POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE E
DELL'INFORMAZIONE

TESI DI LAUREA MAGISTRALE IN TELECOMMUNICATIONS
ENGINEERING

PEER TO PEER STREAMING PEER PROTOCOL

Supervisor:
Prof. Paolo Giacomazzi

Livin Varghese
Student number: 814321

Academic Year: 2017- 2018

ABSTRACT

Peer to Peer technology had a great impact over the internet since the innovation and bulk quantity of data is traded daily on this principle. In Peer to Peer Streaming Peer Protocol, it mainly focused on delivering the dynamic content to the clients in a streaming pattern which are interested to initiate the download. It provides the support for the pre recorded and live contents. The Peer structure follows the client part which employs the content in order to generate a system with user friendly bandwidth. Also to protect the interruption caused by the faulty peers when considering a short time till playback. There are different mechanisms available in order to improve peer uploading, protects free riding and the peer discovering schemes like Centralized trackers and Distributed hash tables. PPSPP supports content integrity protection methods and the chunk addressing schemes. It is drafted as a universal protocol which runs over other protocols and currently on the top of UDP using LEDBAT (Low Extra Delay Background Transport) in order to prevent from the congestion control. Advancement in the peer to peer technology in order for better performance and to enable large scale content streaming resulted in Peer. Peer5 operates as a service that orchestrates peers and provides analytics and control. The service is triggered using JavaScript API in the browser. The client side contains custom WebRTC p2p logic, an HTTP client and a delivery manager that improves speed using the two methods (P2P and HTTP). The client connects to multiple peers and to the HTTP server simultaneously and ensures chunks are received in time for playback.

Keywords: Peer to Peer, Streaming, Peer5, WebRTC, Merkle Hash tree, Content integrity protection, Live streaming, Protocol options. UDP encapsulation.

ACKNOWLEDGEMENTS

This thesis marks the end of the remarkable journey I took in the academic years carrying out my master study. Its successful completion, however, would not have been possible without the support of several people.

First, I would like to express my gratitude to my family for their endless support all through my life right from my coming into this world until making me reach to this stage. No words can be enough for what they have contributed to my life and I do not think that it should be put into just words.

I would like to thank Politecnico di Milano for offering such a nice platform and Scholarship for offering financial aid to cover my tuition fee and living expenses during my stay in Italy without which my degree would not have been realized.

My dearest gratitude is for Prof. Paolo Giacomazzi for his endless support and guidance, constant attention and help in the course of this thesis. Without his encouragement, cooperation and time, it would have been difficult for me to bring this work to completion. He is a really a person to be revered both as a human being and a professional with excellent vigilance and professional skills. No wonder that he is the Master of what he does, and his vision is something to be taking an inspiration from and I just wish I could be as qualified and skilled and compassionate as him some day in my life.

Last but foremost, my heartiest thanks to all my friends at the university especially Jince George, Jithin Varghese, Godwin Jacob Bishnu, Fajo.F Nellissery and Mithoon Vijay. It’s all because their support that I am able to complete my degree. Not only in the studies and thesis, they also helped me and supported me through all thick and thin.

Table of Contents

ABSTRACT ii
ACKNOWLEDGEMENTS iii
1 INTRODUCTION 7
2 STATE OF THE ART 8
2.1 Peer 5 8
2.2 How Do Traditional CDNs Work? 8
2.3 How Does Streaming Work? 9
2.4 What Is Different About Peer5? 10
2.5 A Hybrid Model : Servers + Peer-to-Peer 11
2.6 The Role of WebRTC 12
2.7 Buffer Management and Zero Delay 13
2.8 Security 13
2.9 Peer Efficiency 15
2.10 Robust Routing Policies 16
2.11 Multiplatform Support 16
2.12 Peer5 highlights 16
2.13 Summary 17
3 PEER TO PEER STREAMING PEER PROTOCOL : RFC 7574 18
3.1 Introduction 18
3.2 Terms used 18
3.3 Functional operation 20
3.4 Messages 21
3.5 Chunk Addressing Schemes 22
3.6 Content Integrity Protection and Verification 23
3.7 Live Streaming 25
3.8 Protocol Options 26
3.9 UDP Encapsulation 29
3.9.1 FLOW AND CONGESTION CONTROL 33
3.10 Manageability Considerations 34
3.11 Security Considerations 35
4 CONCLUSION 38
5 REFERENCES 40

LIST OF FIGURES

Figure 2-1:Traditional CDN 9
Figure 2-2: Streaming 9
Figure 2-3: P2P capacity graph 10
Figure 2-4: Hybrid model 11
Figure 2-5: HLS player buffer 13
Figure 2-6:Peer5 security 14
Figure 3-1: bin number tree 23
Figure 3-2: bin representation 32/64 bit 23
Figure 3-3:Merkle hash tree 24

LIST OF TABLES

Table 1:Peer efficiency 15
Table 2: PPSPP options 26
Table 3:Message types 29

1 INTRODUCTION

Peer-to-Peer is a huge success of the Internet and the content in impressive quantities is traded daily with techniques based on this principle. However, it is mainly about downloading (to obtain a static content) whereas one would like to be able to use the same concept, the peer-to-peer, for streaming, in order to distribute dynamic content, like an event happening and filming. This is allowed by the new Peer-to-Peer Streaming Peer Protocol (PPSPP), which is standardized by the RFC 7574. Its purpose is to provide the equivalent of BitTorrent for dynamic content, and many of the PPSPP concepts are very close to BitTorrent (from the RFC 6972). In PPSPP, the content is "self-certified". The identifier of a content allows its verification. Unlike static content, we cannot use a simple cryptographic condensate (BitTorrent method) since we do not know all the bytes in advance. PPSPP therefore uses a recursively calculated Merkle tree as the content becomes available. The identifier of the content is the cryptographic condensate of the root of this Merkle tree. With the self-certification, a malicious peer will not be able to modify the content (it remains to make sure of the authenticity of the identifier.)

Based on the advancements Peer5 technology is introduced. Peer5 is known “Server less CDN” because no servers are involved in the delivery of bytes through our network. However, we don’t aim to completely get rid of servers. Servers do the important work of ingesting, encoding, transmuxing and more. We believe that the optimal content delivery solution for broadcasters is a hybrid model where Peer5 is used in combination with their existing HTTP-based CDNs. Our goal is to enable limitless video delivery by unifying the existing server infrastructure with an elastic computing layer (comprised of individual viewers) that grows with demand.

Peer5 works alongside of a publisher’s origin server, CDN or Multi-CDN architecture. WebRTC is used to create a peer-to-peer mesh network that helps users load video content from each other. The hybrid switching algorithm determines whether a viewer should load the next segment from Peer5’s p2p network or the publisher’s alternative delivery system. This allows Peer5 to shrink a content provider’s bandwidth usage, while also maximizing a user’s viewing experience.

2 STATE OF THE ART

2.1 Peer 5

Peer5 is the world's largest peer-to-peer (p2p) content delivery network (CDN) for video streaming. Peer5 deliver over 90 million hours of video every month. The Peer5 service improves video quality and reduces bandwidth costs by 50%. Unlike older p2p solutions, Peer5 requires no end-user downloads or plugins.

Peer5 known to be the “Server less CDN” because no servers are involved in the delivery of bytes through our network. However, we don’t aim to completely get rid of servers. Servers do the important work of ingesting, encoding, transmuxing and more. We believe that the optimal content delivery solution for broadcasters is a hybrid model where Peer5 is used in combination with their existing HTTP-based CDNs. Our goal is to enable limitless video delivery by unifying the existing server infrastructure with an elastic computing layer (comprised of individual viewers) that grows with demand.

Peer5 works alongside of a publisher’s origin server, CDN or Multi-CDN architecture. WebRTC is used to create a peer-to-peer mesh network that helps users load video content from each other. The hybrid switching algorithm determines whether a viewer should load the next segment from Peer5’s p2p network or the publisher’s alternative delivery system. This allows Peer5 to shrink a content provider’s bandwidth usage, while also maximizing a user’s viewing experience.

2.2 How Do Traditional CDNs Work?

Traditional CDNs such as Akamai / Level 3 / Limelight / etc. all work the same way - they deploy HTTP servers throughout the world and cache (or store) content (images, text, videos, etc.) on these servers. The idea is to move the content physically closer to end-users (no matter where they are in the world) so that such content can be downloaded more quickly by the end-users. Companies that want their content to load faster pay for storage on the CDN’s servers. When an end-user visits a website that has been cached by a CDN, his browser’s HTTP requests will be redirected to the nearest CDN edge server and the site will load faster (versus a site that isn’t cached).

