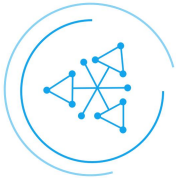
STREAMROOT

Device-side delivery intelligence:
Rethinking the modern video CDN with peer-to-peer delivery and client-side load balancing

Pierre-Louis Théron, Co-Founder & CEO, Streamroot

Mile High Video 2019

Streamroot device-side delivery solutions



STREAMROOT DNA

WebRTC-based
peer-to-peer CDN

COMPASS

Quality-based midstream
multi-CDN selector

DNA ENTERPRISE

Flexible eCDN for global
corporate streaming

STREAMROOT

About Streamroot...

The largest and most trusted device-side delivery provider for OTT video

vimeo TF1 dailymotion CANAL+ EUROSPORT

rtve france•tv VTV M6 WEB mediastream RT

- CONMEBOL - COPA AMERICA BRASIL 2019

FIFA WORLD CUP RUSSIA 2018

Le de TOUR France

australian open

STREAMROOT

About Streamroot...

The largest and most trusted device-side delivery provider for OTT video

vimeo TF1 dailymotion CANAL+ EUROSPORT

rtve france•tv VTV M6 WEB mediastream RT

+20 Million

UNIQUE VIDEO SESSIONS DAILY

STREAMROOT

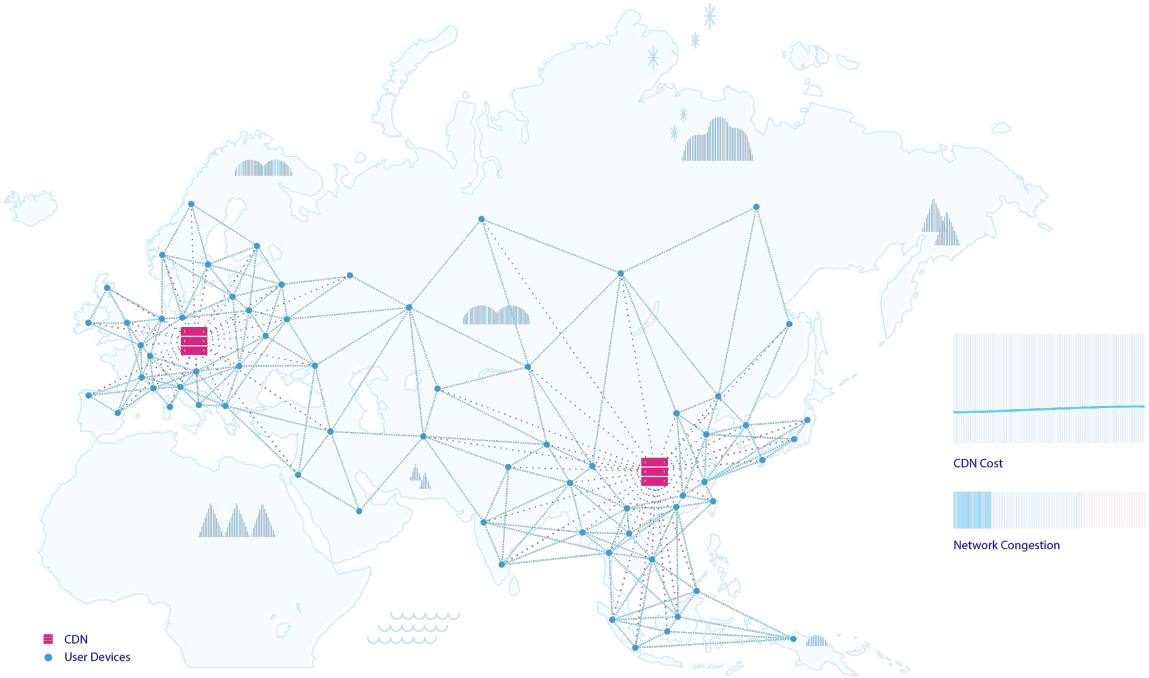
A word on P2P delivery and WebRTC

Dynamic & smart multi-sourcing and matching

Secure & effective delivery via WebRTC

Micro-caching

Intelligent assembling of the segments



Why WebRTC makes the difference

- ✓ **No installation required -** totally transparent for end users
- ✓ Leveraging **open source** libraries
- ✓ Available across **multiple devices and platforms**
- ✓ Efficient protocol built on **Low latency communications**
- ✓ **Secured protocol** with end to end encryption of the data

Low latency or hybrid P2P streaming?

WebRTC

`getUserMedia`

Acquires the audio and video media by accessing a device's camera and microphone.

`RTCPeerConnection`

Enables communication between peers. It performs signal processing, codec handling, peer-to-peer communication, security, and bandwidth management.

Low latency live streaming

`RTCDataChannel`

Allows bidirectional communication of arbitrary data between peers with very low latency.

STREAMROOT

1 Why is an intelligence on the device side important?

Why have an intelligence on the device side?

1. DYNAMIC BEHAVIOR

Vision of the OS, network, etc. allow the device to react to changing conditions.

Example: **Bandwidth Drop**

Player ABR already handles this well…

But what if the solution was not to lower the bitrate, but to go to another CDN? Or get data from a peer?

2. FASTER REACTION TIME

The client is able to take a decision and act immediately. It's longer if the decision happens on a server.

Example: DNS switching a multi-CDN setup

Significant DNS TTL

Why have an intelligence on the device side?

