On data transfer between mobile web clients

Andrei Stephenson, Dmitry Namiot

***Abstract*—In this paper, the authors will conduct a comparative analysis of the following methods of data transfer between mobile web clients: Content Delivery Networks (CDN), Cloud Computing, Peer-to-Peer networking, and WebRTC. By first describing the structure and functionality of each of the main methods of data transfer, and the advantages/disadvantages and social implications associated with them, the author will be able to conclude which method is most effective for on-demand data transfer between individual clients.**

***Keywords*—CDN, cloud services, P2P, WebRTC.**

I. Introduction

Globally, internet use has increased dramatically over the last 15 years. From 2000 to 2009 the number of internet users grew from 394 million to 1.858 billion. By 2014 that number has nearly than doubled to over 3 billion, or 46.3% of the world population. As of January 2014, for the first time in history, internet usage via mobile devices surpassed that of PC usage in the U.S. This rapid growth in internet usage by the general public and the increasing use of mobile devices to access the internet stresses the importance of the development of reliable, convenient, and on-demand methods of data transfer between users.

The fundamental role of the internet is to provide an environment for the transfer of data between users. This data can take a variety of forms such as text files, music, images, video files, banking information, etc. Although numerous methods of data transfer have existed since the time the internet became available for wide-spread public use (such as email and FTP), many of these methods require users to first register an account with an application provider, through which the user would be able to send and receive messages. More recently, many downloadable applications have been created for instant communication between. However, it is required for these clients to have the same application preloaded on their respective devices in order to communicate with other. In addition to registration requirements, many of these applications have limitations as to how much data can be stored or transferred at one time, and they could have charges associated with their usage.

Registration requirements make it difficult for users to communicate in an instantaneous, or on-demand, manner. On-demand communication is especially important in professional fields such as emergency medicine, petroleum engineering, etc., where information must be transferred between specialists working at different locations. For example, a paramedic arrives at the scene of a patient bitten by a venomous snake. The paramedic is able to take a picture of the snake, but cannot properly identify it in order to give the patient the correct anti-venom. How can the paramedic send a picture of the snake to the herpetologist instantly? What will the paramedic do if he either does not know, or does not have the same messaging application preloaded on his phone as the herpetologist?

Currently, a new technology standard called Web Real-time Communication (WebRTC) is being developed by various organizations. WebRTC embeds P2P technology into web browsers, allowing them to create direct links over the internet with the computers of other users, without the need for installing additional software or service applications [1].

There are two networking architectures: the client-server model and Peer-to-Peer model (P2P). In this paper we will describe the methods of data transfer associated with each of these two networking architectures.

- Client-Server-based data transfer:
  - CDN
  - Cloud-based Services

- Peer-to-Peer-based data transfer:
  - P2P Networks
  - WebRTC

We will also conduct a comparative analysis of these data transfer methods and conclude which one is most suitable for users.

II. Client-Server-based data transfer

A. Content Delivery Networks

In CDN's the content provider (CDN origin server) places copies ("replicas") of data in a set of nodes at different geographic locations, and directs the client to use a nearby node as the server. This allows for content to be placed in close proximity to the clients [2]. In contrast to web proxies, which employ reactive caching, CDNs employ solutions based on proactive caching. With proactive caching, content is pre-fetched from the origin server and not cached on demand. In a CDN, multiple replica (CDN servers) host the same content, and requests from browsers for content are transferred to the replica that can most effectively serve the user. In order to choose the best replicas for content delivery, CDNs must utilize dynamic information about network conditions and load on replicas, instead using of

A.Stephenson is master candidate at Lomonosov Moscow State University (email: asteph2406@gmail.com)

D.Namiot is senior scientist at Lomonosov Moscow State University (email: dnamiot@gmail.com)

static information about geographic locations and network connectivity alone. Operating CDNs is complex and expensive, therefore they are typically built and managed by companies that offer content distribution service to several content providers. Examples of a content distribution provider are Akamai, Speedera Inc., Globix Inc., AT&T [3].

The architecture of CDNs consists of surrogate servers (or mirrors [3]), clients, and the origin server. In general, there are three activities that take place on a CDN:

• request-routing infrastructure consists of mechanisms to redirect content requests from a client to a suitable surrogate.

• distribution infrastructure consists of mechanisms to move contents from the origin server to the surrogates.

• accounting infrastructure tracks and collects data on request-routing, distribution, and delivery functions within the CDN creating logs and reports of distribution and delivery activities [3].

The origin server interacts with the CDN in two ways: (1) it pushes new content to the replica servers; (2) it requests logs and other accounting data from the CDN or the CDN itself provides this data to the origin server through the accounting infrastructure. The clients interact with the CDN through the request routing infrastructure and surrogate servers [4].

The proactive caching infrastructure must be transparent to the end-users that must see no difference with being served directly by the central server. Proactive caching to the edges offers better delivery to the client because the content is located in their proximity. This close-to-the-client deployment mode is commonly known as forward proxy caching. Forward proxy implementations can reduce wide area network traffic by 30 to 50 percent [3].

A Web cache monitors Internet traffic, intercepts requests for Web objects and then fulfils those requests from the set of objects it stores (cache hit). If the requested object is not in the cache (cache miss), the cache forwards the request to the origin server, which sends a copy of the object back to the cache. The cache stores the object and sends it back to the requester. Caches in CDN cooperate interacting through the Internet Cache Protocol (ICP). ICP is typically used to build cache clusters or child-parent relationships in hierarchical caching [3].

Caching activity in a CDN may involve different types of contents and therefore different functionalities:

• Static Caching: to cache and replicate static content, such as html pages, images, documents, audio/video file, etc.

• Dynamic Caching: to cache and replicate dynamically generated content. This includes application delivery and replication.