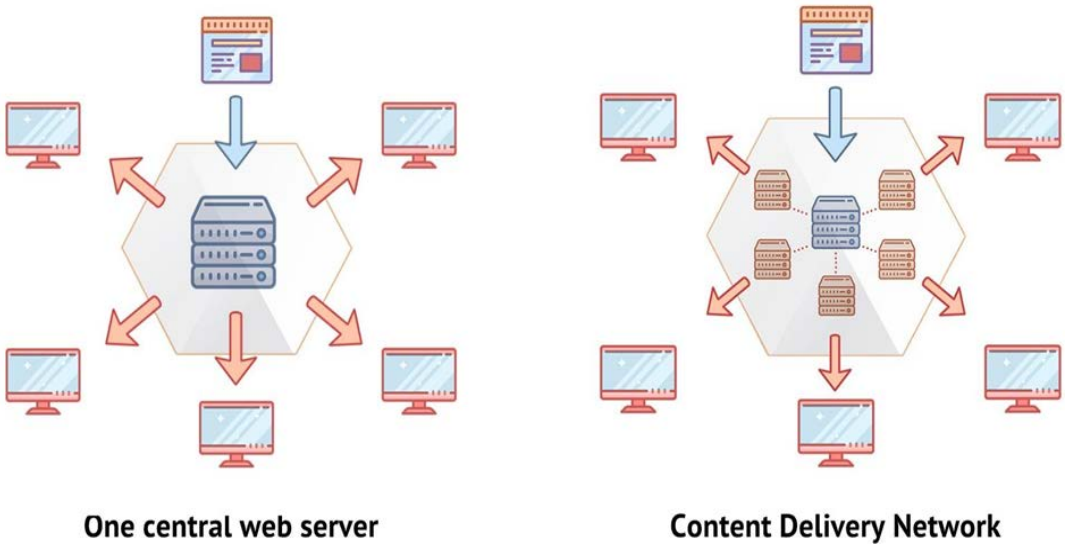


Figure 2-1:Traditional CDN

2.3 How Does Streaming Work?

Industry standard streaming protocols such as HLS (HTTP Live Streaming) and MPEG-DASH are built on the concept of segmentation. A piece of content (like a live sports event) is cut into individual slices of video. These slices (aka “segments” or “chunks”) are typically 10 seconds long, but segment duration is configurable. The basic principle is that you’re downloading lots of small files instead of a single large file. There is a “manifest” file that tells your video player where to fetch the individual segments - e.g., the URL of segment 1, the URL of segment 2, etc. - and your video player makes HTTP requests to fetch these segments. These HTTP requests are routed to the nearest CDN edge server where the segments are cached and the segments are downloaded to your computer. Once received, these segments go into a buffer and are extracted as needed and stitched together by your video player to give you the smooth streaming experience you’ve come to expect

Figure 2-2: Streaming

2.4 What Is Different About Peer5?

Peer5 is a peer-to-peer (p2p) CDN that doesn't use any HTTP servers. In fact, we describe ourselves as the "Server less CDN". We believe that the only way to make streaming video as scalable as broadcast television - able to reach hundreds of millions or billions of simultaneous viewers - is to employ a p2p architecture that enables streaming capacity to grow proportionally with streaming demand:

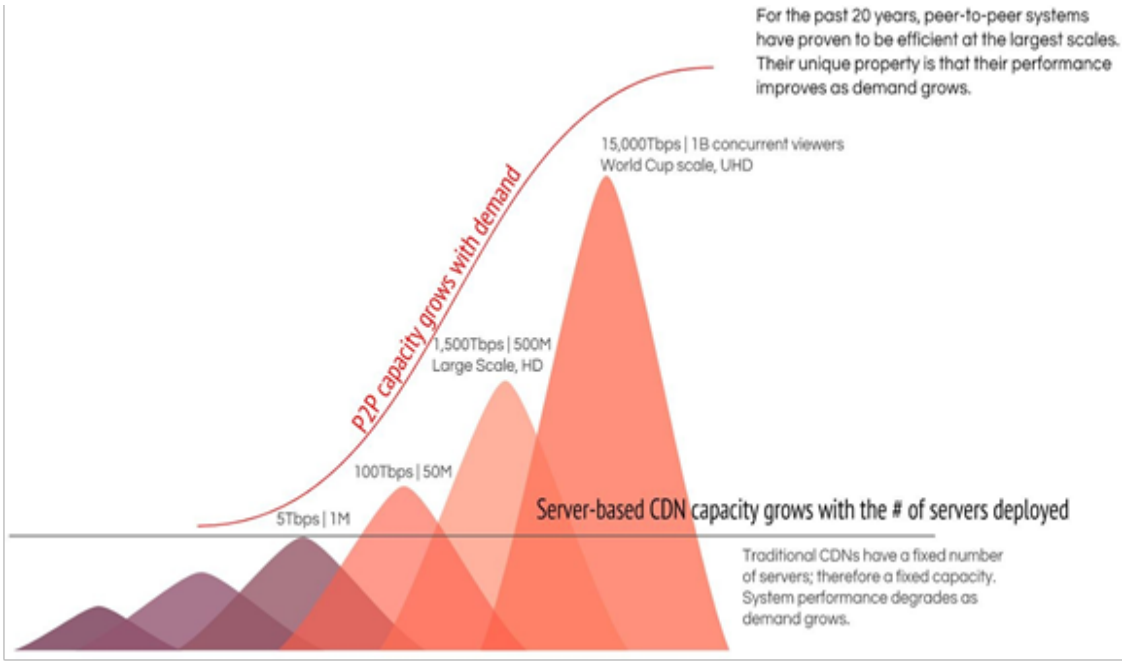


Figure 2-3: P2P capacity graph

As explained above, modern streaming relies on segmentation and the existence of a buffer where individual video chunks are stored until they are used. Imagine a situation where lots of people are trying to stream the same episode of Game of Thrones. Everyone who is watching the same content at the same time needs to download the same underlying segments. At Peer5, we simply ask the question: why should everyone connect to the CDN edge server to download the same segments over and over again when it's possible (and frequently more efficient) to fetch the segments from other viewers?

Peer5 works by creating an elastic mesh network consisting of users (peers) who are watching the same content and coordinating these users so that they can share video segments with each other.

2.5 A Hybrid Model : Servers + Peer-to-Peer

The term “Server less CDN” is to highlight that no HTTP servers are involved in the delivery of bytes via the Peer5 network. However, the goal is not to completely eliminate servers. Servers do the important work of ingesting, encoding, transmuxing and more. The optimal content delivery solution for broadcasters is a hybrid model where Peer5 is used in combination with their existing HTTP-based CDNs. Main goal is to enable limitless video delivery by unifying the existing server infrastructure with an elastic p2p computing layer that grows with demand. This unification is accomplished via two lines of JavaScript.

This client side JavaScript implements a hybrid switcher. Once Peer5 is deployed, users can fetch video segments from either the CDN edge server or from their peer group as depicted here:

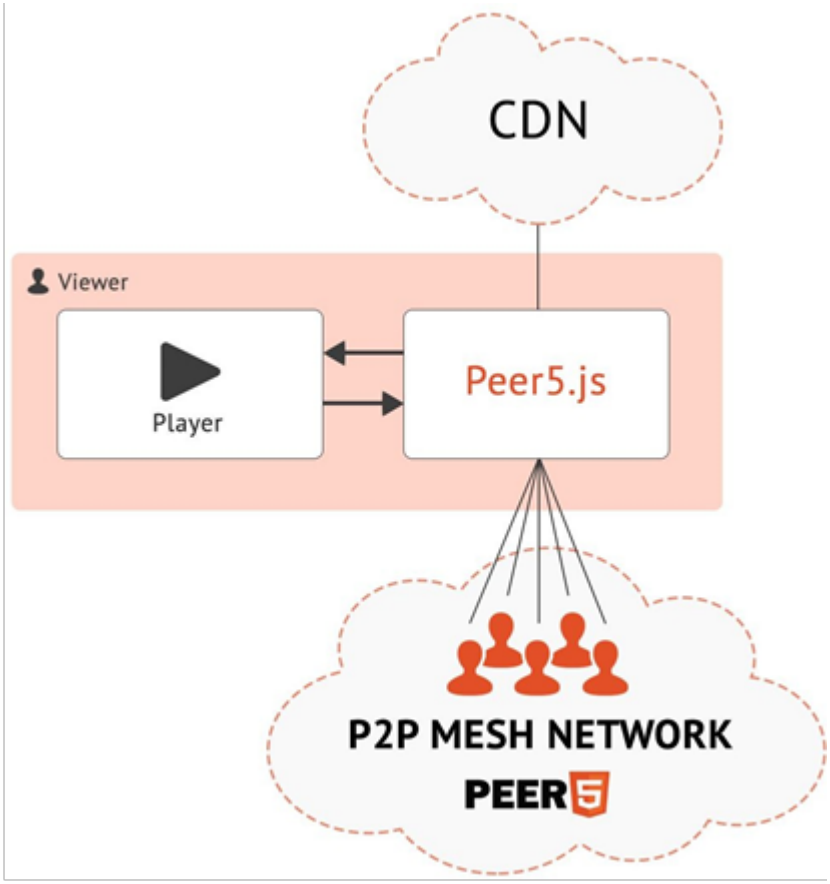


Figure 2-4: Hybrid model

As more and more chunk requests are handled by the peer group, the broadcaster sees dramatic reductions in server load and, consequently, higher stream quality (faster load times, less buffering, longer viewing sessions) for all viewers.

Since Peer5 is SaaS and uses only JavaScript, the client-side integration is transparent to the end-user (no plug-ins) and only requires a small change to the broadcaster's HTML page.

2.6 The Role of WebRTC

HTTP is based on a client-server communication model - an HTTP client (web browser) makes a request to an HTTP server and gets a response. For years, the only communication protocol that web browsers supported was HTTP. This meant that browsers could only communicate with HTTP servers. While this client-server HTTP communication model has enabled the Web to become the world changing communication network that it is today, we're beginning to see strains in the system, mostly driven by the explosive growth of streaming video.

Today, video accounts for 70% of all Internet traffic and broadcasters are already encountering huge scalability and stream quality issues due to the client-server HTTP approach. Basically, it is becoming impossible to deploy HTTP servers fast enough to keep up with the demand from HTTP clients.

A different approach is needed and into the void stepped Google with WebRTC which enables individual web browsers to communicate directly with each other, without having to pass through an intervening server. Google Hangouts is built on top of WebRTC, as are services such as Snapchat, WhatsApp and Facebook Messenger. While it is not widely recognized (even among people in the computing industry), we believe that WebRTC will eventually become as important to the Internet as HTTP. In fact, WebRTC has been (1) incorporated into the HTML5 standard and (2) supported in every version of Chrome, Firefox and Opera (desktop and mobile) for the last 3 years. In 2017, both Microsoft and Apple decided to support WebRTC as well. Ubiquity for WebRTC means ubiquity for Peer5 as well.