3. MORE GRANULAR DECISIONS

Backend aggregates hide the outliers in averages

+ the client itself has the proper data to take action

→ Per user tailored data

2 How do we tailor delivery to each device?

We’re taking client-side logic a step further

P2P multi-sourcing and client-side multi-CDN

USE CASE

- Live vs. VOD
- Streaming format
- Latency
- Bitrate profiles

QUALITY OF SERVICE

- Buffer health
- Bitrate
- Track switches
- Dropped frames

CDN & P2P

- Throughput
- Error rate
- bandwidth per peer

We’re taking client-side logic a step further

P2P multi-sourcing and client-side multi-CDN

DEVICE

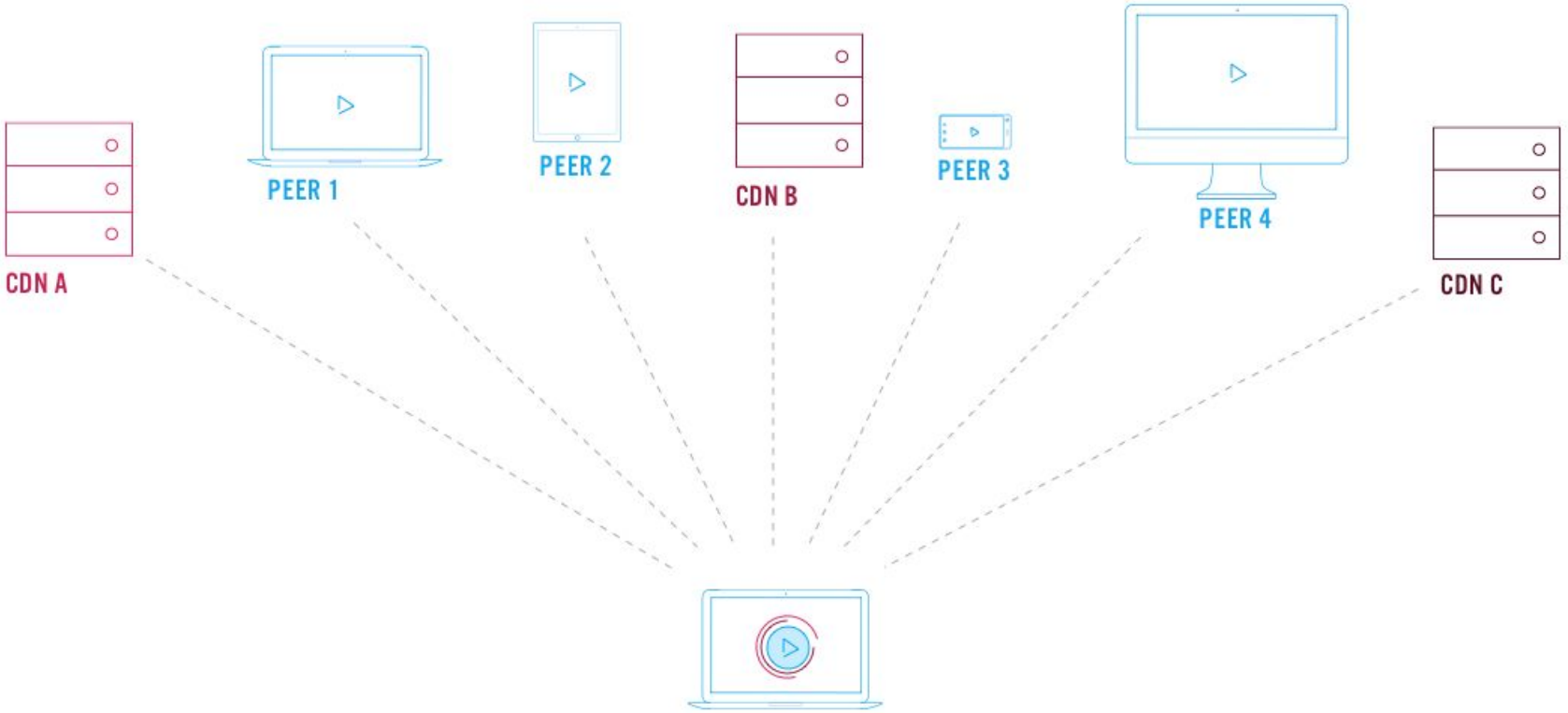
- CPU
- RAM
- Battery level
- Battery depletion rate

NETWORK

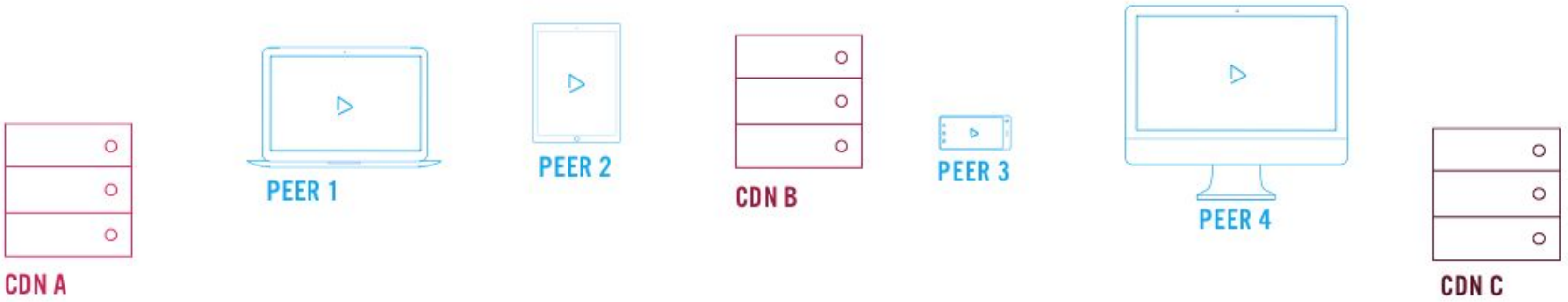
- ISP (data caps or not)
- Wifi
- Ethernet
- Cellular

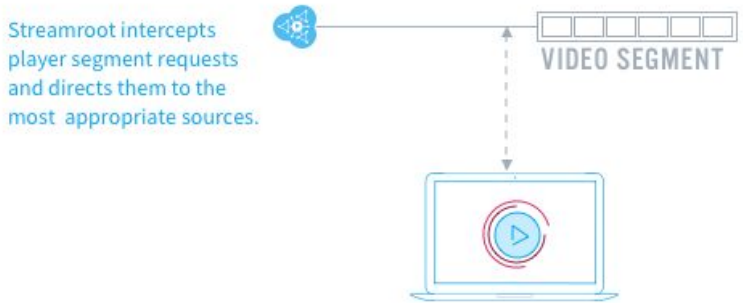
How it works

With Streamroot, content can come from multiple different sources.