• Streaming Media Caching: to store streaming media objects, as well as to serve streaming media to clients. Essentially, the cache acts as a streaming media server, storing media clips for later use.

• Live Splitting: to cache replicated live streams, so that only one copy is pulled down from the upstream server and is then distributed to the subscribing clients [3].

One of the important problems in CDNs is how to manage the consistency of content at replicas with that at the origin server, especially for those documents changing dynamically. Cached objects typically have associated expiration times after which they are considered stale and must be validated with a remote server (origin or another cache) before they can be sent to a client. A technique to achieve cache consistency consists in pre-populating, or pushing, content to the cache before requests arrive. When automatically pushing a new, or updated, Web object to a cache, the content in the cache is guaranteed to be always fresh and there is no reason for the cache to initiate a freshness check with the side effect that this technique often generates a large amount of traffic [3].

In order to direct clients to the appropriate CDN server, a method called DNS redirection is used. Suppose that a client wants to fetch a page with the URL http://www.cdn.com/page.html. To fetch the page, the browser will use DNS to resolve www.cdn.com to an IP address. This DNS lookup proceeds in the usual manner. By using the DNS protocol, the browser learns the IP address of the name server for the web page cdn.com. The browser then contacts the name server to ask it to resolve www.cdn.com. The name server of the web page is run by the CDN, and instead, of returning the same IP address for each request, it will look at the IP address of the client making the request and return the appropriate answer. The answer will be the IP address of the CDN node (or replica) that is nearest the client, and the client will be able to retrieve the web page from the replica [2].

The benefits of CDNs are: 1) Improvement of user experience due to clients being able to download content from nearby servers instead of distant servers. This reduces round-trip time, and thus decreases page load time. 2) There is a reduction in the total load that is placed on the network. 3) CDNs can be scaled up to as many clients as needed by using more nodes [2].

An example of a company that provides content distribution for content providers is Akamai. Akamai is comprised of more than 61,000 servers located across nearly 1,000 networks in 70 countries worldwide, the Akamai platform delivers hundreds of billions of Internet interactions daily, helping thousands of enterprises boost the performance and reliability of their Internet applications [4].

YouTube is a web-based service that provides video sharing via the Internet. Clients can upload their own videos and make them available to the public. Viewers can search for videos and then watch these videos on their computers or mobile devices. YouTube employs a CDN to deliver content to its end users. When a client chooses a specific video, a HTTP GET message is sent from the client to the YouTube web server. The YouTube server responds with a redirect (HTTP303 See Other) message that contains a response-header field that redirects the client to the CDN server that is most appropriate for the user at that time [5].

B. Cloud computing

Cloud computing is a new and increasingly popular technology. It is conceptually built upon the fusion of older technologies such as grid computing, virtualization, Web 2.0, and service oriented architecture [6]. Advances of multi-core technology, the low cost of system hardware, and the increasing cost of the energy needed to operate them is

motivating an increasing amount of organizations to adopt cloud-based solutions for their operational needs [7]. Although a formal definition of cloud computing remains to be formulated, one of the widely-used definitions has been provided by Ian Foster. According to Foster, cloud computing is "A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet" [6; 7].

In cloud computing, applications can be run inside a user's browser, with user data stored on servers in Internet data centers. The browsers can use web protocols to access data stored on the cloud, and to request computation of data on the cloud. This is advantageous to users because they wouldn't have to waste resources on computation of data, storage of data, nor on the installation of applications. Also user data and applications can be accessed from any computer with an internet connection in any geographical location on demand [2]. Other benefits provided by cloud computing are:

- Scalability- Cloud service providers (CSPs) have large data centers with many servers that are used for data storage and processing [7]. CSPs also implement the methods of resource virtualization and statistical multiplexing to provide seemingly limitless resources to their clients.
- Lower operation costs- Clouds provide affordable solutions that handle peaks, or scale easily at a fraction of the traditional costs of space, time and financial investment [7].
- Manageability- The user experience is simplified as no configuration or backup is needed [7].

C. Cloud Elasticity and Virtualization

Cloud computing implements the technology of virtualization, where applications share the underlying physical hardware by running in isolated Virtual Machines (VMs). The VMs are each delegated a certain amount of computing resources such as CPU, memory and I/O. Resource provisioning plays an important role for achieving scalability in the cloud. Resource provisioning refers to optimally allocating resources to VM's to match their workload. Efficient resource provisioning is mainly achieved by two methods:

- Static resource provisioning- VMs are created with a specific size and then are consolidated onto a set of physical servers. VM capacity is static does not change (i.e. is static). It is usually conducted offline and occurs on monthly or seasonal timescales.
- Dynamic resource provisioning- VM capacity is dynamically adjusted to match workload fluctuations.

In both static and dynamic provisioning, VM sizing is the most important step. VM sizing is the estimation of the amount of resources that should be allocated to a VM. The goal of VM sizing is to ensure that VM capacity is commensurate with the workload. While over-provisioning wastes costly resources, under-provisioning degrades application performance and may lose customers. Dynamic resource provisioning implements statistical multiplexing methods to estimate the capacity needs of multiple VMs. For example, the peaks and troughs of a VM's demand characteristics do not necessarily coincide with those of other VMs. The unused resources of a low utilized VM can then be directed to other co-located VMs at their peak utilization. Thus, VM multiplexing can be used to improve resource utilization in clouds [8].

D. Cloud Types

Clouds can be classified in terms of who owns and manages the cloud. The main types of clouds are public clouds, private clouds, community clouds and hybrid clouds [7]. These cloud types are known as deployment models and they describe the scope of services offered on the cloud to the customers [9].

A public clouds (or external cloud) is the most common form of cloud computing in which services are made available to the general public over the internet. Users access services offered by the cloud from a cloud service provider (CSP) on a pay-per-use basis. Companies such as Amazon EC2, Google App Engine, and Salesforce.com have dedicated numerous large data centers to the managing of users' data. This allows users to freely scale and shrink their rented resources with low cost and little management burden. Security, privacy and data governance are major concerns of this type of cloud [7,9].

