approach refinements, item pass/fail criteria, testing tasks, environmental needs. Finally, results are analyzed and a summary of the efficiency evaluation stage is made.

4.1.1 Features to be tested

Next features were tested to ensure sustainability and performance of the proposed approach:

- Downloading map tiles from the OGC WMTS;
- Caching downloaded map tiles;
- Seeding downloaded map tiles;
- Leeching map tiles using BitTorrent protocol;
- Downloading map tiles from the OGC WMTS if seeds are not exist;
- Rendering map tiles downloaded using both OGC WMTS and BitTorrent;
- Concurrent downloading and rendering map tiles using both OGC WMTS and BitTorrent;
- Downloading map tiles using OGC WMTS or BitTorrent when the user is zooming the map.

4.1.2 Features not to be tested

Next features were not tested due to out of master thesis scope context:

- Downloading map tiles using NAT traversal techniques, for instance using TURN servers;
- Downloading binary-based map tiles such as .tiff map tiles which requires exchanging of the binary pieces.

4.1.3 Approach refinements

Preparation stage should include browser installation and checking NAT types. The clients should follow the next NAT types: Full Cone, Restricted Cone, Port Restricted Cone. Prototype does not support WebView [59] so test cases must be running up in the browser tab. Incognito tab is also allowable. The browser tab must get access to the LocalStorage, SessionStorage, XMLHttpRequest and Crypto browsers modules. The prototype should be tested using only valid ResourceURL. Templates of the ResourceURL must follow OpenGIS Web Map Tile Service Implementation Standard [32].

Next requirement should be taken into account. The prototype supports only one region on the map. The prototype collects next metrics in order to evaluate test results: downloading time in milliseconds per every map tile, the mean downloading time for all tiles. The prototype provides GUI to illustrate collected statistics. First GUI dashboard is the percentage of data source types such as torrent or WMTS. Second GUI dashboard is the downloading time per every tile in milliseconds bar chart. Third chart illustrates the mean downloading time for all tiles grouped by data source types. All test scenarios should be repeated 10 times to collect statistics for the extra analysis.

4.1.4 Items pass/fail criteria

The prototype must satisfy the following requirements:

- Downloaded tiles should be rendered on the map;
- The tiles should be downloaded when the user are zooming the map;
- The map tiles should be downloaded using OGC WMT or Torrent servers.

4.1.5 Environmental characteristics

The prototype was deployed to the cloud infrastructure using the Yandex Cloud [58] virtual machine. The system was bootstrapped using docker containers and started up using docker-compose. Next hardware resources were granted:

OS	Platform	vCPU	RAM	HDD	Max IOPS	Max bandwidth
Linux	Intel Cascade Lake	2	2GB	30GB	300	30 Mbit/sec

The prototype was tested using next operation systems:

- macOS Ventura 13.2.1;
- Windows 11, Home Edition.

Next browsers were used to test the prototype:

- macOS:
  - Google Chrome, version 112.0.5615.137;
  - Firefox, version 113;
- Windows:
  - Google Chrome, version 113.0.5672.93;
  - Firefox, version 114.

Next devices were used to test the prototype:

OS	CPU	RAM	GPU
macOS Ventura 13.2.1	2,4 GHz Intel Core i9	32 GB 2667 MHz DDR4	Intel UHD Graphics 630 1536 MB
Windows 11, Home Edition	Intel(R) Core(TM) i3-10100 CPU @ 3.60GHz	8 GB DDR4	Intel(R) UHD Graphics 630

4.1.6 Test scenarios

First scenario verifies that the map tiles are downloading using only OGC WMT service:

1. User opens browser tab;
2. User waits until all map tiles are downloaded and rendered on the map.

Expected result is:

- Tiles are rendered on the map;
- GUI illustrates that all map tiles are downloaded from the OGC WMTS.

Second scenario verifies downloading tiles using both OGC WMTS and BitTorrent protocol. Next preconditions should be satisfied. There are two users. Both users are connected to the same local network, for instance to the same Wi-Fi router. First user is a seeder. Second user is a leecher. Next scenarios should be executed sequentially. Seeder executes scenario before leecher.

Next scenario relevant for both users — seeder and leecher:

1. User opens browser tab;
2. User waits until all map tiles are downloaded and rendered on the map.

Expected result is:

- Tiles are rendered on the map;
- GUI illustrates that all map tiles are downloaded using BitTorrent protocol.

