**CONTENT DELIVERY NETWORK**

**Submitted by:**

Hongyi Guo,

Dimple Modugula,

Wentao Lu,

Scott Fontenarosa

**CONTENTS PAGE NO**

1.Introduction.........................................................................................3

2.Motivation for Content Delivery Network...........................................3

3.Types of Content and Services in a CDN............................................4

4.Content Delivery Network Architecture..............................................7

5.How CDN Works................................................................................10

6.Content Service Protocols...................................................................11

7.Content Delivery Service Providers.....................................................12

8.Video Stream Coding Part……………………………………………13

9.References............................................................................................24

**1.INTRODUCTION**

In today's world in order to cope with the increasing demands on performance, security and reliability, while at the same time reducing costs, the content delivery networks solution has been developed.

Content delivery networks provide:

1. an efficient way of distributing content to decentralised systems, guaranteeing consistency of information in a timely matter
2. an efficient way of directing requestors of information to the information source, which is the closest and most cost- effective for them to retrieve the information from and
3. a more efficient way to distribute content with the end goal of improving the user experience while accessing the information

Content Delivery Networks (CDN), evolved first in 1998, replicate contents over several mirrored web servers (i.e., surrogate servers) strategically placed at various locations in order to deal with the flash crowds. Geographically distributing the web servers’ facilities is a method commonly used by service providers to improve performance and scalability.

A content delivery network or content distribution network (CDN) is a large distributed system of proxy [servers](https://en.wikipedia.org/wiki/Server_(computing)) deployed in multiple [data centers](https://en.wikipedia.org/wiki/Data_center) via the [Internet](https://en.wikipedia.org/wiki/Internet). The goal of a CDN is to serve content to end-users with high availability and high performance. CDNs serve a large fraction of the Internet content today, including web objects such as text, graphics and scripts, downloadable objects-media files, software, documents, applications-e-commerce, portals, [live streaming](https://en.wikipedia.org/wiki/Live_streaming) media, on-demand streaming media, and [social networks](https://en.wikipedia.org/wiki/Social_network)

**2.MOTIVATION FOR CONTENT DELIVERY NETWORK**

While content delivery networks also solve ancillary problems such as improving global availability and reducing bandwidth, the main problem they address is latency: the amount of time it takes for the host server to receive, process, and deliver on a request for a page resource (images, CSS files, etc.). Latency depends largely on how far away the user is from the server, and it’s compounded by the number of resources a web page contains.

CDNs facilitate faster page loads and offer other important benefits including the following advantages:

***1.Eliminate Pauses and Accommodate Heavy Traffic***

Video streaming often results in jitters and pauses due to lags in transmission times, but CDNs help to deliver better user experiences when downloading video and audio content.

***2.Minimize Packet Loss***

Users get improved streaming quality.

***3.Faster Loading***

Internet speeds improve, but people expect nearly instantaneous page loads. Faster load times increase sales according to many studies and a vast body of anecdotal evidence.

***4.File Mirroring***

File mirroring protects data if natural disasters affect certain areas of the Internet. Hurricanes and earthquakes could cause significant disruptions to services in key geographical area.

***5.Optimize Live Delivery***

The success of YouTube and social media sharing has made video extremely popular, and most businesses should consider adding video elements to their content.

Live events help to generate increased traffic.

Small organizations and schools can broadcast live events to strengthen SEO efforts.

Anyone can broadcast live networks of material as part of a marketing strategy.

***6.Enable Linear Networks***

Companies can broadcast in the same way that major networks do, creating their own program schedules for 24-hour CDN deliveries around the world.

***7.Support Video on Demand***

Organizations can enhance their online presence with video libraries, how-to videos, training programs and other marketing tools.

***8.Scalability***

New technology and advanced mobile applications place increasingly greater demands on servers, but large CDNs can handle new material as companies expand their online presences.