When Google introduced WebRTC back in 2012, we saw an opportunity to create a truly seamless p2p service. Now that WebRTC is deployed everywhere, our vision of creating a massively distributed and infinitely scalable video CDN has been realized!

2.7 Buffer Management and Zero Delay

Peer5 will only fetch segments from peers when there is no risk of harming the user experience (UX). When there are no peers, or when the UX might deteriorate, Peer5 falls back to HTTP delivery.

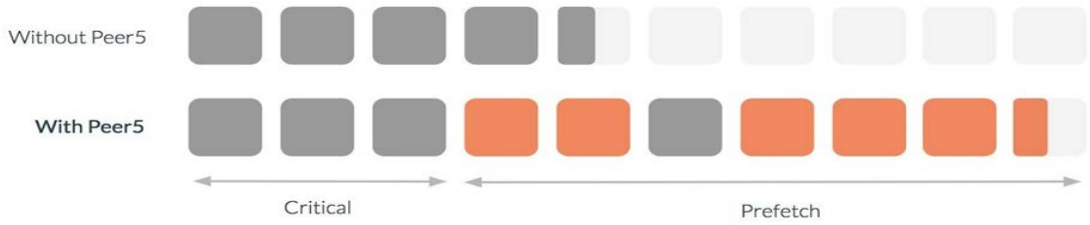


Figure 2-5: HLS player buffer

The diagram above shows a typical HLS player buffer (both with and without Peer5), and illustrates how Peer5 divides the buffer into a critical portion and a prefetch portion. Without Peer5, all segments are fetched via HTTP (grey squares).

With Peer5, one can see hybrid delivery in action, as evidenced by the presence of orange squares (segments fetched via p2p) in the prefetch portion of the buffer. Because HTTP delivery is used to fill the critical portion of the buffer, Peer5 does not add any delay to the playback experience. Once the critical portion is filled, however, Peer5 leverages p2p delivery to dramatically reduce HTTP traffic.

In cases where the end-user cannot fetch segments via HTTP fast enough to fill the critical portion of the buffer, either because he's geographically far from the server or the server is overloaded, p2p prefetching will improve UX since the end-user will have more sources from which to download the required segments.

2.8 Security

The Peer5 service is just as secure as any traditional, server-based CDN service. Because Peer5 is a hybrid solution (one uses Peer5 in combination with a traditional HTTP server), we leverage the existing security infrastructure (tokens, keys, cookies, etc.) that a broadcaster already has in place, as depicted here:

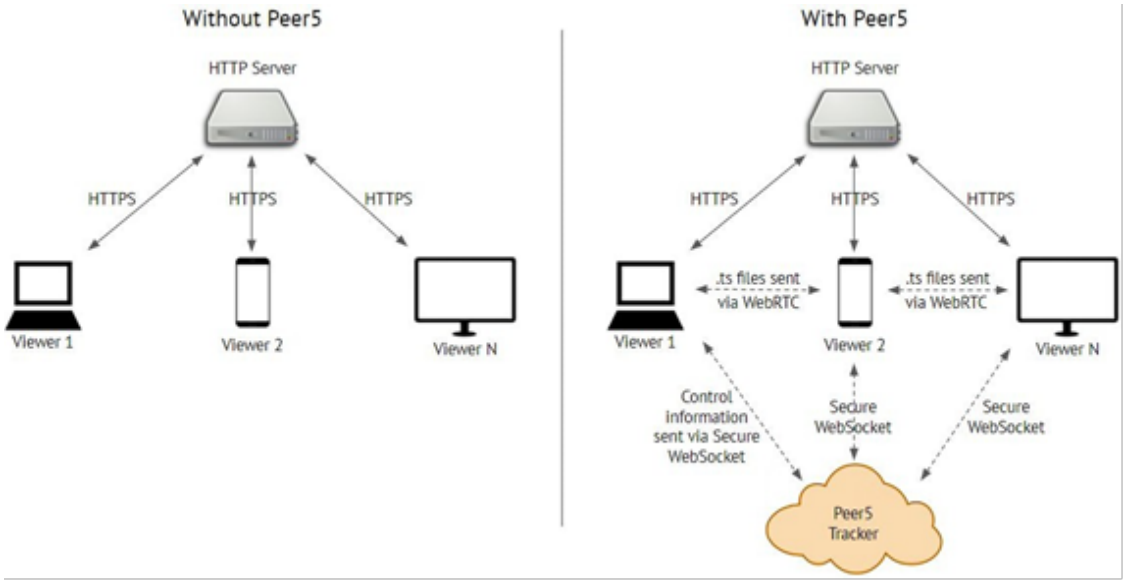


Figure 2-6:Peer5 security

After Peer5 is deployed, each video session still starts with a standard exchange between the viewer and the HTTP server. If user authorization was required before Peer5 was deployed, then authorization will be required after Peer5 is deployed. P2P activity only starts to work AFTER the user is authorized by the server. The HTTP server sees the same exact requests (tokens, keys, cookies, etc.) before and after Peer5 is enabled. In short, the existing content protection and geo-fencing schemes remain exactly the same after Peer5 is deployed.

Peer5 does not interfere with DRM and stream encryption because the video segments sent between users are the same as the segments that users receive from the HTTP server. The Peer5 service doesn't compromise the stream in any way since the only users that can view the stream (and share segments with each other) are the ones who have been authenticated and authorized by the HTTP server and have received a decryption key from the server. Decryption keys are never transferred through the Peer5 network. Content (video segments) are also never stored locally or in any persistent data storage.

Sharing of video segments between viewers is done via the WebRTC data channel. The data channel is secured using the SCTP protocol and TLS encryption. For a discussion of WebRTC security, please see A Study of WebRTC Security. Communication between viewers and the Peer5 backend (our cloud-based “Tracker”) is done via a secure Web Socket connection, which also uses TLS encryption.

2.9 Peer Efficiency

Peer Efficiency is the percentage of content delivery that is offloaded to the p2p network at a given time. HTTP-based CDNs use the term “cache hit ratio”, which is a very similar concept. CDNs usually measure the ratio as requests served from the CDN divided by the total number of requests. Peer Efficiency is calculated by dividing the number of bytes delivered from the p2p network by the total number of bytes delivered (p2p + HTTP). As Peer Efficiency increases, one is able to achieve massive scalability gains while simultaneously improving the stream quality for each viewer, as illustrated in the table below where we assume viewers receive a 1 Mbps stream from a single origin server:

If you’re able to off-load half of the requests for segments to the p2p network (50% Peer Efficiency), then your origin server is working 50% less than it was before and you can now service an audience that is twice (2x) as large without having to deploy additional hardware. If peer efficiency reaches 99%, then your server is doing 1% of the work it previously did and you can now service an audience that is 100x bigger (1 / 0.01 = 100). Because it is a logarithmic function, small gains in Peer Efficiency (e.g., from 99% to 99.9%) result in an order of magnitude improvement in scalability (100x to 1000x).

Peer5 recently broadcasted an event that reached 2 million concurrent users and achieved 99% Peer Efficiency - 2 milestones that no other p2p CDN has achieved.

Peer Efficiency (% Off-load)	Avg p2p Bitrate (Mbps)	Avg HTTP Bitrate (Mbps)	Increase in Scalability
0	0.00	1.00	None
50	0.50	0.50	2x
90	0.90	0.10	10x
98	0.98	0.02	50x
99	0.99	0.01	100x
99.9	0.999	0.001	1000x

Table 1:Peer efficiency

While Peer Efficiency is typically highest during large events (i.e., the most popular content), Peer5 has achieved excellent off-loading ratios even when there is a small number of concurrent viewers. In fact, Peer5 has seen 50% Peer Efficiency for streams

with only 20 concurrent viewers. This means that even less popular content can be efficiently delivered with Peer5.

2.10 Robust Routing Policies

The Peer5 platform offers very fine-grained control over how IP traffic is routed. We can tell whether a peer is connected via a cellular (3G / 4G), WiFi or wired connection and can implement any business rule to govern how and when peers share (or don't share) video segments. For example, our customers can:

- Turn Peer5 on or off globally
- Turn Peer5 on or off for specific connection types (cellular, WiFi, landline)
- Enable Peer5 only for a specific % (or subset) of their viewers
- Enable Peer5 only for certain whitelisted domains or specific web pages
- Enable Peer5 only for viewers using a specific operating system / browser / video player
- Create custom peering schemes to control how viewers are grouped into swarm

2.11 Multiplatform Support

Peer5 can be incorporated into virtually every major streaming platform, including:

- Web browsers: Chrome, Firefox, Opera, Safari
- Android and Android TV
- Apple iOS and tvOS
- Google Chromecast
- Amazon FireTV
- Samsung Tizen
- LG webOS

2.12 Peer5 highlights

Easy to implement; only 2 lines of JavaScript must be added to the web page where your video player resides

- Support for HLS and MPEG-Dash streaming
- Support for multiple platforms: Android, iOS, Chromecast, FireTV, Tizen, webOS, HTML5
- Support for all delivery architectures (Origin Server, Single CDN, Multi-CDN)
- DRM agnostic

- Broadcasters don’t need to modify their content preparation workflow or any server-side settings
- Can be used for live and on-demand content
- Ubiquitous geographic coverage: video delivered into 235 out of the 249 countries and territories on Earth
- 100% supply elasticity: additional capacity is available the instant you need it
- Use of WebRTC means end-users don’t have to install any plug-ins
- Sustained over 2 million concurrent viewers
- Serve over 500M video session per month
- Average Peer Efficiency is 70%; Peak Peer Efficiency is 99%

2.13 Summary

Peer5 enables large-scale video streaming. Peer5 does not replace traditional HTTP-based CDNs, but, rather, works in combination with them to increase scalability by a factor of up to 100x, delivering a TV-grade broadcasting solution for the Internet.