4.1.7 Test summary

Figure 4.1 illustrates the mean time per tile in milliseconds for downloading tiles from the OGC WMTS while figure 4.2 illustrates the mean time per tile in milliseconds for downloading tiles using BitTorrent. Figure 4.3 illustrates the mean time per tile in milliseconds divided into four main steps of the BitTorrent protocol: initialization or identification available peer, initialization peer connection, hashing file content for identification pieces of the tiles and transforming tile data from arrayBuffer to Blob.

Next statistics were aggregated. The mean time for downloading map tiles using OGC WMTS was approximately 45 milliseconds. The mean time for downloading map tiles using BitTorrent protocol was approximately 133 milliseconds. The mean time for the identification available peer was 29 milliseconds. The mean time for the initiation peer connection was 80 milliseconds. The mean time for hashing pieces of the tiles was 3 milliseconds. The mean time for transforming tile data from arrayBuffer to Blob was 2 milliseconds. It is obvious that the establishing peer connection significantly influences the mean time for downloading map tiles. Considering this, next statistics after initialization of the peer connection should be presented. The mean time for downloading map tiles using BitTorrent protocol was approximately 16 milliseconds. The mean time for the identification available peer was 9 milliseconds.

According to the test results both steps of the BitTorrent protocol identification available peer and initialization peer connection were most time expensive for the torrent-based approach while hashing and transforming took insignificant time in the exchanging map tiles process. In the next analysis hashing and transforming will be ignored due to their insignificance for the proposed approach.