**3.TYPES OF CONTENT AND SERVICES IN A CDN**

Types of Content and Services in a CDN CDN providers host third party contents to fasten the delivery of any type of digital content, e.g. audio/video streaming media, html pages, images, formatted documents or applications. The content sources could be media companies, large enterprises, broadcasters, web/Internet service provider. Due to the heterogeneous nature of the content to be delivered, various architectures and technologies can be adopted to design and develop a CDN. We now analyze the characteristics of the content and of the applications that most likely take advantages of a CDN architecture.

– *Static web based services*. Used to access static content (static html pages, images, document, software patches, audio and/or video files) or content that change with low frequency or timely (volatile web pages, stock quote exchange). All CDN provider (Akamai Inc., Speedera Inc., AT&T inc., Globix Inc. just to mention some) support this type of content delivery. This 6 N. Bartolini, E. Casalicchio, and S. Tucci type of content can easily be cached and its freshness maintained at the edge using traditional content caching technologies.

– *Web storage services*. Essentially, this application can be based on the same techniques used for static content delivery. Additional features to manage logging and secure file transfer should be added. This type of application can require processing at the origin site or at the edge.

– *File transfer services*. World wide software distribution (patch, virus defi- nition, etc.), e-learning material from an enterprise to all their global employees, movies-on-demand from a large media company, highly detailed medical images that are shared between doctors and hospitals, etc. All these content types are essentially static and can be maintained using the same techniques adopted for static web services.

– *E-commerce services*. The semantic of the query used in browsing a product catalogue is not complex, so frequent query results can be successfully cached using traditional DB query caching techniques. Shopping charts can be stored and maintained at the replica server and also orders and credit card transactions can be processed at the edge: this requires trusted transaction-enabled replica servers. In the authors propose a framework for enabling dynamic content caching for e-commerce site.

– *Web application*. Web transactions, data processing, database access, calendars, work schedules, all these services are typically characterized by an application logic that elaborates the client requests producing as results a dynamic web page. A partial solution to the employment of a CDN infrastructure in presence of dynamic pages is to fill the replica servers with the content that most frequently composes the dynamically generated web pages, and maintaining the application and its processing activity that produces the dynamic pages at the origin server. Another approach is to replicate both the application (or a portion of it) and the content at the edge server. In this way all the content generation process (application logic and content retrieval) are handled by the replica server thus offloading the origin server

. – *Directory services*. Used for access to database servers. For example, in the case of a LDAP server, frequent query results or a subsets of directories can be cached at the edge. Traditional DB query caching techniques may be adopted.

– *Live or on-demand streaming*. In this case the edge server must have streaming capability

**3.1 Streaming Media Content**

Streaming media can be live and on-demand, thus a CDN needs to be able to deliver media in both these two modes. Live means that the content is delivered ”instantly” from the encoder to the media server, and then onto the media client. This is typically used for live events such as concerts or broadcasts. The end-to-end delay is at a minimum 20 seconds with today’s technologies, so ”live mode” is effectively ”semi real-time”. In on-demand, the content is encoded and then stored as streaming media files on media servers. The content is then available for request by media clients. This is typically used for content such as video or audio clips for later replay, e.g., video-on-demand, music clips, etc. A specialized server, called a media server, usually serves the digitalized and encoded content. The media server generally consists of media server software that runs on a general-purpose server. When a media client wishes to request a certain content, the media server responds to the query with the specific video or audio clip. The current product implementations of streaming servers are generally proprietary and demand that the encoder, server, and player all belong to the same vendor. Streaming servers also use specialized protocols (such as RTSP, RTP and MMS) for delivery of the content across the IP network. In streaming media CDNs a replica server must have, at least, the additional functionalities listed below.

– The ability to serve live content such as newscasts, concerts, or meetings etc. either in Multicast or Unicast mode.

– Support for delivery of stored or on-demand content such as training, archived meetings, news clips, etc.

– Caching capability of streaming media. Caching a large media file is unproductive, so typically media files are split in segment. Neighbor replica must be capable to share and exchange segment to minimize the network load and cache occupancy.

– Peering capability to exchange and retrieve content from the neighbor streaming cache in case of cache miss. Streaming cache node can be organized in a hierarchy.