Typical results:

- 10x more capacity during peak hours
- 23% reduction in buffering occurrences
- 50% costs savings

3 PEER TO PEER STREAMING PEER PROTOCOL: RFC 7574

3.1 Introduction

Peer-to-Peer is a huge success of the Internet and the content in impressive quantities is traded daily with techniques based on this principle. However, it is mainly about downloading (to obtain a static content) whereas one would like to be able to use the same concept, the peer-to-peer, for streaming, in order to distribute dynamic content, like an event happening and filming. This is allowed by the new Peer-to-Peer Streaming Peer Protocol (PPSPP), which is standardized by the RFC 7574. Its purpose is to provide the equivalent of BitTorrent for dynamic content, and many of the PPSPP concepts are very close to BitTorrent.

In PPSPP, the content is "self-certified". The identifier of a content allows its verification. Unlike static content, we cannot use a simple cryptographic condensate (BitTorrent method) since we do not know all the bytes in advance. PPSPP therefore uses a recursively calculated Merkle tree as the content becomes available. The identifier of the content is the cryptographic condensate of the root of this Merkle tree. With the self-certification, a malicious peer will not be able to modify the content (it remains to make sure of the authenticity of the identifier.)

PPSPP is an application protocol that can run on several transport protocols. At present, the only standard is LEDBAT (RFC 6817), the "durable" protocol, which sends bytes only when the pipe is free. Thus, the use of PPSPP will not interfere with other applications. (Section 8 details the use of LEDBAT on UDP.)

PPSPP can discover the list of peers in several ways: via a centralized tracker, or via a DHT. This new RFC assumes the peer list already known and only describes the protocol to retrieve the content.

3.2 Terms used

Message: It is the fundamental unit of PPSPP communication. There are several representations on the wire that depend on the transport protocol used. Messages are usually multiplexed in a datagram for transmission.

Datagram: A sequence of messages that is offered as a unit to the underlying transport protocol (UDP, etc.). The datagram is PPSPP’s Protocol Data Unit (PDU).

Content: It contains a live transmission or a prerecorded multimedia file.

Chunk: It represents the division of content.

chunk ID: A unique identifier is given for the chunks of content. And the type depends on the scheme used.

chunk specification: An expression which represents one or more chunk IDs.

chunk addressing scheme: Scheme used for detecting chunks and expressing the chunk availability map of a peer in a compressed fashion.

chunk availability map: It represents the set of chunks that has successfully downloaded and completed the Integrity check.

Bin: A number representing a specific binary interval of the content.

content integrity protection scheme: Scheme to protect the integrity of the content as it is distributed via the peer-to-peer network.

Merkle hash tree: A tree of hashes whose base is formed by the hashes of the chunks of content, and its higher nodes.

root hash: The root in a Merkle hash tree calculated recursively from the content.

munro hash: The hash of a subtree in the Unified Merkle Tree content authentication scheme for live streaming.

Swarm: A collection of peers take part in the distribution of the same content.

swarm ID: It is the unique identifier for a swarm of peers. Merkle hash tree is the unique identifier used for video on demand with content integrity protection enabled. The swarm ID functions as a public key for live streaming.

Tracker: It records the addresses of peers contributing in a swarm or a set of swarms. And it provides the necessary information to other peers on request.

Choking: When Peer A is choking Peer B, it means that A is currently not willing to accept requests for content from B.

Seeding: Peer A is said to be seeding when A has downloaded a static content file completely and is now offering it for others to download.

Leeching: Peer A is said to be leeching when A has not completely downloaded a static content file yet or is not offering to upload it to others.

Channel : Channel is the logical connection between two peers. And it permits peers to use the same transport address for communicating with different peers.

channel ID: It is the unique identifier for a channel. So the two peers logically connected by a channel and each have a different channel ID for the channel.

heavy payload: A datagram has heavy payload when it contains DATA, SIGNED_INTEGRITY or a large number of smaller messages.

3.3 Functional operation

Operation of PPSPP starts with the fundamental unit message. The transmission of several messages are multiplexed in a datagram using the UDP transport protocol. Merkle hash tree Scheme is used for the content Integrity protection. And a specific policy is introduced for downloading the required chunks of content.

The functionalities of PPSPP RFC7575 are explained using the following 3 examples:

In the first example, a peer joins a swarm. A human user looks at a Web page that contains a <video> element. When the user clicks the play button. The URL behind the video indicates PPSPP. The browser will then retrieve the list of peers using a centralized tracker. The PPSPP software will register with this tracker. User has a list of peers and can start communicating each other with the HANDSHAKE message. Then the user will receive HAVE messages indicating which parts of the data stream the user peers already know and can therefore provide to their peers.

In the Second example, exchange of data takes place. Here PPSPP software will send REQUEST messages with respect to the HAVE messages received. The peers send DATA messages containing the bytes of the video, as well as INTEGRITY messages containing the elements of the Merkle tree and verifying the content in which the mediators have not modified the video. Once our software has data, it can upload. It will in turn send HAVE messages to peers, and respond to REQUEST requests. The "clean" start of the swarm is done by sending a HANDSHAKE message (and by separating from the tracker, if there is one). But, of course, in a peer-to-peer network, machines often do not leave cleanly: network shutdown, crash of machine or software, and so on. Peers must be prepared to handle the case of unresponsive peers.

In the Third example, a Peer leaves a Swarm. Port of Peers removes a Peer A from the peer list. Here, the Peer A deregister itself from the tracker using a specific HANDSHAKE message.

3.4 Messages

In the Section Messages of RFC7574 presents all the messages that can be exchanged between the peers which enhances the uninterrupted communication. There are twelve types of messages involved in PPSPP. A lack of information from the peer indicates that there is a problem exists in enhancing a reliable communication between the peers.

The different types of messages are discussed below:

HANDSHAKE is one of the most important type of message used in PPSPP. It enhances the communication between two peers using a HANDSHAKE procedure. If a HANDSHAKE message is send to peers which are not interested in receiving the message, here the peers will not respond with an error message but they will simply refrain from answering. So there are two kinds of peers exist here: one of those respond quickly and the other respond little or not, and so we will gradually decide to ignore. We do not insist on nonresponding peers.

HAVE, DATA and ACK are the other types of messaged involved in the communication. HAVE messages allows the peer to find the available chunks for the download. The availability of chunks are expressed by the chunk addressing and chunk availability map schemes. DATA messages are useful for the transfer of chunks of content. So, it must contain chunk ID. Whereas the ACK messages are used to acknowledge a peer on receiving chunks over a unreliable or reliable transport protocol.

Points to be noted here: Each swarm transmits a stream of bytes. A swarm has an identifier (swarm ID), exchanged in HANDSHAKE messages. These messages also contain other parameters such as the size of chunks or the exact function used for the Merkle tree. The stream of bytes is indeed cut into pieces, each having an identifier (chunk ID). This identifier is a parameter of HAVE, REQUEST and DATA messages. Thus, a DATA message contains Chunks, and the identifier of each chunk. This allows the peer to know the location and the information missing.

INTEGRITY message which contains a cryptographic condensate of a subtree of the Merkle tree corresponding to the transmitted content. INTEGRITY messages are not signed so they only protect against accidents, not against a malicious peer. There are SIGNED_INTEGRITY messages which are signed. Note that, in this case, the public key is in the swarm identifier, encoded according to the DNSKEY format of DNSSEC.

The REQUEST messages mentioned above are used to request a portion of the data stream from another peer. Unlike BitTorrent, they are not essential: a peer can send you

songs that you have not explicitly requested. This is logical for PPSPP, which is intended for runoff: PPSPP is more push than a pull model.

CANCEL messages are introduced to eliminate the previous request of data send to the Peer. So, it considers only the current request for the Peer holding HAVE messages.

CHOKE and UNCHOKE messages represents the willingness to respond to the REQUEST message from one Peer to another one. If Peer A is to able send a CHOKE message to Peer B, then it will no longer be able to respond to the REQUEST messages send from the Peer A. With the UNCHOKE message Peer A will be able to respond to REQUEST message from Peer B but it removes the old requests which are in queue.

Peer Exchange (PEX) messages are used to exchange the transport addresses between the Peers in which they are communicating each other. It runs on the internet and also in fair private networks. In order to sustain a reliable PEX messages on the internet, a stable Peer transport addresses are required and also a Peer is not allowed to obtain other Peer addresses through PEX messages.

Channels are the multiplexing scheme which allows multiple swarms to use the same transport address. Here the two peers connected by a channel which uses two different ID’s to indicate the same channel. ID’s are made different for Peers in order to provide better security.

A Peer which sends Keep alive messages are intended to keep a signaling channel which is open to the Peers that are in interest to communicate each other.

3.5 Chunk Addressing Schemes

Chunk Addressing schemes are used for the implementation for finding chunks and chunk availability map of a Peer. Same Chunk Addressing method is introduced for all the Peers in a swarm. Start-End ranges are used to identify the range of chunks. The two available schemes are: Chunk ranges and Byte ranges. Chunk identifiers are 32 bit or 64 bit unspecified integers whereas the byte offsets in the content are 64 bit.