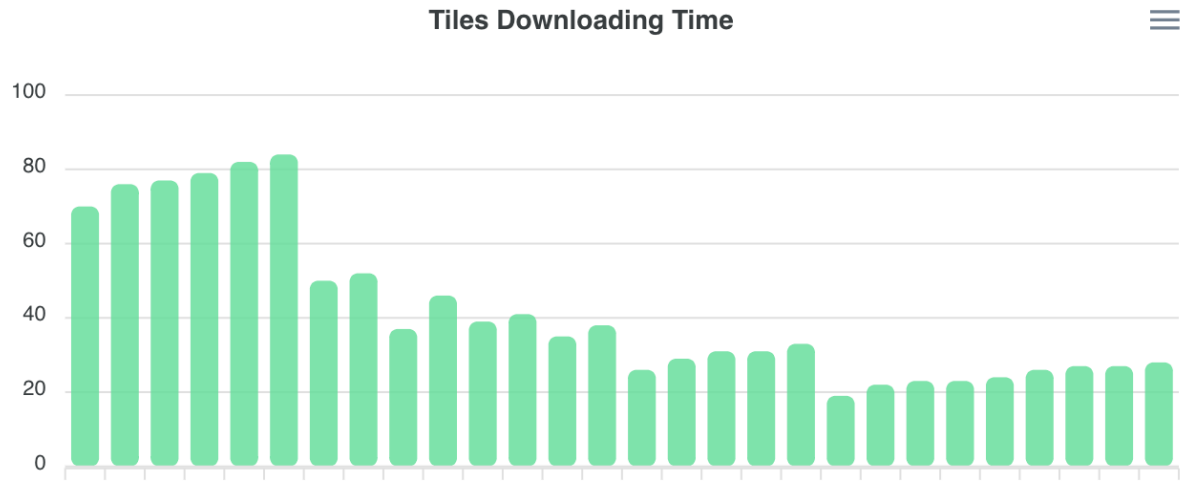

Figure 4.1 Mean time in milliseconds for downloading map tiles using OGC WMTS

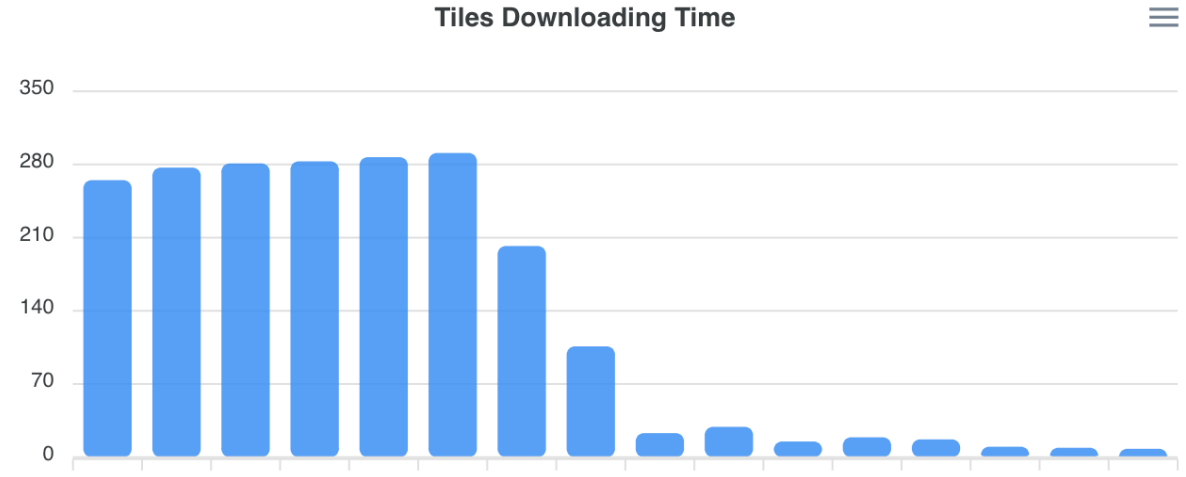

Figure 4.2 Mean time in milliseconds for downloading map tiles using BitTorrent

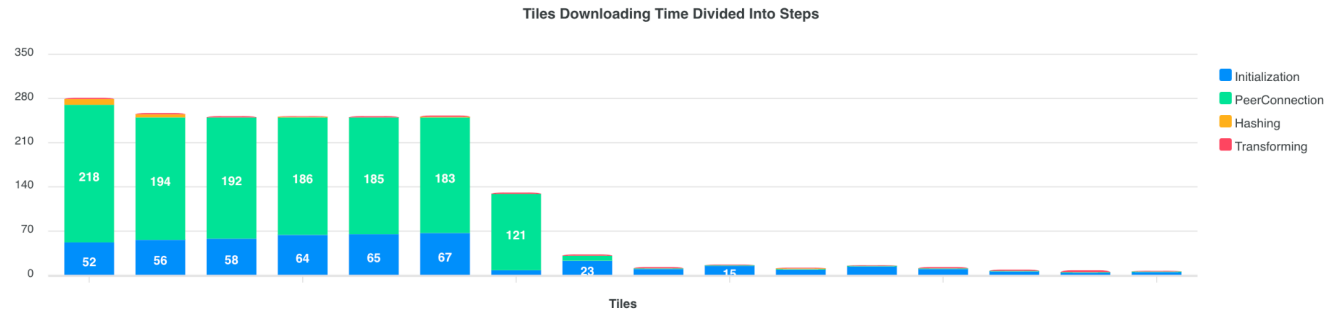

Figure 4.3 Mean time in milliseconds divided into four main steps of the BitTorrent protocol

4.2 Discussion

Collected statistics illustrate that the proposed approach for decentralized exchanging spatial data using Web technologies satisfies functional requirements. Prototype was successfully deployed to the Cloud infrastructure and all described scenarios were successfully tested. The hypothesis that it is possible to partially exclude OGC WMTS from the communication chain using Web implementation of the BitTorrent protocol was proved. Both seeders and leechers successfully downloaded the map tiles and rendered them on the map. However, collected during the test stage statistics presents ambiguous conclusions.

On the one hand, the proposed approach demonstrates technical issues which are connected with the Web environment limitations such as peculiarities of the WebRTC in the different browsers, for instance max WebRTC connections per tab, NAT traversal which technical difficulties were described in the previous sections and remains important and unresolved issue. Moreover, testing scenarios helped identify and measure challenges connected with WebRTC performance. In particular, extra costs for establishing WebRTC connections. According to the statistics, such delays during establishing the connection partially devalue benefits of the proposed approach.

On the other hand, efficiency evaluation of the second scenario illustrates that the mean time for downloading tiles was reduced approximately three times from 284 milliseconds to 87 milliseconds if comparing statistics after establishing WebRTC connection. Moreover, according to the proposed approach the OGC WMTS was partially excluded from the communications. Such approach undoubtedly allows significantly reducing workload of the OGC WMTS due to redirecting requests from OGC WMTS to peers. On the assumption that the WebRTC has the possibility to identify and establish the most efficient connection type using ICE candidates checking steps, it is proved that the close BitTorrent clients location in the compartment with local network utilization built an excellent environment to establish and maintain peer-to-peer communications.

However, the algorithms of the BitTorrent implies that the pieces of the files are sent from the multiple clients in order to distribute the workload and ensure efficient network utilization. Considering this, transmitting many single tiles using BitTorrent protocol may lead to extra time costs and browser resources consumption due to the need to create and maintain many WebRTC connections. Despite the fact that Google Chrome supports 500

PeerConnections [60] per tab, every connection consumes browser resources and reduces network bandwidth that leads to stagnation of the browser performance.

4.3 Future directions

Described above challenges create the basis for proposition next approaches for decentralized exchanging spatial data. On the assumption that establishing WebRTC connections takes significant time, the next hypothesis was formulated and analyzed in this section.