– Media transcoding functionality, to adapt media streams for different client capabilities, e.g., low quality/bandwidth content to dial-up users, high quality/bandwidth to xDSL users.

– Streaming session handoff capability. The typically long life of a streaming session, in presence of user’s mobility, causes the need for midstream handovers of streaming session between replica servers.

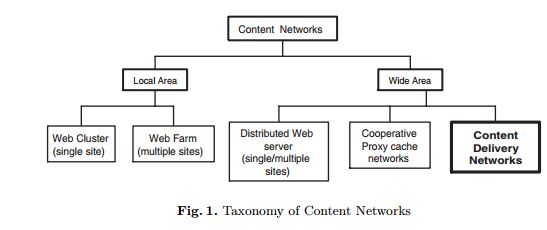
**3.2 Web Application**

Accessing dynamic content and other computer applications is one of the major challenges in CDN. CDN supporting this kind of content and services are also called Application Content Delivery Networks (ACDN). Some providers like AppStream Inc. and PIVIA Inc., implement ACDN using the so called ”fat client” solution: the application is partitioned in ”streamlets” or special applets and sent to the client. The client receives enough code to start the application and execute it, the other parts of the application are sent on demand. These solution use patented and proprietary technologies. Another approach is to migrate the application to the edge server using general utility such as Ajasent and vMatrix. However application replication may be expensive especially if performed on demand. A completely different solution is to automatically deploy the application at the replica server.Authors define an ACDN architecture relying on standard technologies such as HTTP protocol, web servers, CGI/FastCGI scripts or servlets.

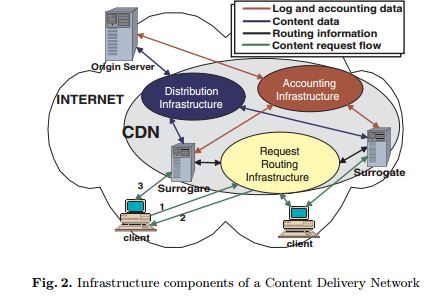
**4. CONTENT DELIVERY NETWORK ARCHITECTURE**

The main goal of server replication in a CDN is to avoid large amounts of data repeatedly traversing possibly congested links on the Internet. As Figure 1 shows, there are a variety of ways and scale (local area or wide area networks) in which content networks may be implemented. Local solutions are web clusters, that typically hosts single site, and web farms, typically used to host multiple sites. Wide area solutions include: distributed web server systems, used to host single or multiple sites; cooperative proxy cache networks (a service infrastructure to reduce latency in downloading web objects) and content delivery networks that are the focus of this paper.

A typical server farm is a group of servers, ranging from two to thousands, that makes use of a so-called cooperative dispatcher, working at OSI layers 4 and/or 7, to hide the distributed nature of the system, thus appearing as a single origin site. A layer 4 web switch dispatches the requests, among a group of servers, on the basis of network layer information such as IP address and TCP port. A content switch, working at the application layer, examines the content of requests and dispatches them among a group of servers. The goals of a server cluster/farm include: load-balancing of requests across all servers in the group; automatic routing of requests away from servers that fail; routing all requests for a particular user agent’s session to the same server, if necessary to preserve session state.