Bin Numbers are in which the PPSPP introduces in order to address the chunks of content. It uses the numbering system which permits PPSPP to run with simpler data structures. A bin number is a very clever mechanism for designating an almost arbitrary byte interval with a single number. The bin number tree of an interval with width W=8 is given below:

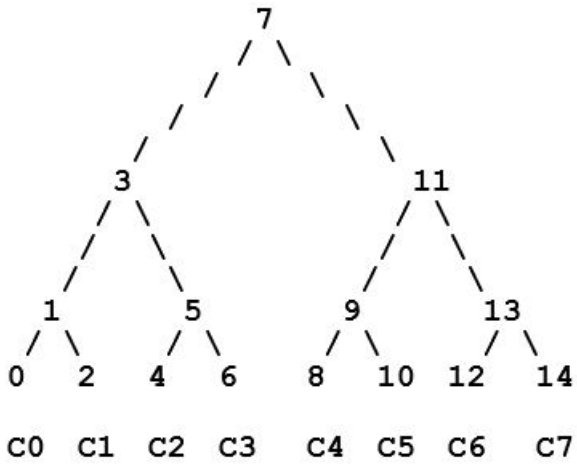


Figure 3-1: bin number tree

bin 7 represents the entire interval, bin 3 represents the chunk interval c0,c1,c2,c3 whereas bin 1 consists of only two interval of chunks c0 and c1.

A single bin representation of 32/64 bit is given as follows:

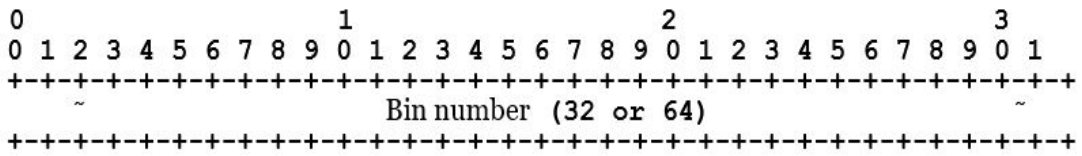
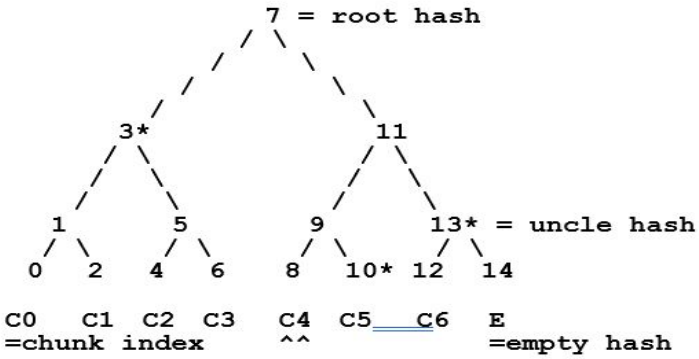


Figure 3-2: bin representation 32/64 bit

3.6 Content Integrity Protection and Verification

There are different methods available for the protection of content integrity during the distribution of content in the Peer to Peer network. Content integrity protection can be maintained over a fair environment conditions.

Merkle hash Tree Scheme is one of the method introduced for the protection of static contents integrity. This scheme must be used if the PPSPP operates over the internet and also it may be available to use in the fair environmental conditions. It usually operates with help of the different chunk addressing schemes. And it depends on the capability to address the range of chunks.



Merkle hash tree of a content file with N=7 chunks

Figure 3-3: Merkle hash tree

Merkle hash Tree scheme uses self-certifying hash tree which permits each peer to notice the distribution of fake content by a non-trusted peer. It makes sure that only a small amount of information is required to initiate the download. The division of Merkle hash tree into N chunks can be illustrated by an example. In the following figure an example of division into 7 chunks is shown. The structure does not assume chunks of content to be of a fixed size. The hashes of all the chunks of the content are calculated with a given cryptographic hash function (MDC [HAC01]). The leaves of the tree correspond to a chunk are assigned the hash of that chunk, starting at the leftmost leaf. Remaining leaves in the tree are assigned an empty hash value of all zeros. The hash values of the higher levels in the tree are calculated, by interconnecting the hash values of the two children (again left to right) and computing the hash of that aggregate. The process finally ends in a hash value for the root node called the “root hash” which only depends on the content.

The procedure for the Content Integrity verification starts when a peer receives the root hash of the content, it first calculates the hash of the chunk it received. Using the hash tree information, the peer then recalculates the root hash of the tree and compares it to the root hash it received from the trusted source. Once the match found, the chunks of content verified to be the requested part of the content. On the other hand, the sending

peer sent either the wrong content or the wrong sibling or uncle hashes. For simplicity, the set of sibling and uncle hashes is collectively referred to as the "uncle hashes". For live streaming, the tree of chunks grows dynamically and the root hash is indefinite.

For both the live and predefined content, Atomic Datagram Principle is introduced in which it deals with the verification of chunks received by receiver is carried out. Here, the receiver must recollect all the chunks it is omitted but still wants to download for the purpose of verification.

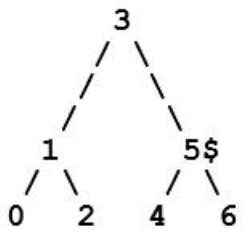
3.7 Live Streaming

For live streaming, PPSPP supports two methods for a peer to verify the content it collects from another peer, called "Sign All" and "Unified Merkle Tree".

In the "Sign All" method, the live injector signs each chunk of content using a private key. When the chunk is received, peers check the signature using the matching public key obtained from a trusted source. A peer that needs to send a chunk of content builds a datagram that MUST have a SIGNED_INTEGRITY message with the chunk's signature, followed by a DATA message with the actual chunk. If the SIGNED_INTEGRITY message and DATA message cannot be confined into a single datagram, the SIGNED_INTEGRITY message MUST be sent first in a different datagram.

On the other hand in the "Unified Merkle Tree" method, PPSPP combines the Merkle Hash Tree scheme for static content with signatures to unify the video-on- demand and live streaming scenarios. In this method, the chunks of content are used as the basis for a Merkle hash tree as for static content. However, because chunks are continuously generated, this tree is dynamic and the tree does not have a root hash. Hence, a public key serves as swarm ID of the content.

The updates in the tree is carried out by the process: Signed Munro Hashes, in which it allows to create a number of chunks which are added as new leaves to the current hash tree.



Extended version of a live tree is given above. Node 5 with the $ sign is considered as the new munro with the new subtree spanning 4 and 6. In the integrity verification process, if the system has peers that keep the whole broadcast and provides the content available when the broadcast ends using a stabilized tree and the final root hash as swarm identifier.

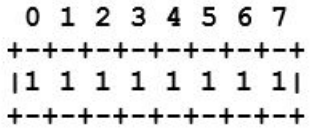
3.8 Protocol Options

The different protocol options available in PPSPP is given in the following table. 8-bit code followed by an 8-bit value represents the protocol option. Here the protocol options are sorted on code value in a HANDSHAKE message which are in ascending order.

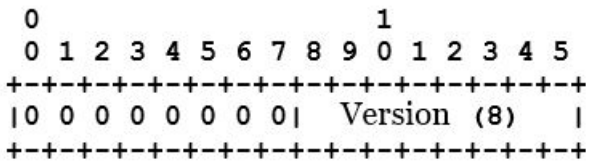
Code	Description
0	Version
1	Minimum Version
2	Swarm Identifier
3	Content Integrity Protection Method
4	Merkle Hash Tree Function
5	Live Signature Algorithm
6	Chunk Addressing Method
7	Live Discard Window
8	Supported Messages
9	Chunk Size
10-254	Unassigned
255	End Option

Table 2: PPSPP options

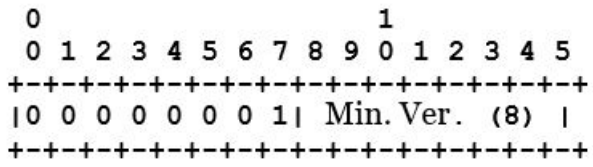
End option is the 1 octet length protocol option in which successive octets are considered as protocol messages. The code of the End option is given as 255 and the representation is given below:



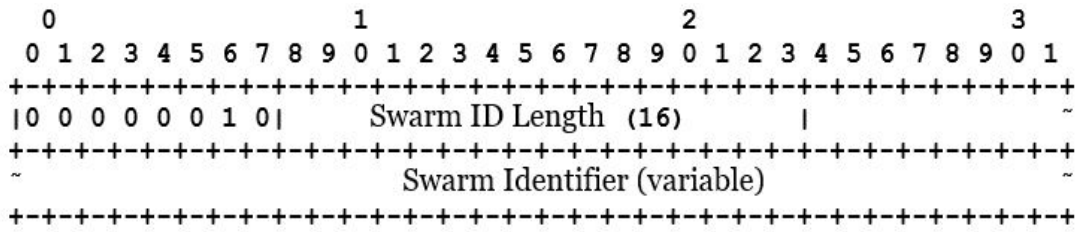
Version is the first protocol option in the list in which a peer possess the maximum version of the PPSPP. The code is given by 0.



The code for the Minimum version protocol is 1. And when the peer initiates the communication containing the HANDSHAKE message, the minimum version PPSPP must be included.

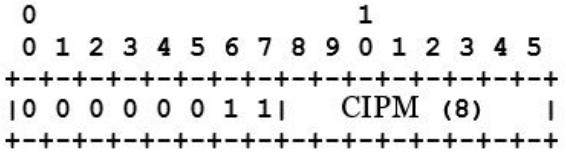


A swarm identifier option is included when a peer initiates the communication. The structure following represents the Swarm identifier protocol option.

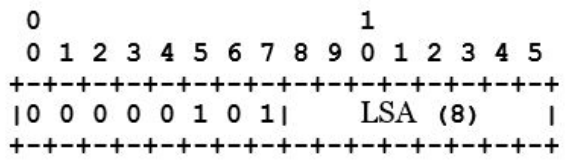


The Swarm ID Length field contains the length of the single Swarm Identifier that follows in bytes. The Length field is 16 bits wide to allow for large public keys as identifiers in live streaming. Each PPSPP peer knows the IDs of the swarms it joins, so this information can be immediately verified upon receipt.