In a private cloud (or internal cloud) the computing resources are operated exclusively by or for one organization, and serves users within the business fire-wall [7, 9]. They may be managed by the organization itself or by a CSP. The owner of the private cloud is the organization. Because private clouds can be accessed exclusively by trusted users inside the organization, they are considered more secure than public clouds. Scalability for private clouds is limited to the installed infrastructure dedicated to the organization. The limited amount of resources available to private cloud also leads to higher maintenance costs (i.e. for space, cooling, energy consumption and hardware) [Cloud types- Jin]. The other two deployment models, community and hybrid clouds, fall between public and private clouds [9].

The concept of Community Clouds is derived from the Grid computing and Volunteer Computing paradigms [7]. In a community cloud, several organizations that have the same mission, policy and security requirements share cloud infrastructure and computing resources [9]. This allows the organizations to increase their scale while sharing the cost [7]. Universities and institutes around the world use educational clouds to provide education and research services [9]

A hybrid cloud is a combination of a private and a public cloud. In this deployment model a private cloud is able to maintain high-level service availability by scaling up their system with externally provisioned resources from a public cloud whenever there are workload fluctuations or hardware failures. Hybrid clouds also allow an enterprise to keep their critical data and applications amongst trusted users within their firewall, while hosting less critical data on a public

cloud [7].

E. Cloud Service Models

Cloud service models are a Service-Oriented-Architecture (SOA) that describe cloud services at different levels of abstraction. There are three cloud service models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a service (IaaS).

SaaS is based on licensing software use on demand, which is already installed and running on a cloud platform. These complete software applications are delivered over the Internet and are accessed via a web browser. Cloud hosted software, such as Google Apps, are available on demand on any client devices that has access to the Internet. Google Apps has several components: Communication components (Gmail, Google Talk); Office components (Google Spreadsheets, Google Docs); Google calendar allows multiple users to organize events and meetings; Google Web Pages allows users to easily publish web pages [7]. Unlike traditional software, SaaS has the advantage that the customer does not need to buy licenses, install, upgrade, maintain or run software on his own computer [9]. It has also other advantages such as multitenant efficiency, configurability and scalability [9]. Another implementation of SaaS is Desktop as a Service. It provides a virtualized desktop-like personal workspace, and sends its image to the user's real desktop. This service allows users to access their own cloud-based desktop from any device. The "Global Hosted Operating SysTem" (G.ho.st) is a free and complete Internet-based Virtual Computer (VC) service suite including a personal desktop, files and applications. Other examples the of SaaS model are: Salesforce, A2Zapps.com, Envysion.com, OpenId, Zoho, etc. [7].

Platform as a Service (PaaS) cloud systems provide a software execution environment that application services can run on. The environment is not just a pre-installed operating system but is also integrated with a programming-language-level platform, which can be used to develop and build applications for the platform. From the point of view of PaaS clouds' users, computing resources are encapsulated into independent containers, they can develop their own applications with certain program languages, and APIs are supported by the container without having to take care of the resource management or allocation problems such as automatic scaling and load balancing. In this section we introduce two typical PaaS : Google App Engine (2010), and Microsoft Azure (2010).

Infrastructure as a Service (IaaS) is one of the "Everything as a Service" trends. IaaS is easier to understand if we refer it as Hardware as a Service (i.e. instead of constructing our own server farms, a small firm could consider paying to use infrastructure provided by professional enterprises). Companies such as Google, Microsoft and IBM are involved in offering such services. Large-scale computer hardware and high computer network connectivity are essential components of an effective IaaS [7].

The IaaS is categorized into: (1) Computation as a Service (CaaS), in which virtual machine based servers are rented and charged per hour based on the virtual machine capacity – mainly CPU and RAM size, features of the virtual machine, OS and deployed software; and (2) Data as a Service (DaaS), in which unlimited storage space is used to store the user's data regardless of its type, charged per GByte for data size and data transfer. In this section we will describe some popular IaaS systems such as Amazon EC2 (2010), GoGrid(2010), Amazon S3 [7].

III. Peer-to-Peer-based data transfer

A. Peer-to-Peer Networks

P2P networking became well-known in the late 90's early 2000's with the popularization of the now infamous web service Napster. At its peak, there were over 50 million users on Napster. Because Napster allowed users to exchange copyrighted data (mainly music) without the copyright owner's permission, Napster was shut down by the courts. Nevertheless, other P2P technologies were developed and P2P traffic quickly exceeded web traffic. At present, BitTorrent is the most popular P2P protocol. It is used do widely to share videos, as well as other content, that it accounts for a large fraction of all internet traffic.

According to Kellerer [10], a distributed network architecture may be called a P2P network if the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity, printers, etc). These resources are accessible by other nodes directly without passing intermediary entities. The nodes are called "peers" because all of the participants of the P2P network can serve as both resource providers and resource requestors [10]. In a P2P file-sharing network many peers come together and pool their resources to form a content distribution system [2].

B. Computer Systems Architecture

Computer systems architecture is divided into centralized systems and distributed systems. Centralized systems represent single, non-networked units which include single and multi-processor machines, and high-end machines, such as supercomputers and mainframes. Distributed systems are those in which components located at networked computers communicate and coordinate their actions only by passing messages [11].

Distributed systems are in turn divided into the client-server model and the P2P model. An example of a client-server model is a CDN, where clients retrieve their resources from a central server. The client server model can be flat or hierarchical. In the flat model all clients only communicate with a single server. In the hierarchical client server model, each server has its own set of clients, and those servers, in-turn, are also clients of higher level servers. An example of the hierarchical model is DNS [11].