A type of content network that has been in use for several years is a caching proxy deployment. Such a network might typically be employed by an ISP for the benefit of narrow bandwidth users accessing the Internet. In order to improve performance and reduce bandwidth utilization, caching proxies are deployed close to the users. These users are encouraged to send their web requests through the caches rather than directly to origin servers, by configuring their browsers to do so. When this configuration is properly done, the user’s entire browsing session goes through a specific caching proxy. This way the proxy cache would contain the hot portion of content that is being viewed by all the users of that caching proxy. A provider that deploys caches in many geographically locations may also deploy regional parent caches to further aggregate user requests thus creating an architecture known as hierarchical caching. This may provide additional performance improvements and bandwidth savings. Using rich parenting protocols, redundant parents may be deployed such that a failure in a primary parent is detected and a backup is used instead. Using similar parenting protocols, requests may be partitioned such that requests for certain content domains are sent to a specific primary parent. This can help to maximize the efficient use of caching proxy resources. Clients may also be able to communicate directly with multiple caching proxies. Though certainly showing better scalability than a single origin server, both hierarchical caching and server farms have their limits. In these architectures, the replica servers are typically deployed in proximity to the origin server, therefore they do not introduce a significant improvement to the performance difficulties that are due to the network congestion. Caching proxies can improve performance difficulties due to congestion (since they are located in proximity to the final users) but they cache objects reactively to the client demand. Reactive caching based on client demand performs poorly if the requests for a given object, while numerous in aggregate, are spread among many different caching proxies. To address these limitations, CDNs employ a solution based on proactive rather than on reactive caching, where the content is prefetched from the origin server and not cached on demand. In a CDN, multiple replicas host the same content. A request from a browser for a single content item is directed to the replica that is considered the best suited at the moment of the request arrival, and the item is served to the client in a shorter time than the one it would have taken to fetch it from its origin server. Since static information about geographic locations and network connectivity are not sufficient to choose the best replica, a CDN typically incorporates dynamic information about network conditions and load on the replicas, to redirect requests and balance the load among the servers. Operating a CDN is therefore a complex and expensive activity. For this reason a CDN is typically built and operated by a network/service provider that offers a content distribution service to several content providers.



A content delivery architecture consists of a set of surrogate servers that deliver copies of content to the users while combining different activities (see figure 2).

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

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

– *the 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.

The origin server (hosting the content to be delivered) interacts with the CDN in two ways (see figure 2):

– it pushes new content to the replica servers, (the replica themselves request content updates from the origin server through the distribution infrastructure);

– 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. Figure 2 shows one of the possible scenarios of interaction between the clients, the access routers, the replica servers and the origin server. The user agent sends (1) a content request to the routing infrastructure, that redirects (2) the client request to a surrogate server, to which the client subsequently asks (3) the desired content.

**5.HOW CDN WORKS**

Content delivery networks (CDNs) are an important part of Internet infrastructure that are frequently used without a full understanding of what’s happening behind the scenes.

When a web browser makes a request for a resource, the first step is to make a DNS request. The browser gives the domain name and expects to receive an IP address back. With the IP address, the browser can then contact the web server directly for subsequent request.

Physics determines how fast one computer can contact another over physical connections, and so attempting to access a server in China from a computer in the United States will take longer than trying to access a U.S. server from within the U.S. To improve user experience and lower transmission costs, large companies set up servers with copies of data in strategic geographic locations around the world. This is called a CDN, and these servers are called edge servers, as they are closest on the company’s network to the end-user.

**DNS resolution**

When the browser makes a DNS request for a domain name that is handled by a CDN, there is a slightly different process than with small, one-IP sites. The server handling DNS requests for the domain name looks at the incoming request to determine the best set of servers to handle it. At it’s simplest, the DNS server does a geographic lookup based on the DNS resolver’s IP address and then returns an IP address for an edge server that is physically closest to that area. So if I’m making a request and the DNS resolver I’m routed to is Virginia, I’ll be given an IP address for a server on the East coast; if I make the same request through a DNS resolver in California, I’ll be given an IP address for a server on the West coast. You may not end up with a DNS resolver in the same geographic location from where you’re making the request.

That’s the first step of the process: getting the request to the closest server possible. Keep in mind that companies may optimize their CDNs in other ways as well, for instance, redirecting to a server that is cheaper to run or one that is sitting idle while another is almost at capacity. In any case, the CDN smartly returns the best possible IP address to handle the request.

**Accessing content**

Edge servers are *proxy caches* that work in a manner similar to the browser caches. When a request comes into an edge server, it first checks the cache to see if the content is present. The cache key is the entire URL including query string (just like in a browser). If the content is in cache and the cache entry hasn’t expired, then the content is served directly from the edge server.