It is expected that combining multiple requests together and fetching a single package of the map tiles instead of concurrently fetching multiple tiles may partially resolve the issue which is connected with extra delay during establishing many WebRTC connections. In this context, the application should analyze for instance rendered on the map region and request a package of the map tiles. Considering this, BitTorrent establishes a WebRTC connection with only one seed and excludes repeatable delays for the establishing many WebRTC connections. Figure 4.4 illustrates a modified approach for map tiles exchange using BitTorrent protocol. In accordance with such an approach users may exchange groups of the map tiles packaged by region.

Next bottle neck in the proposed algorithm is the webtorrent NPM package. It was detected that the webtorrent library due to cross platform nature faces multiple challenges in a browser environment. For instance the webtorrent github repository has an unclosed issue [66] which is connected with a hardcoded “trickle” parameter. The webtorrent creates a new PeerConnection and specifies “trickle” parameter as true without any possibility to modify this value. It is investigated that hardcoded “trickle” parameter leads to extra delay during establishing WebRTC connection. This is acceptable for some users but absolutely unacceptable especially for the proposed approach when multiple WebRTC connections are created to transmit many tiles through the WebRTC Data Channel. There are many webtorrent issues which are unfortunately still unresolved. All in all, there is a hypothesis that the migration from webtorrent to native WebRTC browser API in case of single map tile transmitting can help to avoid issues which are connected with Web-based BitTorrent implementation. Such a hypothesis should be clearly analyzed in the next research because proposed architecture is fully based on the BitTorrent protocol.