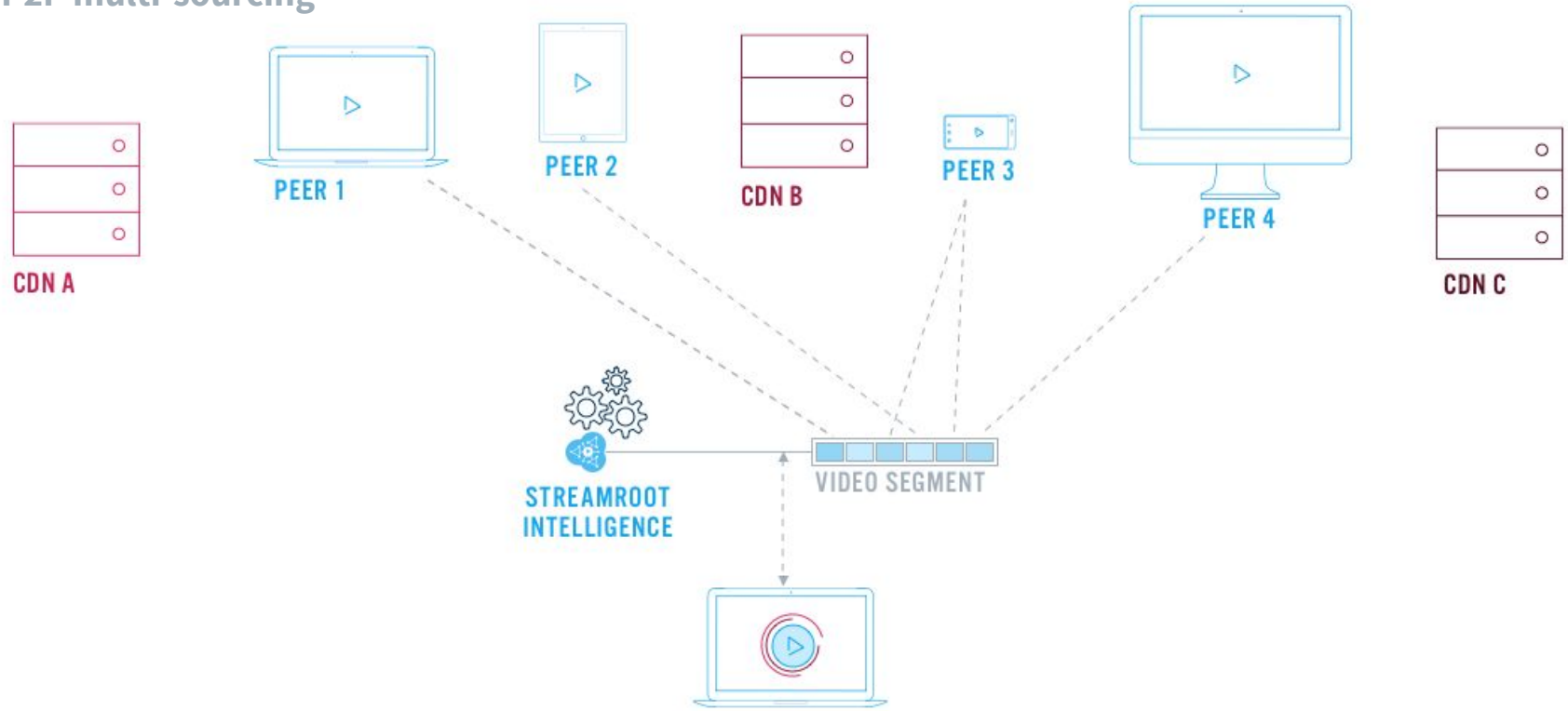
How it works



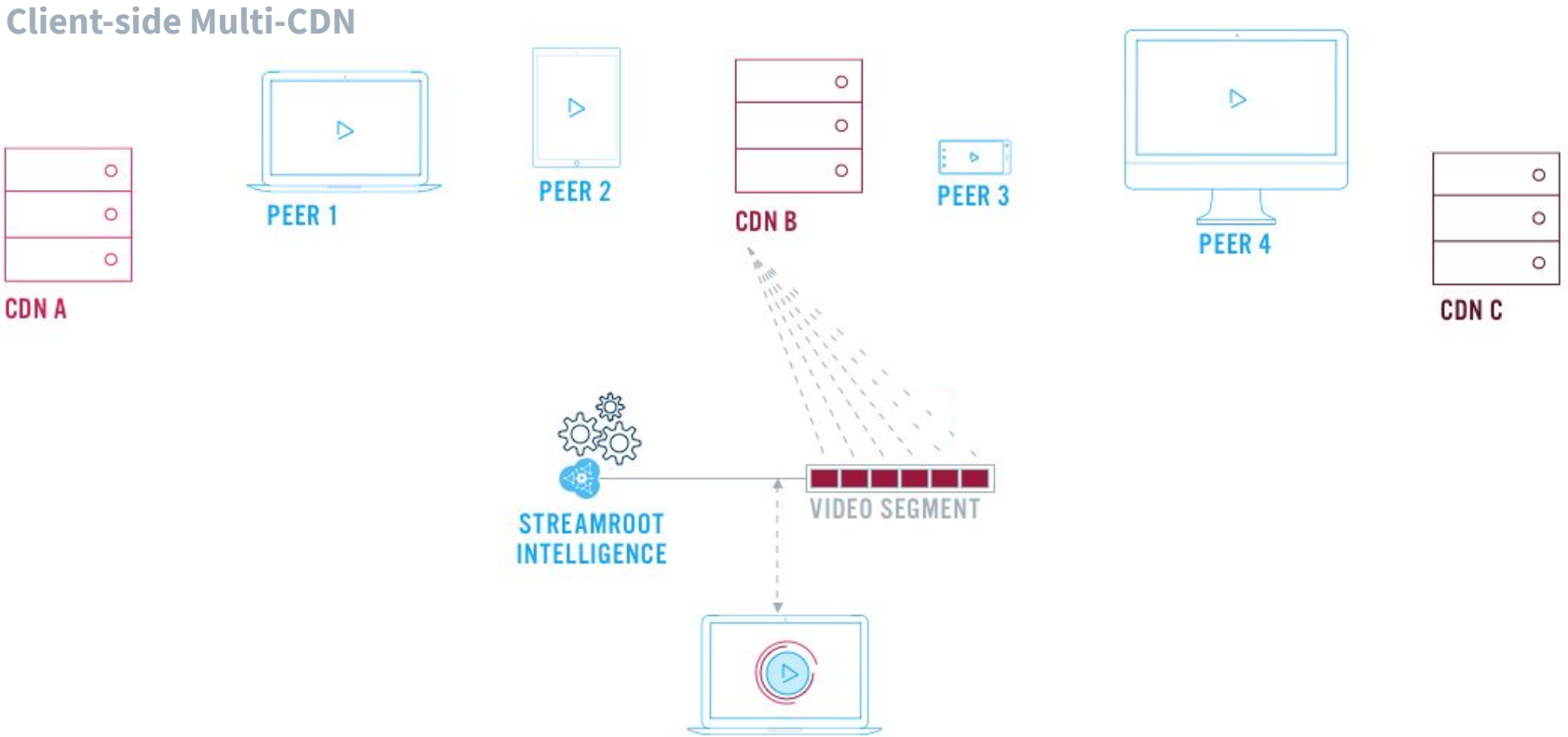


How it works

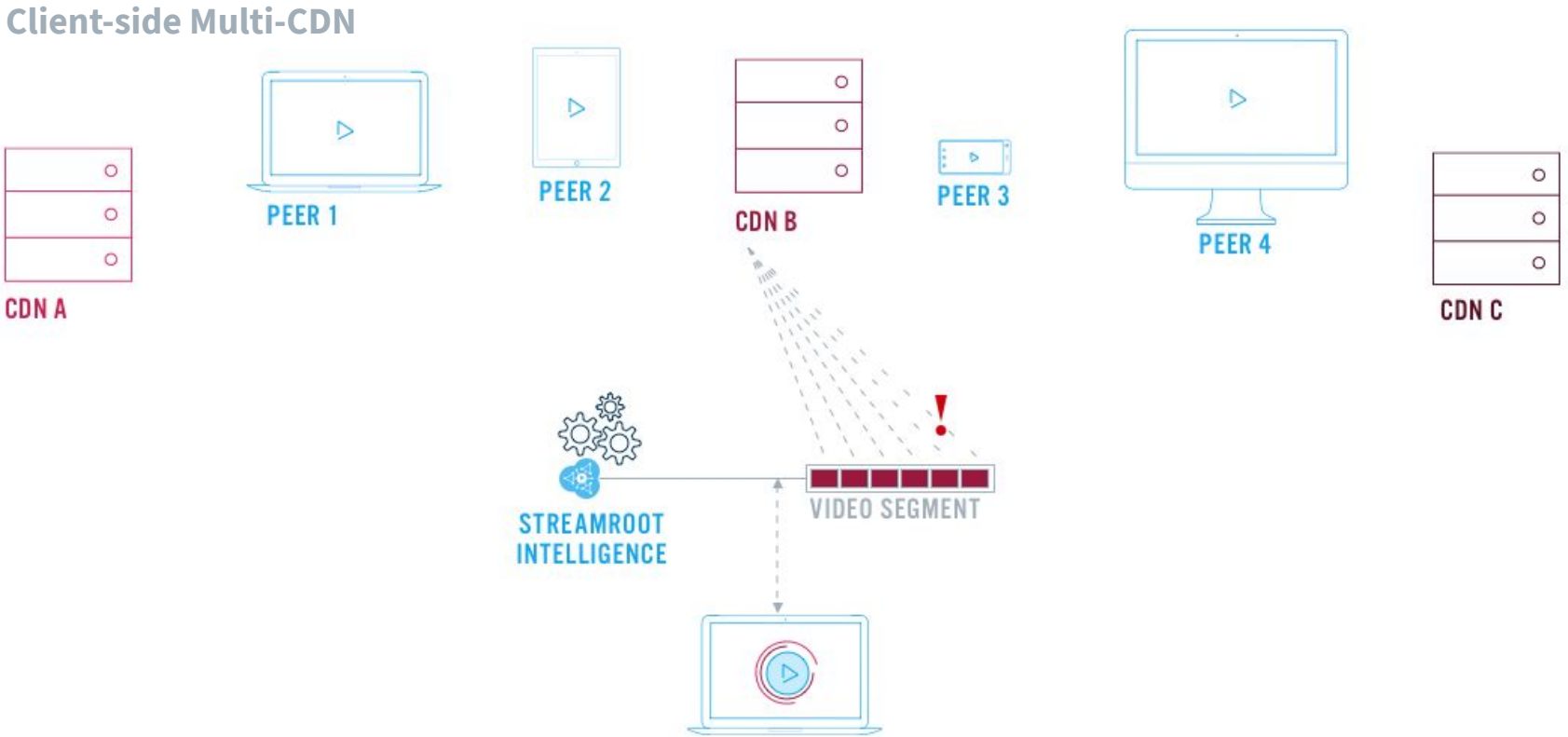
P2P multi-sourcing



How it works

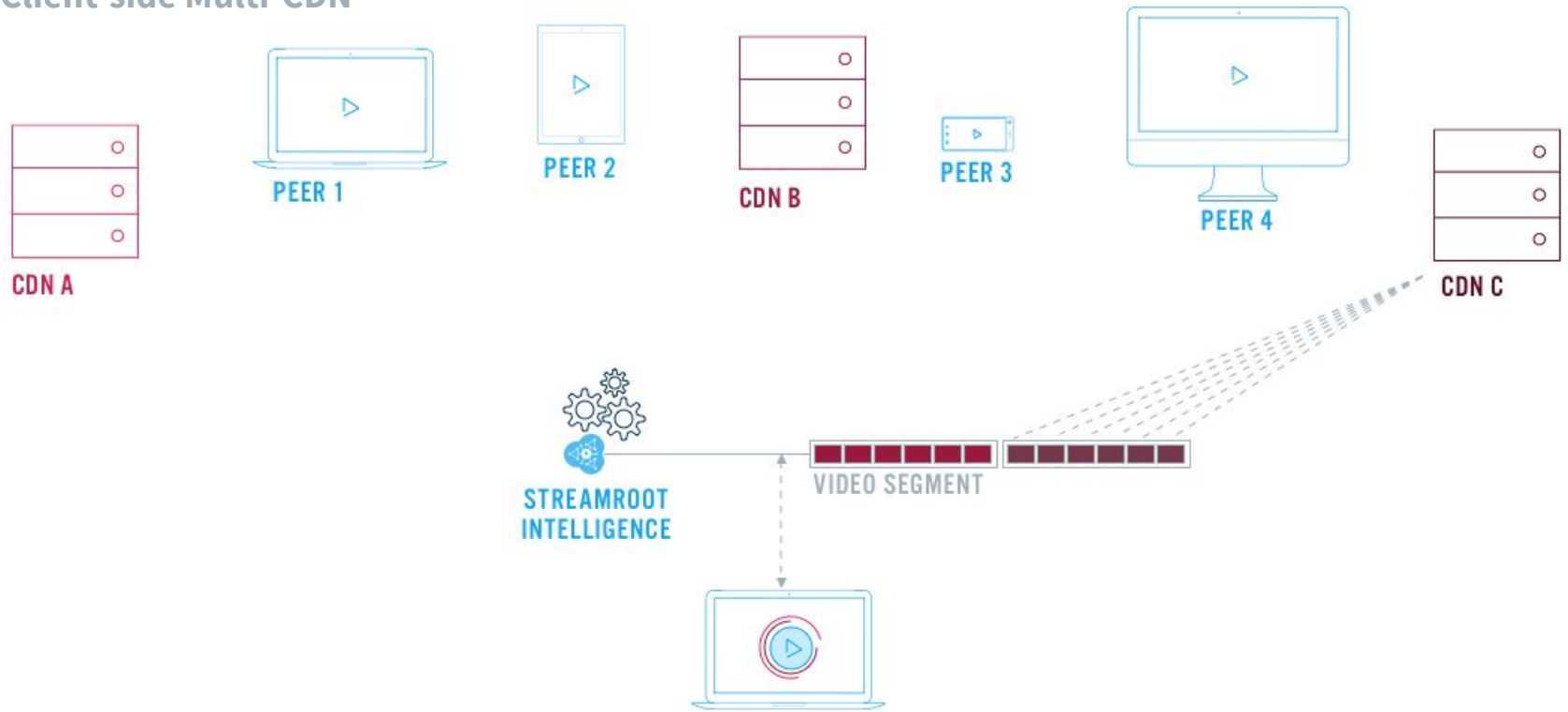


How it works



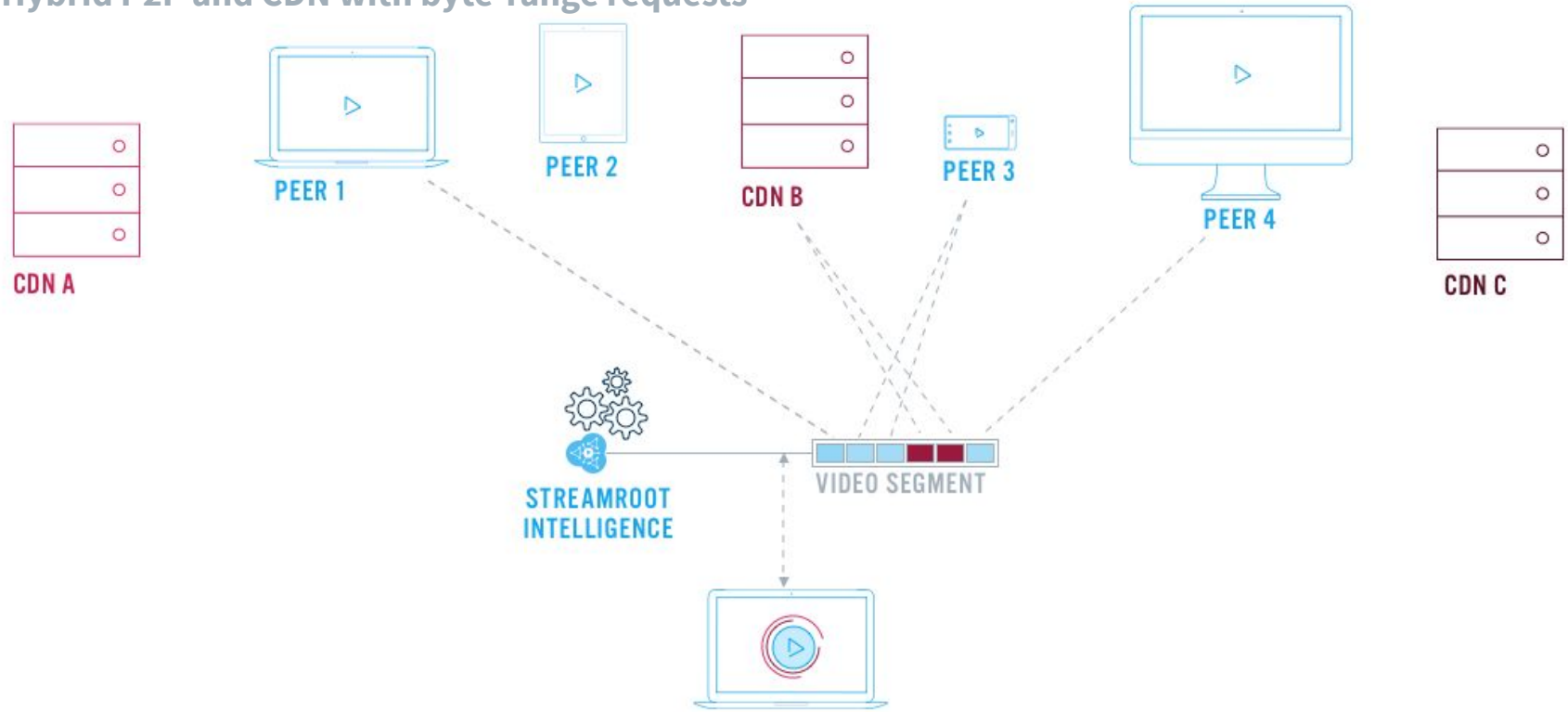
How it works