The P2P model can be either pure or hybrid. In the pure model there is no centralized server within the network. Examples of pure P2P networks are Gnutella and Freenet [11]. Because there are no servers on which the network can rely on, peer have to organize themselves into a coherent architecture to accomplish the task of content delivery [2].
In a hybrid P2P model, a central sever is used to provide meta-information to peers about the identity/address of other

peers in the network, with whom they want to establish a direct P2P communication. There is also an intermediary P2P model that sonsists of two tiers of peers: ordinary peers and SuperPeers (or SuperNodes) [11]. The SuperPeers are ordinary machines that have high speed internet connection. Both ordinary machines and SuperPeers store IP addresses, but SuperPeers have a larger amount of addresses stored on their systems . Topology-wise, this intermediary P2P model is also classified as hybrid. Examples of this type of P2P model are FastTrack, Kazaa and Skype. Other examples of the hybrid P2P model are Instant messaging applications such as Whatsapp, Viber, etc.

C. *Characteristics of P2P systems*

According to the Intel P2P working group, P2P is "The sharing of computer resources and services by direct exchange between systems" [12]. This gives P2P systems several characteristics:

1. Scalability: there is no algorithmic, or technical limitation of the size of the system, e.g. the complexity of the system should be somewhat constant regardless of number of nodes in the system [12]. In centralized systems, such as client server, scalability is limited by the capacity of the central server itself and by the amount of centralized operations it needs to perform[11].

2. Reliability: The malfunction on any given node will not effect the whole system (or maybe even any other nodes).

3. Anonimity: there are six popular techniques employed by P2P networks in order to insure anonymity of users: Multicasting; Spoofing address of sender; Identity spoofing; Covert paths; Intractable aliases; Non-voluntary placement.

4. Shared Cost of Ownership: Shared ownership reduces the cost of owning the systems and the content, and the cost of maintaining them [11]

5. Decentralization: In traditional centralized client-server models (CDNs or Clouds) information is stored on and distributed from centrally located servers. Although a central topology is useful for certain applications and tasks (granting access rights and security), the whole system is vulnerable to interruption of service and bottlenecks if the central server is corrupted. One of the more powerful ideas of decentralization is the emphasis on the users' ownership and control of data and resources. In a fully decentralized system, every peer is an equal participant. This makes the implementation of the P2P models difficult in practice because there is no centralized server with a global view of all the peers in the network or the files they provide. This is the reason why many P2P file systems are built as hybrid approaches as in the case of Napster, where there is a centralized directory of the files but the nodes download files directly from their peers

IV. WEBRTC

WebRTC is a newly developing standard technology that will allow web browsers to capture video and audion streams without the use of additional plugins and to communicate with other WebRTC-enabled web browsers in a P2P fashion [13].

Traditional web browsers only support the viewing of rich text and images. Third-party browser plugins such as adobe Flash and Microsoft Silverlight have to be installed in order for the browsers to support video and audio [13]. In October 2014, the hypertext markup language specification, HTML5, was given recommendation status (i.e. finalized) by the W3C. HTML5 adds support for playing video and audio without the presence of third-party browser plugins [13].

Classic web browsers are only able to view content located on remote servers. They lack the capability to capture audio or video input from a microphone or camera for the purpose of transferring that data to other clients or servers. Although there are third-party plugins that provide this functionality, these plugins are prone to security flaws [13]. Plugins may also lead to compatibility and performance issues, or they may be unavailable on certain devices or platforms [13]. Additionally, using classic browsers to transfer data between clients requires the presence of a server that relays the data from one client to the other [13].

In May 2010, Google purchased the Oslo-based company Global IP Solutions (GIPS) for $68 million in order to develop its real-time audio and Internet capabilities. GIPS developed embedded solutions for transmitting audio and video data in real time. By using the GIPS technology, Google was able to develop WebRTC. Mozilla also contributes to WebRTC's development. There are two groups currently involved in the standardization of WebRTC: W3C, which is standardizing the API's, and IETF, which is working on protocols and P2P connections capability (Webtorial). As of the time of writing, recent versions of Chrome, Firefox, and Opera support WebRTC in a browser-specific manner. Because the standards for WebRTC have not been fully formalized, browsers differ in their implementation. Thus WebRTC applications require browser-specific code in order to be run in the three browsers. Google created a Javascript library that can be used to deal with issued of interoperability between browsers [13].

A. *WebRTC Architecture*

WebRTC applications are developed on top of a layered architecture, where the WebAPI sits on top of the native implementation of WebRTC specific to the browser. WebAPI is the interface used by web developers when writing applications that utilize WebRTC. Once the W3C standard is formalized, there will be one uniform WebAPI that will be used in all browsers and environments that support WebRTC. Although the native implementation for WebRTC may vary amongst the different browsers, they all should contain the following functional components:

1. Transport Component. In order to establish P2P connection between clients, the browser needs to implement various protocols such as: ICE (Interactive Connectivity Establishment), STUN (Session Traversal Utilities for NAT) and TURN (Traversal Using Relays around NAT).

2. Video Component. Provides encoding and decoding procedures by using various codecs. The video component should also allow for smooth video streams by employing solutions such as jitter buffers. Browsers should be able to stream video data from web cams or other video sources.

3. Audio Component- The audio component should provide codecs for collecting and transferring audio data, as well as providing additional features such as noise reduction, echo cancellation, and audio streaming [13].

B. Signaling and NAT Traversal

One of the primary functions of WebRTC is to establish P2P connections between clients without the involvement of web servers. However servers are still necessary for establishing P2P connections for two reasons:

1) To store the web application that utilizes WebRTC;
2) To initialize sessions between clients.