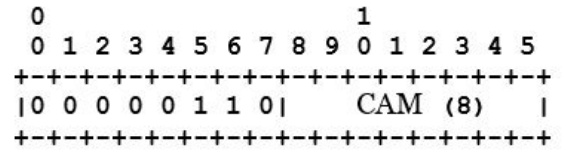
The code for the content integrity protection method is 3. For the static content Merkle hash tree is used as default. Sign All and Unified Merkle tree are the two methods of Merkle hash tree scheme that are suitable for the live contents.



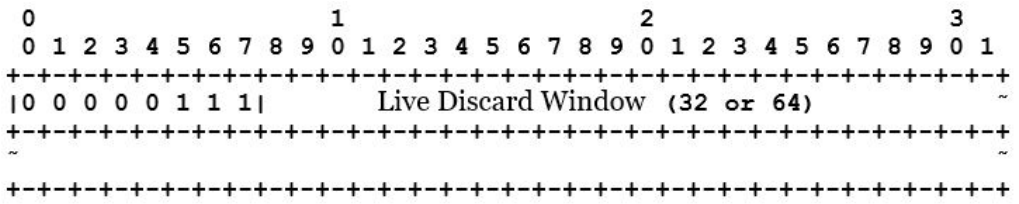
Live signature Algorithm must be defined if the Sign All or Unified Merkle tree are used in the content integrity protection method. This 8 bit protocol option is included in the “Domain Name Security (DNSSEC) Algorithm Numbers” registry.



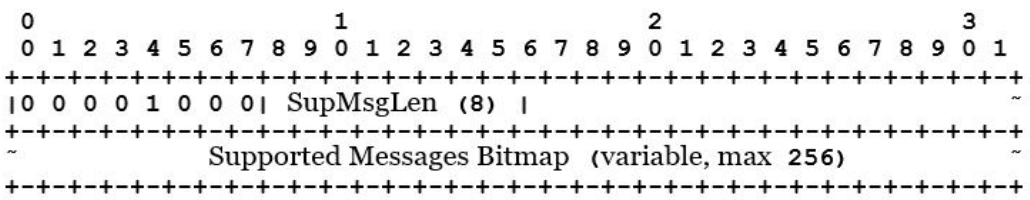
Chunk addressing method protocol option with the code 6 supports the 32/64 bit chunk ranges. 32 bit is the default one.



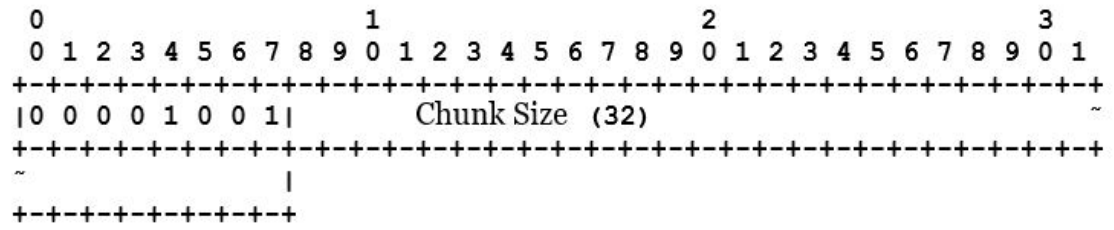
Live Discard window with the code 7 contingent on the type of Chunk addressing method used. The value is 32/64 bit and the method must appear on the list before using this protocol. option.



Supported messages are 256 bits in length supported in the PPSPP. In order to build a bitmap the values given are 1 for each type supported and 0 for the non-supported message types.



Chunk size is a 32-bit integer representing the size of the chunk in which it is used by the Swarm. When the size varies it must be set to a special value 0xFFFFFFFF.



3.9 UDP Encapsulation

UDP as the transport protocol and LEDBAT for the congestion control are considered for the implementation of PPSPP. LEDBAT which supports and provides the non-interrupted content when the playback option is introduced for the content.

Chunk sizes are one of the dependent factor of the protocol. As the size of the chunk gets decreased which leads to affect the efficiency of the transport protocol.

The table below represents the message types in PPSPP:

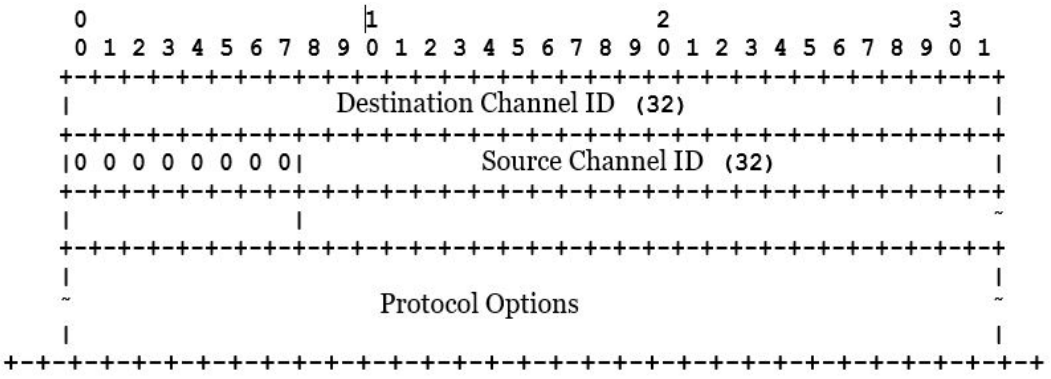
Msg Type	Description
0	HANDSHAKE
1	DATA
2	ACK
3	HAVE
4	INTEGRITY
5	PEX_RESv4
6	PEX_REQ
7	SIGNED_INTEGRITY
8	REQUEST
9	CANCEL
10	CHOKE
11	UNCHOKE
12	PEX_RESv6
13	PEX_REScert
14-254	Unassigned
255	Reserved

Table 3: Message types

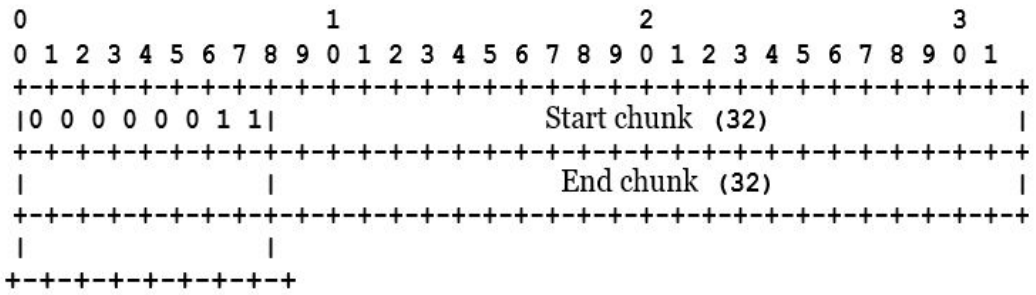
These unchanged messages such as DATA, ACK, HAVE, INTEGRITY, PEX, SIGNED_INTEGRITY, REQUEST, CANCEL, CHOKE and UNCHOKE messages are capable of resending without the interruption when a loss in the information is

suspected. In the case of HANDSHAKE message, a duplication can be accepted due to the documented version of HANDSHAKE message in the initial attempt.

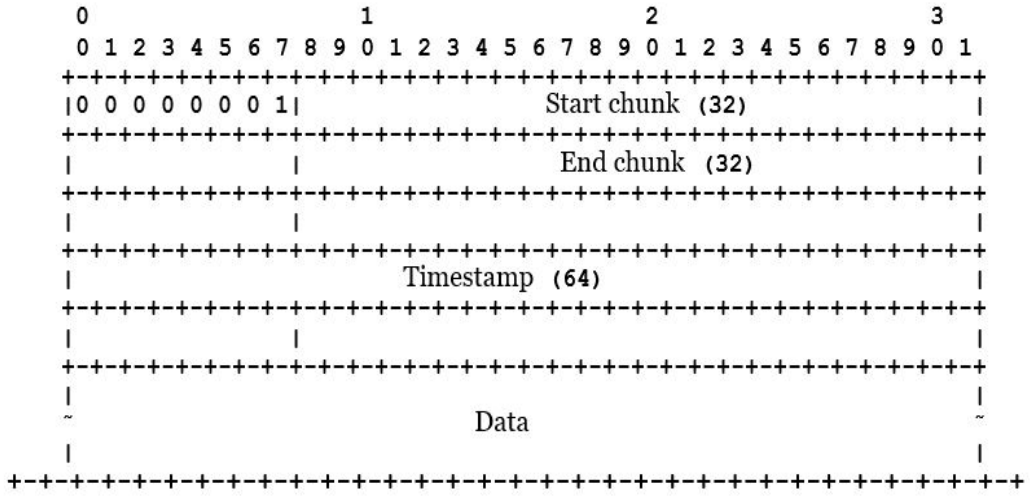
Channels are the multiplexing scheme which permits the swarm to use the same UDP port. In the UDP encapsulation Peers acquire the channel ID information each other during the HANDSHAKE procedure. A datagram having HANDSHAKE message with the channel ID is given below:



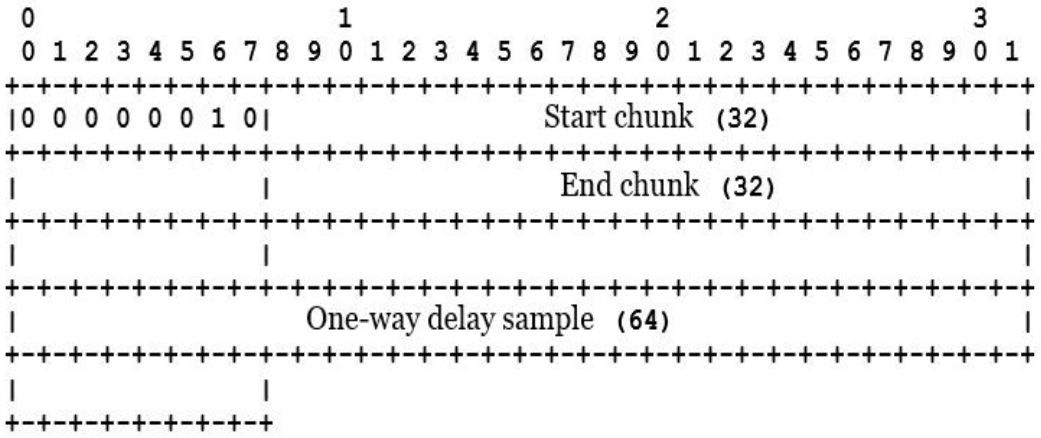
Chunk specification are defined in the HAVE message. And it contains the information of the positively verified range of chunks. A HAVE message with 32- bit chunk ranges as follows:



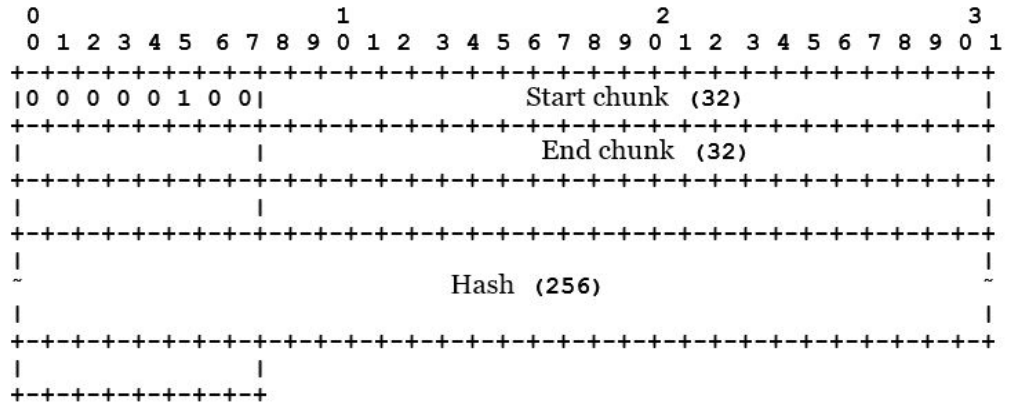
A DATA message contains the details of a timestamp and the actual chunk. And it can found at the tail of the datagram. By the usage of LEDBAT congestion control a timestamp with 64-bit integer is included by the sender. A DATA message with 32-bit ranges of chunk as follows:



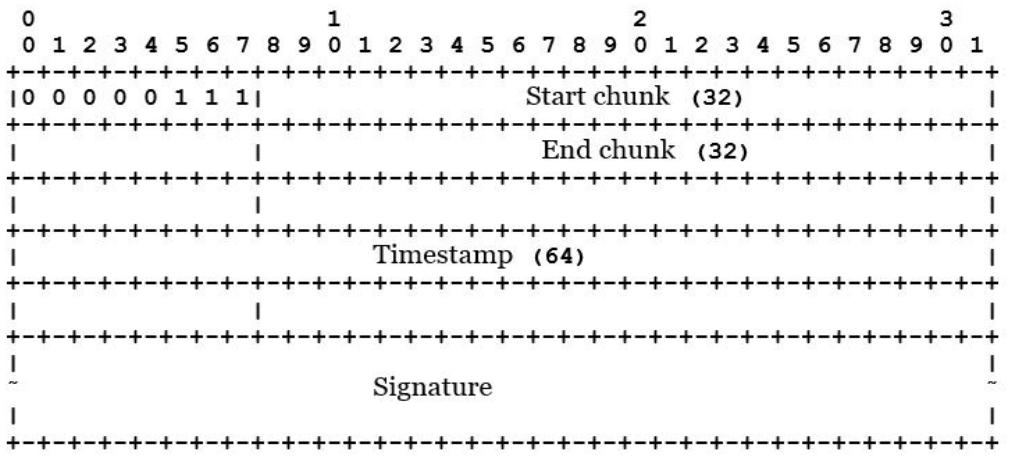
ACK message acknowledges the data received from the recipient in order to comply with the LEDBAT delay-based congestion control. It contains a chunk specification and a timestamp demonstrating one-way delay sample obtained from the previous DATA message. A ACK message with 32-bit ranges of chunk as follows:



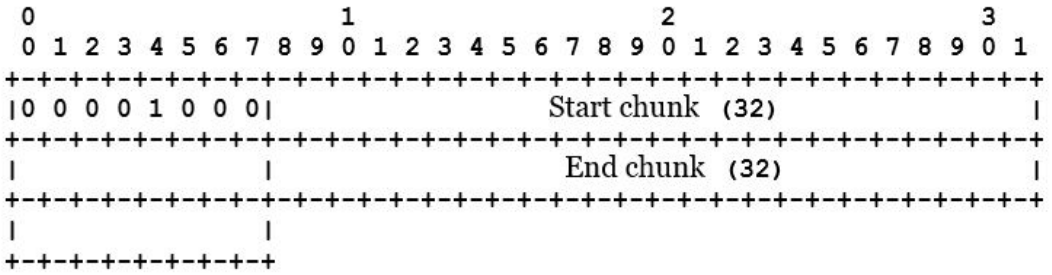
An INTEGRITY message contains the chunk specification and the cryptographic hash of the indicated chunk. And it depends on the protocol options. An INTEGRITY message with 32-bit ranges of chunk as follows:



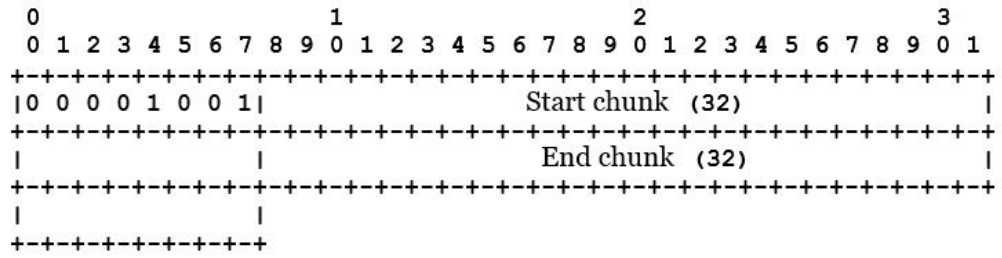
SIGNED_INTEGRITY message contains a chunk specification, a 64-bit timestamp in NTP format and a signature field with digital signature encoded. Live signature Algorithm represents the signature algorithm in which it relies on the content integrity protection. A SIGNED_INTEGRITY message with 32-bit chunk ranges as follows:



A REQUEST message contains the chunk specification for the chunks in which the client wants to download. A REQUEST message with 32-bit ranges of chunk as follows:

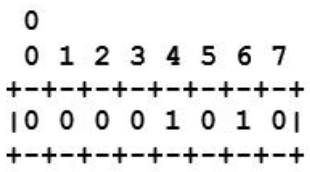


A CANCEL message contains the chunk specification for the chunks in which the client no longer wants to initiate the operation. A CANCEL message with 32-bit ranges of chunk as follows:

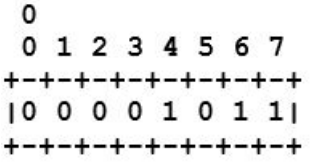


CHOKE and UNCHOKE messages represents the willingness to respond to the REQUEST message from one Peer to another one. It doesn't carry payload.

CHOKE:



UNCHOKE



A PEX_REQ message doesn't carry payload where as a PEX_RESv4 contains a IPv4 address and a PEX_RESv6 with IPv6 address. A PEX_REScert message contains 16-bit integer stipulating the membership certificate states that Peer p at time T is an affiliate of Swarm S.

KEEPALIVE does not contain a message type on UDP protocol. It only consists of the channel ID of the target.

3.9.1 Flow and Congestion Control

In case of video on demand, receiver demands for the content from the peers which will be in control of how much information is reaching towards it. And in the live streaming, a push model is modified. Here the incoming information is limited in which the receiver

will be able to provide a non-interrupted playback option for the content. PPSPP can support several congestion control algorithms. LEDBAT congestion control algorithm is used currently which is based on delay congestion control algorithm commonly used nowadays as transmission protocol for BitTorrent.

LEDBAT observes the delay on the information path which is used in the one way delay variations and capable of limiting the congestion in the network. PPSPP are mainly focused on distributing the content over the peers which are interested in without distracting the user even if the playback is finished. In the case of congestion near the sender, which mainly effects in the transmission of chunks to the receiver. And if the bottleneck is near the receiver, the transmissions through this bottleneck will withdraw quickly due to LEDBAT. In peer to peer context, LEDBAT is the algorithm in which it mostly works for many scenarios.

3.10 Manageability Considerations

Operations and management considerations are discussed here in this section:

PPSPP client and server is defined here, where the client part may not have a copy of the content but the server part provides the copies of the content to the swarm.

A Content provider wishes to distribute the content is done by setting up at least a PPSPP server. These servers are required to possess accessibility to the static or live contents. The output of this configuration process will have a list of metadata records each for a swarm. A metadata record contains swarm ID, chunk size used, the chunk addressing method used, Content integrity protection method used and if applicable the used Merkle hash tree function. A content provider usually sets up a tracking facility for the content either by a protocol tracker or Distributed hash table. And it provides the transport addresses for the purpose of tracking. Metadata records and the transport addresses are distributed to users in various ways, typically as website links.