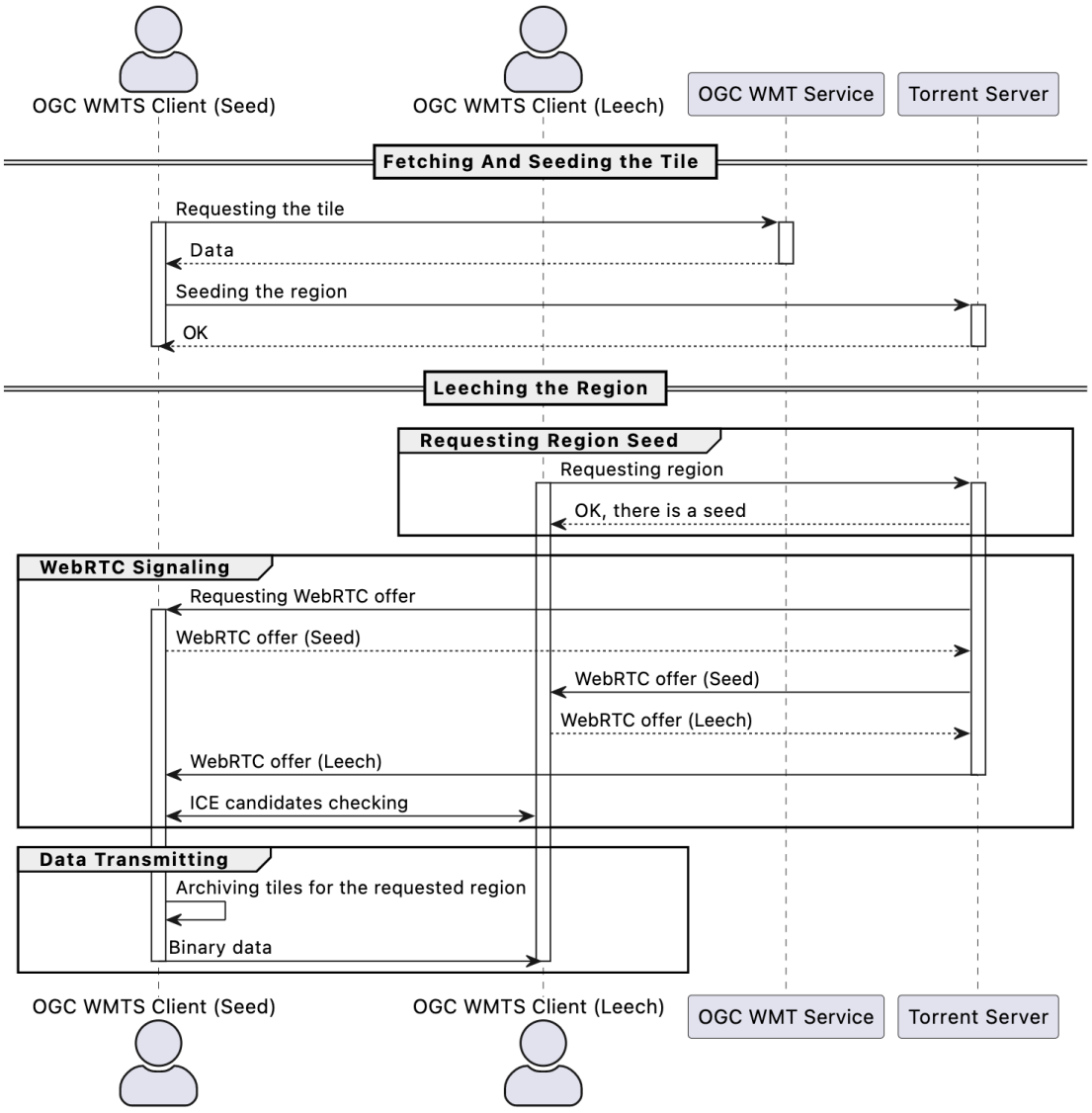

Figure 4.4 Algorithm for exchanging packed by region map tiles using BitTorrent protocol

Conclusion

In conclusion the results of the preliminary research, existing systems analysis and implementation stage are analyzed in order to highlight limitations, constraints and benefits made by introducing decentralized Web protocols to the existing exchanging spatial data approaches. The research made in Introduction and Existing systems analysis sections illustrated that in the modern GIS infrastructure only centralized solutions are presented. Despite the fact that cloud computing resolves many issues of the client-server architecture such as vertical and horizontal scaling, elasticity, network access, the developers still face challenges which are connected with a centralized approach for processing and exchanging spatial data for instance data storage and organization, access to big data, data processing pipelines. According to modern research these issues were identified as one of the most important Earth observation big data challenges.

Fortunately, the Fog computing paradigm offers a mixed approach for processing and exchanging data. Fog computations utilizes Cloud infrastructure as a top level data processing layer to process complex tasks while users hardware is utilized as a low level processing layer for processing lightweight tasks. According to the research, Fog architecture partially mitigates Cloud computing limitations such as single point of failure, network workload and latency, distance to the users, storage capacity, costs for data transmitting and real-time applications handling. However such an architecture introduces new challenges. For instance, node management, data security, NAT traversal, users hardware limitations as well as other conceptual and technical issues. Meanwhile, papers illustrate that due to the decentralized nature of the Fog computing paradigm the implication of the Fog computing to the existing GIS systems may significantly reduce workload and resolve mentioned above challenges of the Cloud architecture such as storage and access to big data.

Fog computations are based on the existing and widely used set of Web technologies that made it possible to develop real-time applications. In particular, Google, Facebook, Discord, Snapchat and Cisco have already developed peer-to-peer applications based on the decentralized Web protocols. It means that such decentralized protocols as WebRTC and BitTorrent now have excellent support by the browsers and simplify developers tasks. In particular, WebRTC orchestrates a set of existing and well tested protocols to handle real-time data transmitting. For instance TCP/UDP protocols are utilized as low level messaging protocols, SRTP/RTP protocols are utilized to handle media streams while DTLS/SCTP protocols to handle data transmitting using WebRTC DataChannel. Moreover, introducing WebRTC made it possible to adapt already existing decentralized technologies such as BitTorrent protocol to the modern Web environment.

On the assumption that BitTorrent presents a set of technologies to manage decentralized data exchange such as clients workload management, distributing clients workload and management of exchanging large amounts of data, the next hypothesis was formulated after research of the existing technologies. It is expected that integration of BitTorrent protocol to existing OGC Web services can significantly reduce workload of the OGC Web services and introduce decentralized approach for exchanging spatial data to the existing GIS systems. In order to prove such a hypothesis the prototype of the decentralized GIS application was designed and developed in the context of this master thesis. Functional testing of the implemented prototype proved that the proposed algorithms for decentralized exchange spatial data are valid and satisfies functional requirements while performance testing identified important issues of the proposed approach such as time costs for establishing WebRTC connections and concurrent PeerConnections limit in the modern browsers. As a

result of the evaluation stage, next recommendations were formulated. First of all, the time for downloading spatial data using BitTorrent was much lower than time for downloading spatial data using OGC WMT service in case of transmitting large amounts of data. Collected during the testing statistics illustrate that BitTorrent protocol provides algorithms for workload distribution which allow to reduce time to transmit data and exclude torrent clients overload. At the same time extra costs for establishing many peer-to-peer WebRTC connections in case of concurrent fetch spatial data significantly increase downloading time in the case of utilizing many different torrent clients to fetch the data. Considering this, the approach for decentralized exchanging spatial data based on WebRTC and BitTorrent protocols which were described in the previous sections should be modified. It is expected that combining spatial data by the regions and package-based transmitting using BitTorrent will be more efficient. However, it needs extra research which is out of scope of this master thesis.