Thus, WebRTC utilizes a hybrid P2P topology for data transfer. The Session Description protocol is used to exchange session descriptions which contain details on the form and nature of the data to be transmitted. This process is referred to as Signaling. In WebRTC signaling is can be performed by any technology, provided that session descriptions are exchanged. For example, a WebRTC For example, a WebRTC application that involves VoIP softphones may want to use SIP for signaling. A chat application might choose to use XMPP and plain web applications can use HTTPS . Once the SDP session descriptions have been exchanged, the browsers of the clients can establish a P2P connection for data transfer. Clients may be located behind NATs or firewalls within a private network. NATs replace the private source and destination IP addresses with public ones while maintaining an interal lookup table of the mapping to the original device. Because of this, devices behind the NAT can communicate without being aware that the other nodes, with whom it sharing data, may be located in different networks. In order to understand how NAT traversal works in WebRTC it is necessary to describe the different types of NAT and understand the following protocols involved in NAT traversal: STUN, TURN, and hole punching. These protocols are combined to form the ICE protocol [13].

C. Classification of NAT

In order to describe the classification of NAT we will employ a hypothetical scenario, where IP addresses and ports will be represented by variables:

1. A client computer *A* behind NAT with a private IP address a
2. A router *R* with a public IP address r
3. Two STUN servers *S1* and *S2* with public IP addresses *s1* and *s2*
4. A public server P with a public IP address *p*

Packets are modified whenever they pass through a router by NAT traversal. If *a:321* sends a packet to *p:90*, *R* will modify the packet so that it appears to P as if the traffic was coming from *r:x* and not *a:321* where *x* is some port number chosen by *R*. This involves replacing all occurrences of *a:321* with *r:x* and remembering the mapping of private IP address and port to public IP address and port in a NAT forwarding table. Incoming packets from the server *P* to the destination *r:x* will be edited so that the destination becomes *a:321*. To both the client *A* and the server *P*, it seems like they are communicating directly, even though *R* performs network address translation. The following NAT types are as listed [13]:

1. Full-cone NAT- is the least restrictive NAT. *R* creates a NAT forwarding table entry the first time *A* sends a packet to P. For example, the mapping may be (a*:321 -> r:x*). Once this mapping has been established, any server Q can send data from *q:z* to *r:x* and *R* will forward the packets to *a:321*, even if *A* never contacted *Q*.

2. Address-restricted-cone NAT- imposes a restriction on full-cone NAT, in that each port forwarding table entry is only valid for a specific target IP. For example, a mapping may be (*a:321 -> r:x* to *p*) and only *P* can send packets to *r:x*. Packets sent from other IP addresses are dropped by *R*. The port is still unrestricted, meaning that P can send data to *r:x* from any port and the data will be forwarded to *a:321*.

3. Port-restricted-cone NAT- imposes the additional restriction that only packets coming from the exact *IP:port* pair contacted by *A* will be forwarded. This means *R* may create a mapping (*a:321 -> r:x* to *p:90*) when A contacted P on p:90, *P* can only send data from *p:90* to *r:x* for it to be forwarded to *A*. If the server *P* sends data from any other port, then *R* drops the packets.

4. Symmetric NAT- is the most restrictive NAT. Same as with any other type of NAT, when A contacts P from *a:321*, *R* chooses a port *x* at random and creates a NAT forwarding table entry, for example (*a:321 -> r:x* to *p:90*). However, if *A* establishes another connection to *P*, even if using the same private port, i.e. sending data from *a:321*, *R* will choose a second random public port y, creating a second mapping (*a:321 -> r:y* to *p:90*). The public server *P* can reply to the packets coming from *A* via *R*, since it knows which port has been used (*R* replaced all occurrences of *a:1234* with *r:x* or *r:y* inside the packet), but *P* only knows this if it received the packets coming from *R*. If *P* has not received any packets from *R, P* does not know the port which *R* has mapped for traffic to pass through to *A*. This condition has important implications for NAT traversal.

D. STUN

STUN is a protocol which can assist other applications in performing NAT traversal. It does not perform NAT traversal by itself. Clients located behind NAT are unaware of their own public IP address and port, which are handled by the router. These clients only know their own private IP address and port. A STUN server can be used by clients behind NAT to determine their public IP address and port. STUN servers are placed on the public web and clients behind NAT can send messages to the STUN server. The STUN server will then reply with a message containing the client's public IP address and port as seen from the STUN

server. The client may even, to a certain degree, be able to determine the type of NAT employed by the router by sending requests to multiple distinct STUN servers. When sending requests to different STUN servers using the same private port, the answers of the STUN servers will reflect the nature of the NAT employed by the router: For example, assuming a client *A* has a private IP address and port *a:1* and uses this port to contact STUN servers *S1* and *S2*, it may occur that both *S1* and *S2* reply to the request saying that the public IP address and port are *r:9999*. However, it may also happen that the two STUN servers give different replies, for example *r:9999* and *r:8888*. The latter case would indicate that symmetric NAT is used by the router, since each outgoing connection has been assigned another port, even though A used the same private port to contact both STUN servers [13].

E. TURN

Another protocol that may be used in NAT traversal is TURN. TURN employs a publicly known server for clients behind NAT to connect to. In essence, two clients A and B both connect to the TURN server in a classical client-server topology and the TURN server then relays data between A and B. As such, if TURN is used, the traffic between A and B is no longer peer-to-peer. TURN can be used as a final fall-back if all other NAT traversal techniques fail, since normal client-server connections are possible in all NAT scenarios. However, relaying the data incurs additional cost. While STUN servers require very little resources to operate, TURN servers have high traffic demands, since all traffic between the peers is relayed through the server [13].