If, on the other hand, the content is not in the cache or the cache entry has expired, then the edge server makes a request to the *origin server* to retrieve the information. The origin server is the source of truth for content and is capable of serving all of the content that is available on the CDN. When the edge server receives the response from the origin server, it stores the content in cache based on the HTTP headers of the response.

Yahoo! created and open sourced the [Apache Traffic Server](http://trafficserver.apache.org/), which is what Yahoo! uses in its CDN for managing this traffic. Reading through the Traffic Server documentation is highly recommended if you’d like to learn more about how cache proxies work.

**6.CONTENT SERVICE PROTOCOLS**

Several protocol suites are designed to provide access to a wide variety of content services distributed throughout a content network. The [Internet Content Adaptation Protocol](https://en.wikipedia.org/wiki/Internet_Content_Adaptation_Protocol)(ICAP) was developed in the late 1990s to provide an open standard for connecting application servers. A more recently defined and robust solution is provided by the Open Pluggable Edge Services (OPES) protocol. This architecture defines OPES service applications that can reside on the OPES processor itself or be executed remotely on a Callout Server. [Edge Side Includes](https://en.wikipedia.org/wiki/Edge_Side_Includes) or ESI is a small markup language for edge level dynamic web content assembly. It is fairly common for websites to have generated content. It could be because of changing content like catalogs or forums, or because of the personalization. This creates a problem for caching systems. To overcome this problem a group of companies created ESI.

**Peer-to-peer CDNs**

In [*peer-to-peer*](https://en.wikipedia.org/wiki/Peer-to-peer)*(P2P)* content-delivery networks, clients provide resources as well as use them. This means that unlike [client-server](https://en.wikipedia.org/wiki/Client-server) systems, the content centric networks can actually perform better as more users begin to access the content (especially with protocols such as [Bittorrent](https://en.wikipedia.org/wiki/Bittorrent) that require users to share). This property is one of the major advantages of using P2P networks because it makes the setup and running costs very small for the original content distributor.

**Private CDNs**

If content owners are not satisfied with the options or costs of a commercial CDN service, they can create their own CDN. This is called a private CDN. A private CDN consists of POPs that are only serving content for their owner. These POPs can be caching servers, reverse proxies or application delivery controllers. It can be as simple as two caching servers or large enough to serve petabytes of content.

**CDN trends**

*Emergence of telco CDNs*

The rapid growth of [streaming video](https://en.wikipedia.org/wiki/Streaming_video) traffic uses large [capital expenditures](https://en.wikipedia.org/wiki/Capital_expenditures) by broadband providersin order to meet this demand and to retain subscribers by delivering a sufficiently good [quality of experience](https://en.wikipedia.org/wiki/Quality_of_experience).

To address this, [telecommunications service provider](https://en.wikipedia.org/wiki/Telecommunications_service_provider) (TSPs) have begun to launch their own content delivery networks as a means to lessen the demands on the [network backbone](https://en.wikipedia.org/wiki/Network_backbone) and to reduce infrastructure investments.

**7.CONTENT DELIVERY SERVICE PROVIDERS**

**Commercial CDNs**

Akamai Technologies

Cloud Fare

Amazon CloudFront and many more

**Telco CDNs**

At&T

Bharati Airtel

Level 3 Communicatins.. etc.

**8.Video Stream Coding Part**

The Program includes 5 files:

Client.py

ClientLauncher.py

RtpPacket.py

Server.py

ServerWorker.py

What we will do is

1.Run Server.py on Server Terminal to start server:

E.g.

python Server.py server\_port

server\_port is the port your server listens to for incoming RTSP connections

# we can give it the value 1025

# Standard RTSP port is 554

# In this project we shall make the value > 1024

2.Run ClientLauncher.py on Client Terminal to start a client:

E.g

python ClientLauncher.py server\_host server\_port PRT\_port video\_file

server\_host is the IP address of local machine (we can use “127.0.0.1” )

server\_port is the port the server is listening on (here “1025”)