Putting it all together, it is proved that the integration of the modern decentralized Web protocols to the existing centralized approaches reduces time for downloading spatial data in particular use cases, decreases workload of the existing centralized Web services and generates perspectives for the next generation of the GIS systems based on peer-to-peer decentralized Web protocols.

Bibliography

[1] Aaron Weiss. Computing in the Clouds. netWorker - Cloud computing, 11(4):16–25, December 2007.

[2] Aaron Weiss. Computing in the Clouds. netWorker - Cloud computing, 11(4):16–25, December 2007. URL: https://files.eric.ed.gov/fulltext/EJ867963.pdf. [Online; accessed 26 March 2023].

[3] Agrawal, S., Gupta, R. D. Web GIS and its architecture: a review. Arabian Journal of Geosciences 10. URL https://link.springer.com/article/10.1007/s12517-017-3296-2. 2017. [Online; accessed 26 March 2023].

[4] AL-Chief, Financial., Sabri, N., Latiff, N., Malik, N., Abbas, M., Al bader, A., Mohammed M., AL-Haddad, R., Salman, Y., Ghani, M., Obaid, O. Performance Comparison between TCP and UDP Protocols in Different Simulation Scenarios. URL: https://www.sciencepubco.com/index.php/ijet/article/download/23739/11888. [Online; accessed 26 March 2023].

[5] Alvestrand, H. (2016a). Overview: Real Time Protocols for Browser-based Applications, draft-ietf-rtcweb-overview-15. URL: https://tools.ietf.org/html/draft-ietf-rtcweb-overview-15. [Online; accessed 26 March 2023].

[6] Amazon Web Services (AWS). Case Studies and Customer Success with the AWS Cloud - Why Customers choose AWS.

[7] Amir Vahid Dastjerdi, Harshit Gupta, Rodrigo N. Calheiros, Soumya K. Ghosh, and Rajkumar Buyya. Fog Computing: Principles, Architectures, and Applications. 2016. URL: https://arxiv.org/pdf/1601.02752.pdf. [Online; accessed 26 March 2023].

[8] Big Earth Observation Data Analytics: Matching Requirements to System Architectures. 2016. URL: http://plutao.sid.inpe.br/col/sid.inpe.br/plutao/2016/12.05.19.23.08/doc/camara_big.pdf. [Online; accessed 26 March 2023].

[9] Bonomi, F., Milito, R., Zhu, J., Addepalli, S. Fog Computing and Its Role in the Internet of Things. 2012. URL: https://conferences.sigcomm.org/sigcomm/2012/paper/mcc/p13.pdf. [Online; accessed 26 March 2023].

[10] Bryan Hayes. Cloud computing. Communications of the ACM, 51(7), July 2008.

[11] C. Portele and S. Sankaran. GeoServices REST API. Open GIS Consortium, 2012. URL: http://www.opengeospatial.org/standards/requests/89. [Online; accessed 26 March 2023].

[12] C. Portele. OpenGIS®Geography Markup Language (GML) Encoding Standard 3.2.1. Open GIS Consortium, 2007. URL: http://www.opengeospatial.org/ standards/gml. [Online; accessed 26 March 2023].

[13] Can I use – WebRTC. URL: https://caniuse.com/?search=webrtc. [Online; accessed 26 March 2023].

[14] Christian Baun, Marcel Kunze, Jens Nimis, and Stefan Tai. Cloud Computing: Web-Based Dynamic IT Services. Springer Publishing Company, Incorporated, 1st edition, 2011.

[15] Gajbhiye, A., Mohan, K. Cloud Computing: Need, Enabling Technology, Architecture, Advantages and Challenges. 2014. URL: https://ieeexplore.ieee.org/document/6932989. [Online; accessed 26 March 2023].

[16] Gandomi, A.; Haider, M.; Beyond the hype: Big data concepts, methods, and analytics. 2015. URL: https://www.sciencedirect.com/science/article/pii/S0268401214001066. [Online; accessed 26 March 2023].

[17] Global Web Real-Time Communication (WebRTC) Industry. URL: https://www.reportlinker.com/p06324849/Global-Web-Real-Time-Communication-WebRTC-Industry.html?utm_source=GNW. [Online; accessed 26 March 2023].

[18] Hansen, M. S., Potapov, P. V., Moore, R., Hancher, M., Turubanova, S. A., Tyukavina, A., & Townshend, J. R. G., 2013. High-resolution global maps of 21st-century forest cover change. Science, 342(6160), 850-853.