F. Hole Punching

Hole punching is an umbrella term for a variety of techniques that can be used to perform NAT traversal between peers. The general mechanism by which it functions can be described as follows [10]: If *A* and *B* are clients behind NAT routers *R1* and *R2*, A and B first contact STUN servers to discover their public *IP:port* pairs. They subsequently exchange the pairs via a signaling server and then use the same private ports to contact each other. When *A* attempts to contact *B*, *R2* will drop any incoming packet, because *R2's* NAT forwarding table does not contain any entries that would allow *A* to contact *B*. However, in sending a packet to *B*, *A* has caused *R1* to create a new mapping, for example (*a:321-> r1:9999* to *R2:9999*), even though the packet was eventually dropped when it reached *R2*. Now, B can send a packet to r1:9999, causing R2 to create a mapping, for example (*b:5678 -> r2:1111* to *R1*) as well and *R1* will forward the packet to *a:321*, because this mapping exists in *R1*'s NAT forwarding table. Now, both routers have created appropriate NAT forwarding table entries and peer-to-peer communication can take place. This mechanism usually works for all types of NAT except when both parties are behind symmetric NAT. When replaying the scenario above with both *A* and *B* behind symmetric NAT, *A* would send a packet to *B*, causing *R1* to create a mapping like (*a:321 -> r1:9999* to *R2:9999*) and *R2* would drop the packet as usual. However, B will still not be able to send a packet to *A* because it cannot know which random port *R1* has used to create the mapping, since it is almost certainly a different port other than the one reported by the STUN server in the first place. In principle, it is possible to traverse certain symmetric NAT configurations by contacting multiple echo servers many times to collect a large number of public *IP:port* candidate pairs. Given enough samples, it may be possible to predict the next port chosen by the router. However, there is no indication that this technique is used by current WebRTC capable browsers [1].

G. ICE

WebRTC uses ICE, a protocol which utilizes STUN, TURN and hole punching, to establish a connection between peers. The procedure can be broken down into the following steps [1]:

1. Gathering candidates: As a first step, each client constructs a list of possible IP address and port pairs for other to connect to. These pairs are called 'candidates'. The first and obvious candidate is the local, private IP address and port. More pairs can be gathered by contacting one or more STUN servers, who will reply with public IP address and port pairs. Another candidate can be obtained by contacting a TURN server. At the end of the gathering process, the candidate list will contain several pairs.
2. Distributing candidates via the signaling server: The clients send their candidate lists to the signaling server which distributes them to the other clients. Now, all clients know the candidates of the other clients.
3. Connection and NAT traversal: Each client attempts to connect to other clients using the first candidate in the list. In case that a target client is not behind NAT, the first candidate - typically the IP address and port of the client machine itself, may already be sufficient. If the target client is behind NAT, the second candidate (returned by a STUN server and representing the public IP address and port of the router) may be the correct match. In case that the target client is behind a restricted NAT, hole punching is employed by the peers to establish a connection. Should this also fail, because all clients are behind symmetric NAT, the final candidate (representing the IP address and port of the publicly known TURN server) has to be used, and all traffic will be relayed via the TURN server.

By combining STUN, TURN and hole punching, ICE is able to penetrate most NAT types, including those where all peers are behind symmetric NAT. If at least one client is not behind symmetric NAT, TURN is usually not necessary because hole punching can be employed in the direction of the client behind the non-symmetric NAT. In other words, the client behind non-symmetric NAT can send a packet to any client behind symmetric NAT, thereby creating a new entry in the NAT forwarding table, after which the client behind symmetric NAT is able to contact the client behind non symmetric NAT. TURN is only necessary when all participants are behind symmetric NAT [1].

H. Security and Privacy

All traffic in WebRTC is encrypted using AES in SRTP for media and DLTS for data transmission. All security mechanisms are provided by WebRTC in the browser by default. This is beneficial because it would prevent mistakes being made by developers when encrypting data [1].

V. Comparative Analysis of the Technologies of Data Transfer

A. Client-Server-based data transfer

The differences between cloud computing and CDNs may seem unclear as they both can offer remote data storage, computing and networking capabilities to clients. However, there are more important differences between the core functions and capabilities of the two technologies. One of the main goals of CDNs is to provide end-user proximity to content by replicating content from the origin to the edge caches geographically located closer to the end-users (clients). End-user proximity is not a priority for cloud services. The main goal of cloud services is to offer online computing and storage capabilities to clients irrespective of their geographical location. Cloud servers usually consist of distributed computing systems located within one or a few data centers.

Because Cloud computing and CDNs are both based on the client-server networking architecture, they share common advantages and disadvantages associated with it [14].

Advantages of both technologies:

1. Data management is much easier because the files are in one location. This allows fast backups and efficient error management. There are multiple levels of permissions, which can prevent users from doing damage to files.
2. The server hardware is designed to serve requests from clients quickly. All the data are processed on the server, and only the results are returned to the client. This reduces the amount of network traffic between the server and the client machine, improving network performance.
3. Thin client architectures allow a quick replacement of defect clients, because all data and applications are on the server.

Disadvantages of both technologies:

1. Client-Server-Systems are very expensive and need a lot of maintenance.
2. The server constitutes a single point of failure. If failures on the server occur, it is possible that the system suffers heavy delay or completely breaks down, which can potentially block hundreds of clients from working with their data or their applications. Within companies high costs could accumulate due to server downtime.
3. Content stored on third-party servers can be easily monitored a third-party

CDNs provide end-user proximity to content by storing replicas of digital content at various locations near the edges of the internet [15]. By being located closer to clients, CDNs have the advantage over cloud services of providing them with faster PLT's. CDNs also reduces the system failure risk by providing redirection to many replica servers that contain the same content. CDNs can provide load balancing between servers in order to avoid network bottlenecks and, therefore, ensuring greater performance and QoE (Quality of Experience) to the end user [16].

However, there are two main disadvantages of CDNs that prevent it from being a more widely-used service for data transfer:

1) expensive construction cost and
2) administration complexity.