Client-side Multi-CDN



How it works

Hybrid P2P and CDN with byte-range requests



3 Examples of AB tests: peer connection numbers & upload

1. Experiment on number of peer connections

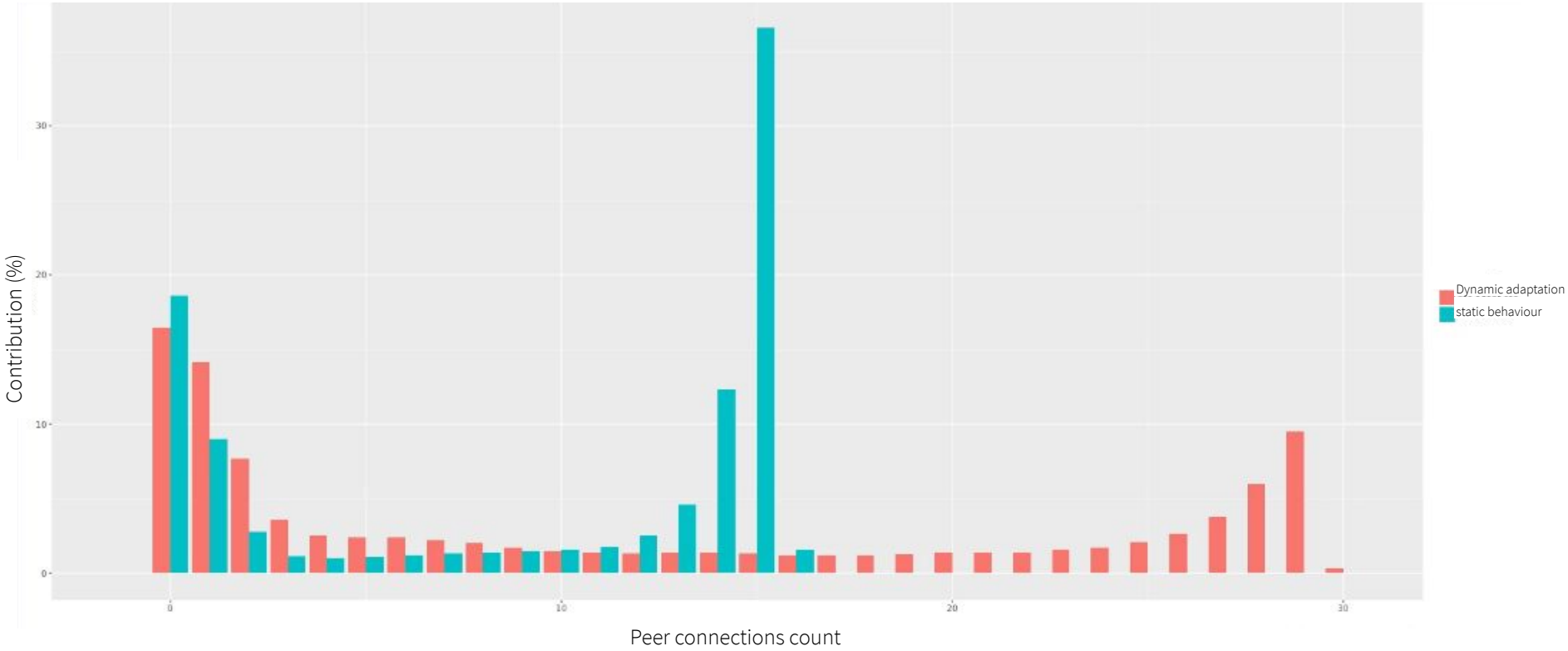
Number of peers = number of open WebRTC connections