[19] I. W. Stats, "Internet world stats". URL: https://www.internetworldstats.com. [Online; accessed 26 March 2023].

[20] I. W. Stats, "Internet world stats". URL: https://www.statista.com/statistics/265149/internet-penetration-rate-by-region/. [Online; accessed 26 March 2023].

[21] Introduction to ArcGIS GeoEvent Server. URL: https://enterprise.arcgis.com/en/geoevent/latest/get-started/what-is-arcgis-geoevent-server.htm. [Online; accessed 26 March 2023].

[22] ISO/IEC 25010:2011Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. URL: https://www.iso.org/standard/35733.html. [Online; accessed 26 March 2023].

[23] J. de La Beaujardière. OpenGIS® Web Map Server Interface Implementation Specification Revision 1.3.0. Open GIS Consortium, 2006. URL: http://www.opengeospatial.org/standards/wms. [Online; accessed 26 March 2023].

[24] John Leslie King. Centralized Versus Decentralized Computing: Organizational Considerations and Management Options. ACM Computing Surveys, 15(4):319–349, December 1983.

[25] Johnsen, J., Karlsen, L. Peer-to-peer networking with BitTorrent. URL: https://web.cs.ucla.edu/classes/cs217/05BitTorrent.pdf. [Online; accessed 26 March 2023].

[26] Luis M. Vaquero and Luis Rodero-Merino. Finding Your Way in the Fog: Towards a Comprehensive Definition of Fog Computing. ACM SIGCOMM Computer Communication Review, 44(5):27–32, October 2014.

[27] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A View of Cloud Computing. Communications of the ACM, 53(4):50–58, April 2010.

[28] OGC. Web services context document (OWS Context). URL: https://www.ogc.org/standards/owc. [Online; accessed 26 March 2023].

[29] Open Geospatial Consortium. URL: https://www.ogc.org. [Online; accessed 26 March 2023].

[30] Open Layers. URL: https://openlayers.org. [Online; accessed 26 March 2023].

[31] OpenFog Consortium Architecture Working Group. OpenFog Reference Architecture for Fog Computing. 2017. URL: https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf. [Online; accessed 26 March 2023].

[32] OpenGIS Web Map Tile Service Implementation Standard. URL: https://www.ogc.org/standard/wmts/. [Online; accessed 26 March 2023].

[33] OpenGIS Web Services Common Implementation Specification. URL: https://www.ogc.org/standard/common. [Online; accessed 26 March 2023].

[34] OSGeo Tile Map Service Specification. URL: https://wiki.osgeo.org/wiki/Tile_Map_Service_Specification. [Online; accessed 26 March 2023].

[35] P. Zhao, G. Yu, and L. Di. Geospatial web services. Emerging Spatial Information Systems and Applications, 2007.

[36] Paradies, M.; Schindler, S.; Kiemle, S.; Mikusch, E. Large-Scale Data Management for Earth Observation Data—Challenges and Opportunities. 2018. URL: https://ceur-ws.org/Vol-2191/paper33.pdf. [Online; accessed 26 March 2023].

[37] Perkins, C., Westerlund, M., & Ott, J. (2016). Web Real-Time Communication (WebRTC): Media Transport and Use of RTP, draft-ietf-rtcweb-rtp-usage-26. Retrieved from https://tools.ietf.org/html/draft-ietf-rtcweb-rtp-usage-26. [Online; accessed 26 March 2023].

[38] Peter Lor: International and Comparative Librarianship, Chapter 4 draft 2011-04-20. URL: https://pjlor.files.wordpress.com/2010/07/book-front-matter.pdf. [Online; accessed 26 March 2023].

[39] Peter M. Mell and Timothy Grance. The NIST Definition of Cloud Computing. Technical report, National Institute of Standards and Technology, Gaithersburg, MD, USA, September 2011. URL: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf. [Online; accessed 26 March 2023].

[40] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud Computing: State-Of-the-Art and Research Challenges. Journal of Internet Services and Applications, 1(1):7–18, April 2010.

[41] Quental, N., Goncalves, P.. Mobile-BitTorrent: a BitTorrent Extension for MANETs. URL: https://www.cin.ufpe.br/~pasg/gpublications/ncq-retec.pdf. [Online; accessed 26 March 2023].

[42] Sandvine. The global Internet phenomena report. 2022. URL: https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/2022/Phenomena%20Reports/GIPR%202022/Sandvine%20GIPR%20January%202022.pdf. [Online; accessed 26 March 2023].