The construction of multiple data centers located in various geographical locations is a very expensive endeavor. This is why CDNs have been utilized mainly by large corporations and not at an individual user level. Cloud services usually provide computing resources located within one or a few data centers. Thus the cost of utilizing cloud services is less expensive making them are more appealing to individual clients or small businesses than CDNs[15].

Many of the advantages and disadvantages associated with clouds relate to both individual clients and business (or small groups) in various ways. For businesses, cloud computing offers a cost-efficient alternative to periodically purchasing licensing fees for desktop software. Most cloud service companies offer on-time payment or pay-as-you-go options. For individual clients who usually don't prefer to pay for online services, paying for cloud services might be viewed as a disadvantage of the technology[14]. Some of the other advantages for both businesses and individuals are [14]:

- Almost unlimited storage
- Backup and Recovery
- Automatic software integration
- Easier scalability services for businesses
- Quick deployment of Cloud services throughout the organization

The cloud also offers easy access to information for registered users. This can be an advantage for clients belonging to the same organization, who need to transfer data between each other. However, it may disadvantageous to clients who want to exchange data with other clients not registered to the same cloud [14].

In spite of the many benefits of cloud computing, there are also disadvantages associated with it. Technical limitations of the cloud itself can affect businesses and individual clients negatively. Cloud computing makes the function of the business dependent on its connection to the internet. Loss of internet connection can lead to possible downtimes. Another disadvantage is inflexibility, where an organization or individual would be locked into the proprietary applications or formats of the cloud they use. Security issues of the cloud relate to the reality that an organization or individual will be surrendering all of their sensitive information to a third-party cloud service provider. In addition, storing information on a cloud could make companies prone to attacks [14].

B. Peer-to-Peer-based data transfer

Both P2P and WebRTC are examples of the P2P networking architecture. Thus, they share some fo the same advantages and disadvantages of this networking structure [17].

Advantages of both technologies:

1. In a pure Peer-to-Peer architecture there is no single point of failure, that means, if one peer breaks down, the rest of the peers are still able to communicate.
2. Peer-to-Peer provides the opportunity to take advantage of unused resources such as processing power for computations and storage capacity. In Client-Server architectures, the centralized system bears the majority of the cost of the system. In Peer-to-Peer, all peers help spread the cost; each client not only gets service but also provides service as well.
3. Peer-to-Peer allows to prevent bottleneck such as traffic overload using a central server architecture, because Peer-to-Peer can distribute data and balance request across the internet without using a central server.
4. There is better scalability due to a lack of centralized control and because most peers interact with each other.
5. Privacy. Content cannot be easily monitored by third-parties

Disadvantages of both technologies [17]:

1. Today many applications need a high security standard, which is not satisfied by current Peer-to-Peer solutions.
2. The connections between the peers are normally not designed for high throughput rates, even if the coverage of ADSL and Cable modem connections is increasing.
3. A centralized system or a Client-Server system will work as long as the service provider keeps it up and running. If peers start to abandon a Peer-to-Peer system, services will not be available to anyone.
4. P2P networks are not easy to manage and the quality of service is also a problem. Peers usually perform selfishly, and will ignore the global benefit
5. Most search engines work best when they can search a central database rather than launch a meta-search of peers [14].

This problem is circumvented by the hybrid Peer-to-Peer architecture.

Although P2P applications and WebRTC share the same networking capabilities, there is one key difference between these two technologies. P2P applications require clients to download additional software and be registered in specific communities in order to share files with other peers. These requirements makes it prevents communication to be established between clients subscribed to different data transfer applications in an on-demand (or instantaneous manner). In the case of hybrid P2P apps (such as Whatsapp or Viber), even if the respective clients share the same application, they would only be allowed to exchange data once they are add in each other's friend's list. WebRTC natively embeds P2Pfunctionality into the browser of any mobile or desktop device. Data exchange between peers using WebRTC can occur instantaneously without the need for registration or exchanging of private contact data between peers.

VI. COMPARISON OF DATA TRANSFER TECHNOLOGIES

In order to determine which of the four data transfer technologies is most effective for on-demand data transfer, a comparative analysis of the technologies will be conducted using the following criteria:

1. Service capability and scalability of the service
2. Reliability and stability of the service
3. Accessibility- Is a registration process required in order to use the service?
4. Does additional software need to downloaded in order to use the service?
5. Cost
6. Content Source Monitor (Privacy)- Can information be visible to third-parties
7. Is quality of service (QoS) guaranteed?

Table 1 illustrates this.

	CDN	Cloud Computing	P2P	WebRTC
Service Capability and Scalability	Service capability is limited and expansion cost is higher	Service capability is limited and expansion cost is higher	Service capability can grow up with peer node increases and expansion cost is lower	Service capability can grow up with peer node increases and expansion cost is lower
Reliability and Stability	High reliability, Good Stability. Can redirect service to many replica servers	Relative reliability and stability depends on the functionality. of the Cloud Service	Low reliability, dynamic, and poor stability dependent on functionality and selflessness of peer	Relative reliability and stability dependent on functionality alone (access to internet)

Accessibility (Registration Required?)	Yes	Yes	Yes	No
Additional Software required?	No	No	Yes	No
Cost	Very Expensive	Reasonably Priced for organizations and individuals	Free	Free
Content Source Monitor (Privacy)	Can be monitored	Can be monitored	Difficult to Monitor	Difficult to monitor
QoS Guarantee	Can be guaranteed within the maximum service capacity. Dependent on func. of multiple replica servers	Can be guaranteed within the maximum service capacity. Dependent on func. of the localized cloud server	Best-effort, can't be controlled. Dependent on functionality,& Selflessness of random peers, & internet connection	Best-effort, can't be controlled Dependent on functionality alone (internet connection)