Goal: contain CPU usage under a certain threshold while maximizing the number of productive peer connections

Before: conservative cap of 15 peers per device to protect legacy devices

After: Dynamic adaptation of number of connected peers based on CPU usage

Experiment results



Impact on quality and peer-to-peer efficiency

18%

fewer dropped frames

2.5%

less rebuffering

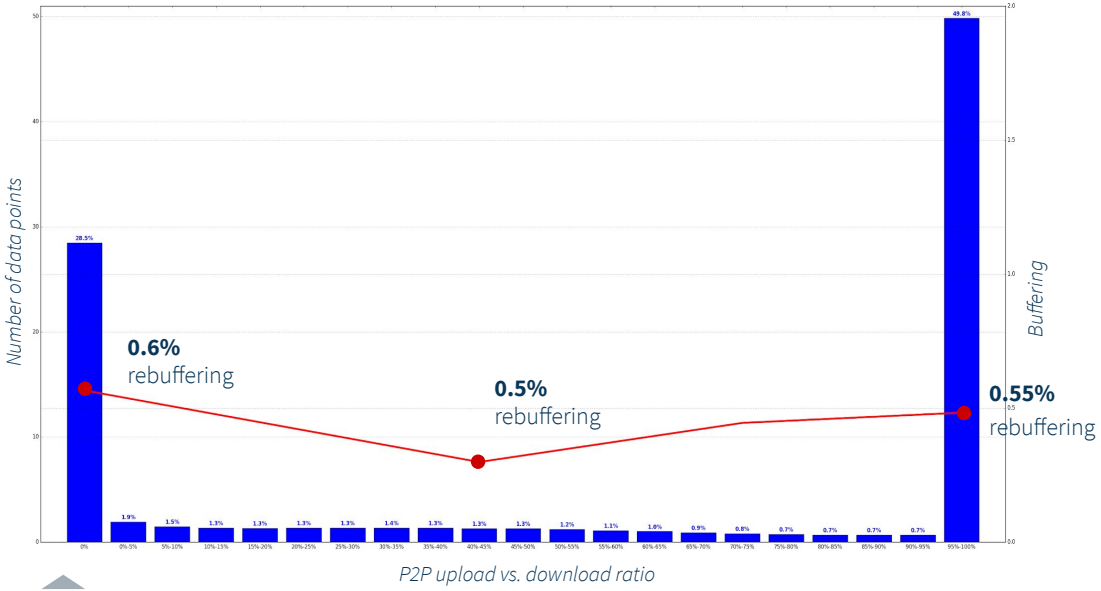
2Pts

more peer-to-peer exchanges

2. Effect of upload on QoS

Upload depends on devices, capacity, network conditions, etc. Majority of data in comes from a small number of powerful devices.