[43] Schut P. OpenGIS® Web Processing Service Revision 1.0.0. Open GIS Consortium, 2007. URL: http://www.opengeospatial.org/standards/wps. [Online; accessed 26 March 2023].

[44] Self-Tuning Distributed Hash Table (DHT) for Resource Location And Discovery. URL: https://datatracker.ietf.org/doc/rfc7363. [Online; accessed 26 March 2023].

[45] simple-peer. URL: https://github.com/feross/simple-peer. [Online; accessed 26 March 2023].

[46] Soille, P.; Burger, A.; De Marchi, D.; Kempeneers, P.; Rodriguez, D.; Syrris, V.; Vasilev, V. A versatile data-intensive computing platform for information retrieval from big geospatial data.

[47] Umesh, A.. Performance analysis of transmission protocols for H.265 encoder. URL: https://www.diva-portal.org/smash/get/diva2:865571/FULLTEXT01.pdf. [Online; accessed 26 March 2023].

[48] Uniform Resource Identifier (URI): Generic Syntax. URL: https://datatracker.ietf.org/doc/html/rfc3986#section-3.4. [Online; accessed 26 March 2023].

[49] WebRTC For the Curious. URL: https://webrtcforthecurious.com/docs/webrtc-for-the-curious.pdf. [Online; accessed 26 March 2023].

[50] webtorrent. URL: https://github.com/webtorrent/webtorrent. [Online; accessed 26 March 2023].

[51] Zhi-Hui Zhan, Xiao-Fang Liu, Yue-Jiao Gong, Jun Zhang, Henry Shu-Hung Chung, and Yun Li. Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches. ACM Computing Surveys, 47(4):63:1–63:33, July 2015.

[52] Walter de Donato, Alessio Botta, Valerio Persico, and Antonio Pescape. Integration of Cloud Computing and Internet of Things: A Survey. Future Generation Computer Systems, 56:684 – 700, 2016. URL: http://wpage.unina.it/valerio.persico/pubs/botta2016fgcs.pdf. [Online; accessed 26 March 2023].

[53] G. Percivall, C. Reed, L. Leinenweber, C. Tucker, and T. Cary. OpenGIS®Reference Model 2.1. Open GIS Consortium, 2011. URL http://www.opengeospatial.org/standards/orm. [Online; accessed 26 March 2023]

[54] Analysis of the maintenance activity for the webtorrent NPM library. URL: https://snyk.io/advisor/npm-package/webtorrent. [Online; accessed 23 April 2023]

[55] NPM package bittorrent-tracker. URL: https://www.npmjs.com/package/bittorrent-tracker. [Online; accessed 23 April 2023]

[56] Issue 825576: RTCPeerConnection objects are released too slowly and reallocating causes exception: Cannot create so many PeerConnections. URL: https://bugs.chromium.org/p/chromium/issues/detail?id=825576. [Online; accessed 23 April 2023]

[57] MDN Web Docs. Crypto: subtle property. URL: https://developer.mozilla.org/en-US/docs/Web/API/Crypto/subtle. [Online; accessed 23 April 2023]

[58] Yandex Cloud. Cloud computing provider. URL: https://cloud.yandex.ru/. [Online; accessed 23 April 2023]

[59] Android developers. Docs. WebView. URL: https://developer.android.com/reference/android/webkit/WebView. [Online; accessed 23 April 2023]

[60] Chromium. Source code. URL: https://chromium.googlesource.com/chromium/src/third_party/+/master/blink/renderer/modules/peerconnection/rtc_peer_connection.cc#160. [Online; accessed 23 April 2023]

[61] IEEE Standard for Software Test Documentation. URL: https://ieeexplore.ieee.org/document/4578383. [Online; accessed 23 April 2023]

[62] V. Gomes, G. Queiroz, K. Ferreira. An Overview of Platforms for Big Earth Observation Data Management and Analysis. URL: https://www.mdpi.com/2072-4292/12/8/1253. [Online; accessed 23 April 2023]

[63] Sentinel Hub. Engine for processing of petabytes of satellite data. URL: https://www.sentinel-hub.com. [Online; accessed 23 April 2023]

[64] Kubernetes. An open-source container orchestration system for automating software deployment, scaling, and management URL: https://kubernetes.io. [Online; accessed 23 May 2023]

[65] Istio. An open platform-independent service mesh that provides traffic management, policy enforcement, and telemetry collection. URL: https://istio.io. [Online; accessed 23 May 2023]

[66] NPM package webtorrent, issues. Proposal: live streaming. URL: https://github.com/webtorrent/webtorrent/issues/1993. [Online; accessed 23 May 2023]

Appendices

Figure 2.3 Layer UML model [32]