From Table 1 we see that service and scalability are most optimal with P2P networking and WebRTC technologies. This is due to the decentralized nature of the P2P-based networking architecture of these services. In terms of reliability and stability, CDNs are the most reliable and stable because it provides content from multiple, readily available servers. WebRTC is reliable/stable as long as the communicating parties have access to the internet. Cloud services are reliable/stable as long as the data center where the information is stored is functional. P2P networking has the lowest reliability/stability because they are dependent on the selflessness of the peers in the network. WebRTC is the most accessible of the four technologies compared because it does not require registration in order to be utilized. Both WebRTC and P2P are free to use, in contrast to CDNs and Cloud computing. CDN's are much more expensive than Clouds, thus they are usually only employed by large corporations. P2P and WebRTC ensure the most privacy because information is transferred directly between the clients and not through a third-party server. Quality of service (QoS) for CDNs and Clouds is guaranteed within the maximum service capacity, while for P2P and WebRTC QoS is guaranteed by the best effort of the peers in the network.

From this comparative analysis of the four data transfer technologies, we can conclude that WebRTC would be the most effective technology for on-demand data transfer between individual clients. This is because WebRTC is the most accessible technology of the four discussed, it is free to use, it provides the most privacy, and it does not require the downloading of additional software for utilization.

VII. Case Study

A paramedic arrives at the scene of a patient bitten by a venomous snake. The paramedic is able to take a picture of the snake, but cannot properly identify it in order to give the patient the correct anti-venom. How can the paramedic send a picture of the snake to the herpetologist instantly? What will the paramedic do if he either does not know, or does not have the same messaging application preloaded on his phone as the herpetologist? How can the paramedic provide the utmost patient confidentiality while transferring patient information over the internet?

• Solution: The paramedic should use WebRTC to transfer the picture to the herpetologist. WebRTC is embedded in the web browsers of both the paramedic's and herpetologist's device. They do not have to have the same messaging application installed in order to transfer data with each other. They both are not required to login to any accounts, so they can utilize the sevice instantaneously. WebRTC provides direct P2P data transfer, thus patient information cannot be monitored by third-parties.

VIII. Conclusion

In this paper the author conducted a comparative analysis of the existing and newly emerging methods of data transfer to determine which method would be most effective for on-demand data transfer between individual mobile clients. The author proposed several formal criteria for the comparison of the following methods of data transfer between mobile web clients were analyzed: CDN, Cloud Computing, P2P networks, and WebRTC. After describing the functionality and structure of each method the author was able to conduct a comparative analysis of these methods

based on the criteria listed in Table 1. Based on this analysis the author concludes that WebRTC is the most effective method to transfer data on-demand between individual clients.

The main results of this work have been presented as a qualification thesis (post-graduate in Open Information Technologies Lab [18]).

REFERENCES

[1] Werner M. J., Vogt C., Schmidt T. C. Let Our Browsers Socialize: Building User-centric Content Communities on WebRTC.

[2] Tanenbaum, A.S.: Computer Networks, Prentice Hall, Upper Saddle River, NJ. 2010.

[3] Bartolini, Novella, Emiliano Casalicchio, and Salvatore Tucci. "A walk through content delivery networks." Performance Tools and Applications to Networked Systems. Springer Berlin Heidelberg, 2004. 1-25.

[4] Nygren, Erik, Ramesh K. Sitaraman, and Jennifer Sun. "The akamai network: a platform for high-performance internet applications." ACM SIGOPS Operating Systems Review 44.3 (2010): 2-19

[5] Zink, Michael, et al. "Characteristics of YouTube network traffic at a campus network–measurements, models, and implications." Computer Networks 53.4 (2009): 501-514.

[6] Foster, Ian, et al. "Cloud computing and grid computing 360-degree compared."Grid Computing Environments Workshop, 2008. GCE'08. Ieee, 2008.

[7] Jin, Hai, et al. "Cloud types and services." Handbook of Cloud Computing. Springer US, 2010, pp. 335-355.

[8] Meng, Xiaoqiao, et al. "Efficient resource provisioning in compute clouds via vm multiplexing." Proceedings of the 7th international conference on Autonomic computing. ACM, 2010.

[9] Youssef, Ahmed E. "Exploring Cloud Computing Services and Applications."Journal of Emerging Trends in Computing and Information Sciences 3.6 (2012): 838-847.

[10] Schollmeier, Rüdiger. "A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications." Peer-to-Peer Computing, IEEE International Conference on. IEEE Computer Society, 2001.

[11] Milojicic, Dejan S., et al. "Peer-to-peer computing." (2002).

[12] Ding, Choon Hoong, Sarana Nutanong, and Rajkumar Buyya. "Peer-to-peer networks for content sharing." Peer-to-Peer Computing: the Evolution of a Disruptive Technology (2005): 28-65.

[13] Alexandru, Carol. "Impact of WebRTC (P2P in the Browser)." Internet Economics VIII (2014): 39.

[14] APOSTU, ANCA, et al. "Study on advantages and disadvantages of Cloud Computing–the advantages of Telemetry Applications in the Cloud."

[15] Lu, Zhihui, Ye Wang, and Yang Richard Yang. "An analysis and comparison of CDN-P2P-hybrid content delivery system and model." Journal of Communications 7.3 (2012): 232-245.

[16] Oliveira, Thiago, and Marcial Fernandez. "Fuzzy Redirection Algorithm for Content Delivery Network (CDN)." ICN 2013, The Twelfth International Conference on Networks. 2013.

[17] Maly, Robin Jan, et al. "Comparison of Centralized (Client-Server) and Decentralized (Peer-to-Peer) Networking." Semester thesis, ETH Zurich, Zurich, Switzerland (2003): 1-12.

[18] Gur'ev, D. E., D. E. Namiot, and M. A. Shneps. "O telekommunikacionnyh servisah." International Journal of Open Information Technologies 2.4 (2014): 13-17.