Result: rebuffering rates for top seeders is on par with those doing little or no upload.



In blue, different populations split according to the ratio of upload vs. download during their video session. In orange, rebuffering rates.

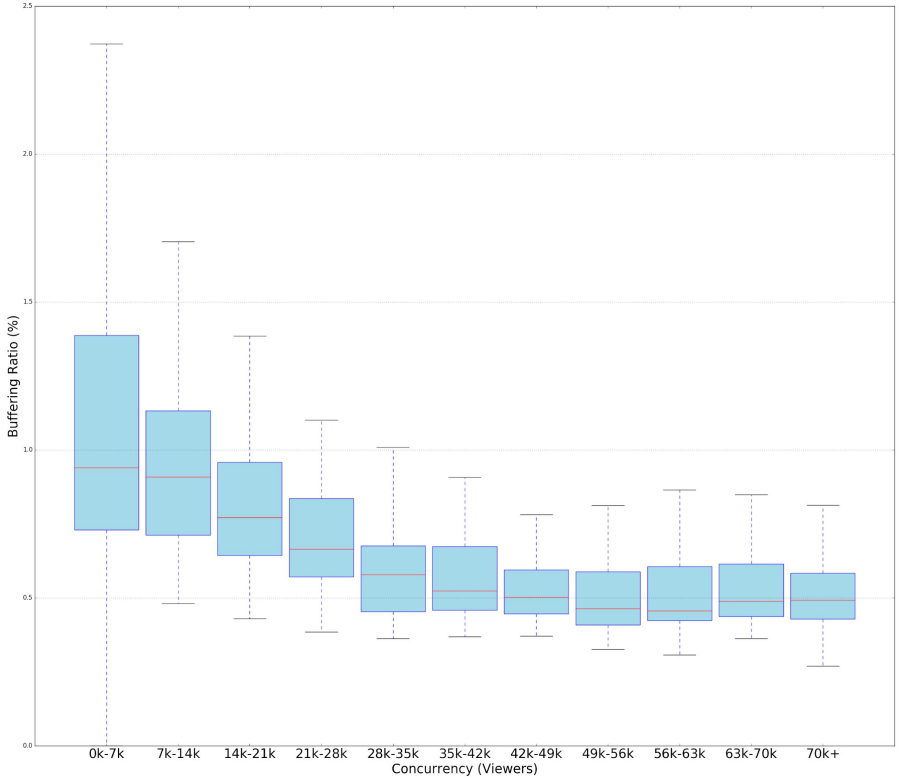
4 Results and Case Study

Improvements to rebuffering across our customer base

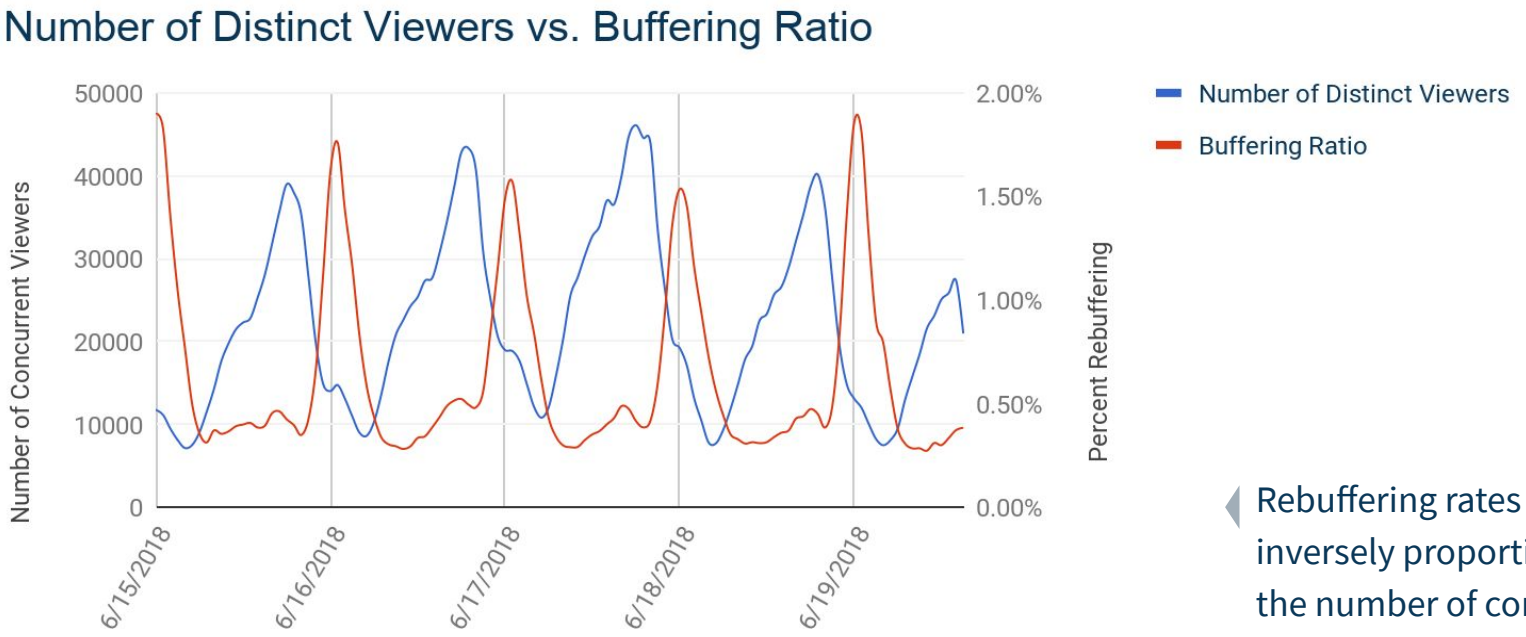
country	sum__payloads	[QOS] - Buffering Ratio SR On	[QOS] - Buffering Ratio SR Off	[QOS] - Buffering Ratio SR On vs SR Off %	[BASIC] - Percentage SR Disabled
France	923M	0.39%	0.42%	-6.33%	0.65%
Russia	521M	0.43%	0.62%	-30.73%	0.97%
United States	264M	0.62%	0.73%	-14.96%	0.77%
Spain	44.7M	0.38%	0.44%	-13.46%	0.11%
Canada	28.1M	0.51%	0.62%	-17.25%	0.69%
Brazil	14.2M	2.38%	3.08%	-22.81%	1.11%
Germany	8.25M	0.49%	0.58%	-15.71%	0.61%