Peer to Peer protocol influences the content which is accessible from different sources which improves the robustness, scalability and performance. PPSPP advantages from the ALTO protocol in order to navigate the peer selection. All the peers receives the content on time when PPSPP is operating properly. Client serves the perfect location in order to verify the protocol’s correct operation. It is not viable to deliver the behavior of peers in all operations due to privacy reasons. There are two other options available: one by monitoring the PPSPP servers and by the tracker protocol which helps to acquire the information of all peers in a swarm. When the PPSPP download starts the packet

inspection for DATA and ACK messages are used to provide the transfer of content, signaling the availability of chunk and the integrity check of content also works.

Management considerations are alike to the protocols which are used for the dissemination of large scale content especially in HTTP. Servers giving the copies of the content are similar to WWW and FTP servers. It can also be arranged and advantages from the standard management facilities. Facilities for identifying the positive operation and server management are said to be satisfactory for fault monitoring. It is accompanied with the host resource and UDP/IP monitoring. When a faulty peer starts distributing wrong chunks, the content integrity protection mechanism detects it and a different source is required to proceed further. Changes can be made when considering the configuration management. By uploading new content which makes it available for downloading.

Content providers are capable of providing PPSPP hosting for the clients and the billing of these clients are performed based on the usage of bandwidth. It is the common scenario for accounting which is similar to the billing of servers for the web servers. The tracker protocol used in PPSPP offers the built-in, application level performance measurement structure. The non- trusty peers are usually clogged out for long term due to the performance reasons.

3.11 Security Considerations

Considering the different network protocols, PPSPP deals with common set of security challenges. For the application protocol reflects on the possibility of excess buffer, DoS attacks and management. Here the assurance of privacy appears to be doubtful because the user reveals the IP address to other peers. And the security considerations of the protocol is mainly discussed here:

Mainly 3 types of attacks over PPSPP are analyzed here: DoS amplification attack in which the attackers use a peer to create more traffic, DoS flood attack in which attackers denies the service to other peers and Disrupt service to an individual peer in which attackers provides the fake messages.

These attacks are protected by the usage of secure handshake procedures with arbitrarily chosen channel ID’s. For the illustration two peers are considered Peer A and B in which Peer A initiates the interaction with Peer B by sending the datagram having the HANDSHAKE message with channel ID’s. These channel ID’s are given below:

A->B: chan 0 + HANDSHAKE (chanA) + …

Here Peer B receives the datagram with channel ID chanA and the response to Peer A containing a HANDSHAKE message with the same ID. A return routability check is introduced before sending the payload. Hence this attack is protected.

B->A: chanA + HANDSHAKE (chanB) +...

Here in the next datagram Peer A sends a chunk to peer B with channel ID chanB. Both peers are capable of interacting each other. So the 3 way handshake is finished. This 3 way handshake is slightly vulnerable when compared to the stream control transmission protocol (SCTP) which responds quickly for the user.

A->B: chanB + HAVE + DATA + ...

Here the attacker would fake the REQUEST, HAVE messages from Peer A to Peer B in order to guide DATA messages sending to Peer A or avoids Peer B from sending them. Protection managed by channel ID's are enough in most scenarios. Also point to point encryption of traffic is considered in the other cases.

PEX messages exchanged between two peers contains IP address and port of other peers. It permits the decentralized tracking with no central tracker is required. The susceptibility of this mechanism allows the fault peers to use it for Amplification attack. Further PEX used in Eclipse attacks where the fault peers detach a distinct peer because it only communicates with faulty peers. Two faulty peer attacks are discussed here: One in the case of live streaming in which the faulty peers tries to eclipse the injector and the other one in which the faulty peers tries to eclipse a customer peer.

Protection against amplification attack contributed with the help of public key cryptography and Swarm membership certification. The certification correlates to the signed node descriptors in secure decentralized peer sampling services. There are different designs for the membership certificates. For the description a model where PPSPP tracker acts as certification authority is introduced. When considering the protection against Eclipse attacks, faulty peers register themselves at the tracker and percentage of bad peers become high which residues the protection of the tracker to PPSPP tracker specification. If there is 50% faulty peers exists, it means that 50% good addresses are available from the tracker.

Confidentiality of a content publisher relies in the distribution of content in cipher text or in a pattern in which Digital Rights Management (DRM) approach have been practiced.

A key management or point to point encryption is favorable for the proposed mechanism. i.e., closed swarm access control.

Further the strength of the Hash function for Merkle Hash Trees are discussed. Hash function used makes it more difficult to generate collisions. Attackers find a collision and just substitute it with a chunk. With the usage of fixed size chunks the collision has to be the concatenation of 2 hashes. In sum, it is difficult to find collision that suits the hash tree.

Analysis of the potential damage caused by the faulty peer in each message types such as HANDSHAKE, HAVE, DATA and ACK varies with the prevention methods introduced in the protocol.

A receiving peer detects faulty senders which excludes a bad peer from the system and is complex. Random monitoring by trusted peers banish the faulty peers which is the only option available in this scenario.

4 CONCLUSION

In the Section Overall Operation of PPSPP RFC 7574 presents the protocol with three examples: first one a peer joins a swarm then the exchange of data and the peer leaving a swarm. Once our software has data, it can upload. It will in turn send HAVE messages to peers, and respond to REQUEST requests. The "clean" start of the swarm is done by sending a HANDSHAKE message (and by separating from the tracker, if there is one). But, of course, in a peer-to-peer network, machines often do not leave cleanly: network shutdown, crash of machine or software, and so on. Peers must be prepared to handle the case of unresponsive peers.

In the Section Messages of PPSPP RFC7574 presents all the messages that can be exchanged (twelve types). The messages do not require an answer: a lack of information from the peer indicates that there is a problem. For example, if a HANDSHAKE message is send to peers which are not interested in receiving the message, here the peers will not respond with an error message but they will simply refrain from answering. So there are two kinds of peers: one of those respond quickly and the other respond little or not, and so we will gradually decide to ignore. We do not insist on nonresponding peers.

Note that PPSPP ignores the format of the data sent: it transmits bytes, without knowing if it is MPEG, or a container gathering several files, as allow formats like AVI. PPSPP is a data transfer protocol, not a complete tele-broadcast solution.

Points to be noted in the messages: each swarm transmits a stream of bytes. A swarm has an identifier (swarm ID), exchanged in HANDSHAKE messages. These messages also contain other parameters such as the size of chunks or the exact function used for the Merkle tree. The stream of bytes is indeed cut into pieces, each having an identifier (chunk ID). This identifier is a parameter of HAVE, REQUEST and DATA messages. Thus, a DATA message contains a piece, and the identifier of that piece. This allows the peer to know where he is and if he is missing something.

INTEGRITY message which contains a cryptographic condensate of a subtree of the Merkle tree corresponding to the transmitted content. INTEGRITY messages are not signed so they only protect against accidents, not against a malicious peer. There are SIGNED_INTEGRITY messages which are signed. Note that, in this case, the public key is in the swarm identifier, encoded according to the DNSKEY format of DNSSEC.

The REQUEST messages mentioned above are used to request a portion of the data stream. Unlike BitTorrent, they are not essential: a peer can send you songs that you have not explicitly requested. This is logical for PPSPP, which is intended for runoff: PPSPP is more push than pull.

In the Section Chunk Addressing Scheme explains the addressing mechanism of the pieces. We do not have to designate them one by one. Intervals ("from song N1 to song N2"), bytes ("all the pieces that make up the first million bytes") or bin numbers can be used. A bin number is a very clever mechanism for designating an almost arbitrary byte interval with a single number.

In the Section Management Considerations of RFC7574 will be of particular interest to system and network administrators. It covers operational issues with PPSPP. For example, as with any peer-to-peer protocol, a peer PPSPP has an interest in choosing "close" peers (in quotation marks because it is not easy to define "close"). The RFC 7285 ALTO protocol can help you better choose your peers. But it must be said that, the concrete experience with PPSPP still missing, we are a little here in unexplored areas.

In the Section Security Considerations of RFC7574 focuses on security issues. First, privacy: Like BitTorrent, PPSPP exposes to each peer the IP address of other peers and we cannot hide from his peers and his interest in this or that content. Note that PPSPP does not provide a mechanism to protect the confidentiality of the content. If you want to distribute restricted content, you encrypt it before, or use IPsec or DTLS.

Then, the risk of reflection attacks, since PPSPP will typically run on UDP. If an attacker lies on his IP address, it triggers the sending of massive data to an innocent. A priority is that the use of a mechanism analogous to TCP's Initial Sequence Number (ISN). The communication requires the knowledge of numbers called channel ID and randomly chosen (RFC 4960). An attacker cannot know them if the user has cheated on the IP address (since user will not receive the message communicating the channel ID to use: in other words, PPSPP tests "Routability").

Peer5 operates as a service that orchestrates peers and provides analytics and control. The service is triggered using JavaScript API in the browser. The client side contains custom WebRTC p2p logic, an HTTP client and a delivery manager that improves speed using the two methods (P2P and HTTP). The client connects to multiple peers and to the HTTP server simultaneously and ensures chunks are received in time for playbacks.

5 REFERENCES

[1] A. Bakker, 2015, Peer to Peer streaming Peer Protocol, Vrije Universiteit, Amsterdam

[2] Merkle, R., 1979, "Secrecy, Authentication, and Public Key Systems", Ph.D. thesis, Stanford University, CA, USA

Webpages

[1] Peer 5: The Server less CDN https://www.peer5.com/

[2] Peer to Peer : Wikipedia