RTP\_port is the port where RTP packets are received (here “5008”)

video\_file is the name of video file that we want to play (here “video.mjpeg”)

**RTSP**

Real Time Streaming Protocol

For entertainment and communications systems to control streaming media servers

Establishing and controlling media sessions between end points

It uses TCP

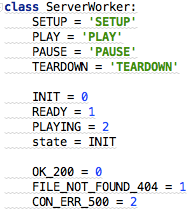
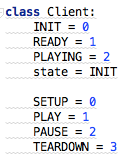
**RTP**

Real-time Transport Protocol

Network protocol for delivering audio and video over IP Networks

It uses UDP

**How RTSP and RTP work together?**

*What will be sent from client to server via RTSP Protocol are the commands like*

SETUP

PLAY

PAUSE

TEARDOWN

These commands will let server side know what is next action it should complete.

*What will be replied from server to client via RTSP Protocol are the parameters like*:

OK\_200

FILE\_NOT\_FOUND\_404

CON\_ERR\_500

To tell the client if the server receive its commands correctly

After client receives server`s reply, it will change its state accordingly to :

READY

PLAYING

If SETUP command was sent from client to server



The “SETUP” RTSP Packet will include

1. SETUP command

2.Video file name to be play

3.RTSP Packet Sequence Number starts from 1

4.Protocol type: RTSP/1.0 RTP

5.Transmission Protocol: UDP

6.RTP Port for video stream transmission

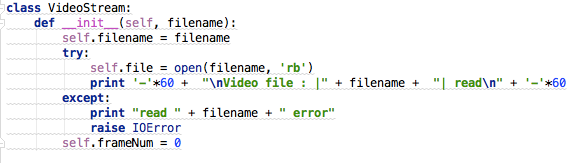


When Server side receives “SETUP” command ,it will

1.Assign the client a Specific Session Number randomly

2.If something wrong with this command or server`s state,it will reply ERROR packet back to client

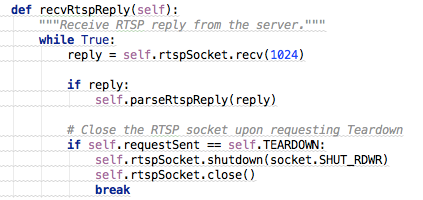
3.If command correct



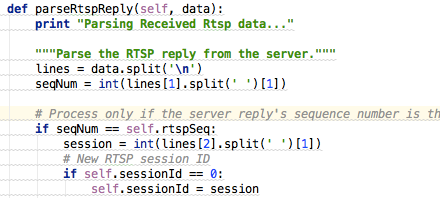
The server will open the video file specified in the SETUP Packet and Initialize its video frame number to 0

3.If command processes correctly, it will reply OK\_200 back to client and set its STATE to READY

The Client side will loop to receive Server`s RTSP Reply

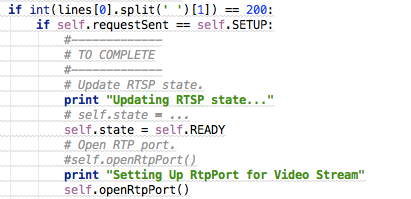


Then Parse the RTSP Relpy Packet:

Get the Session Number 

And if the Reply Packet is respond for the SETUP command

The client will set its STATE as READY

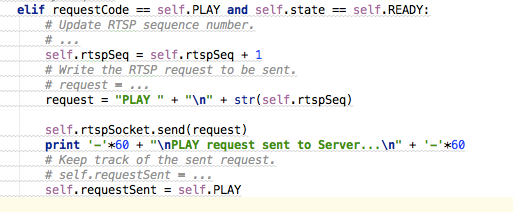


Then open a Rtp Port to receive incoming video stream

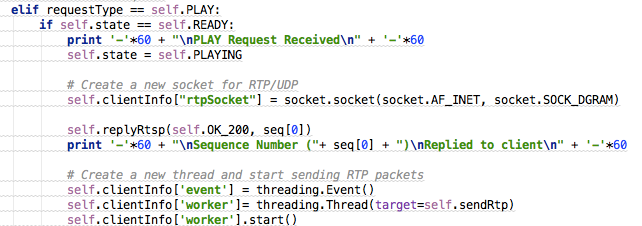


Afterward

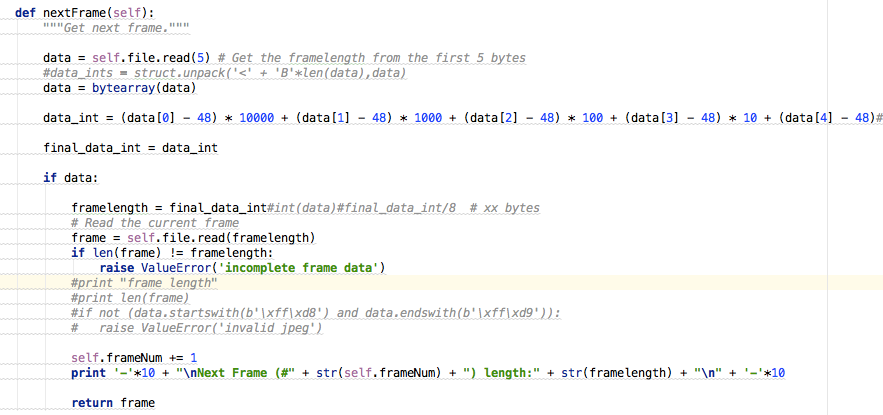
If PLAY RTSP command was sent from client to server:



The Server will create a Socket for RTP transmission via UDP, and start a tread to send video stream packet



VideoStream.py will help chop the video file to separate frame , and put each frame into RTP data packet



Each data packet will also be encoded with a header, the header will include

RTP-version filed

Padding

extension

Contributing source

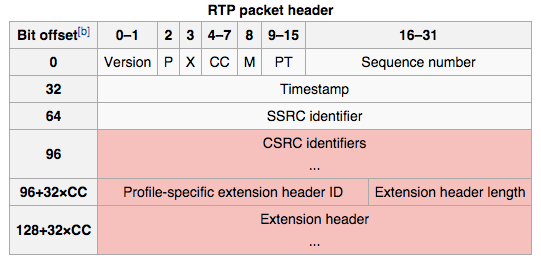
Marker

Type Field

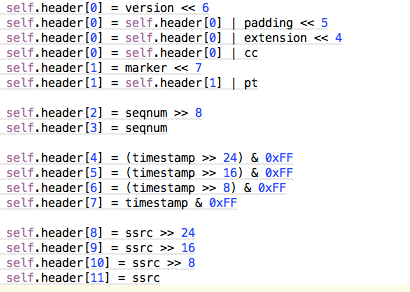
Sequence Number

Timestamp

SSRC



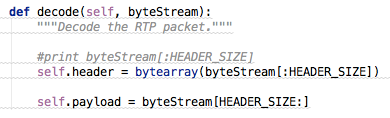
they have been inserted in the RTP Packet via bitwise operations



Finally the RTP Packet will include a header and a video frame be sent to the RTP Port on the client side:



Then Client decode the RTP Packet to get the header and the video frame, reorganize the frames and display on the UI



If a PAUSE command was sent from client to server , it will stop the server from sending video frames to client

If a TEARDOWN command was sent from client to server, it will also stop the server from sending video frames to client and close the client terminal as well

**8.REFERENCES**

1. http://wwwusers.di.uniroma1.it/~novella/myhome/Home\_Page\_di\_Novella\_Bartolini/papers/CDN\_tutorial.pdf
2. http://www.hit.bme.hu/~jakab/edu/litr/CDN/CDN-Taxonomy.pdf
3. http://www.webperformancetoday.com/2013/06/12/11-faqs-content-delivery-networks-cdn-web-performance/
4. https://en.wikipedia.org/wiki/Content\_delivery\_network
5. http://ieeexplore.ieee.org/xpls/abs\_all.jsp?arnumber=1250586