The more viewers connected to streams, the less rebuffering

Rebuffering rates drop as the number of concurrent viewers increases on streamroot-enabled streams

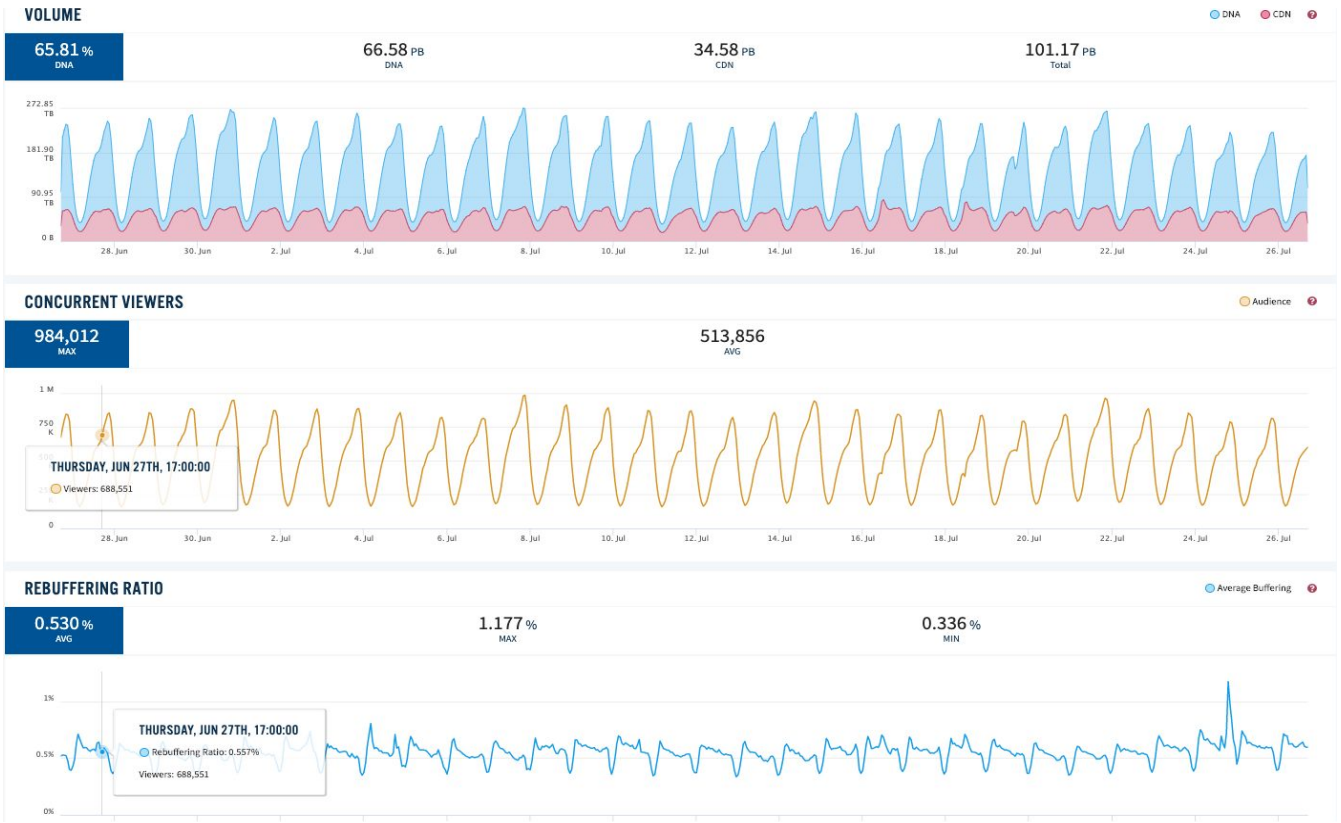


Global buffering improvement during spikes

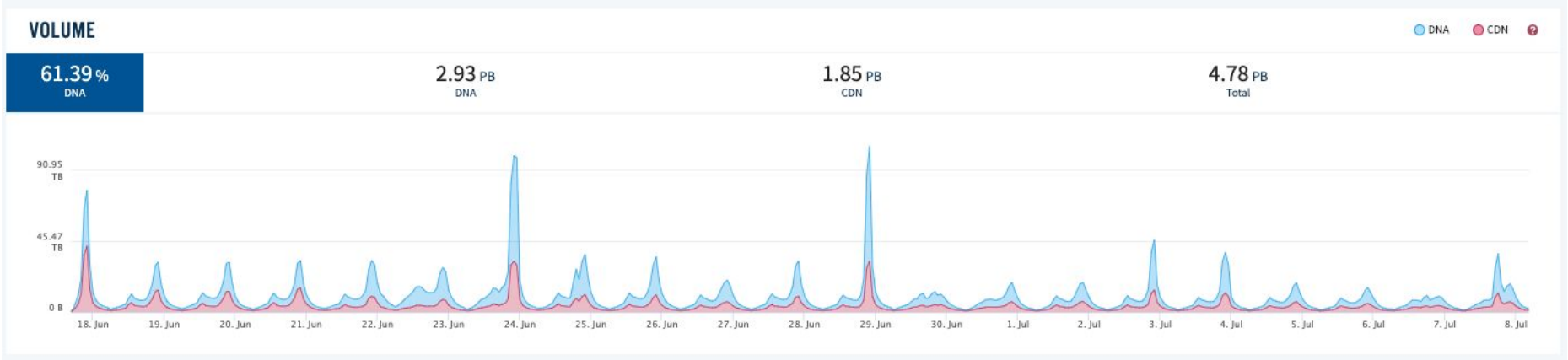


Rebuffering rates are inversely proportional to the number of concurrent viewers.

Consistently high offload and quality



Consistently high offload and quality



Thank you!

Pierre-Louis Théron
pierre-louis@streamroot.io