

# System Evaluation

Al Hinai, Abdullah	Andersson, Conny	Forouzani, Sepehr
Halteh, Faris	Ivanou, Aliaksandr	Karkanis, Iosif
Klingsbo, Lukas	Lan, Fangming	Lång, Magnus
McCain, Daniel Sean	Noorani Subramanian, Varun	
Omer, Enghin	Santos Rivera, Juan De Dios	

January 16, 2015

## **Abstract**

We experimentally test the cache effectiveness and latency of our streaming solution, comparing it to other existing streaming services. The results of these experiments show that although our solution still has a few issues that have to be fixed, it already shows interesting results, leading us to believe that using NetInf for live video streaming is feasible.

# Contents

<b>1</b>	<b>Cache effectiveness evaluation</b>	<b>1</b>
1.1	Experiment setup . . . . .	1
1.2	Results . . . . .	1
1.3	Discussion . . . . .	1
<b>2</b>	<b>Experimental latency comparison</b>	<b>2</b>
2.1	General information on other streaming solutions . . . . .	2
2.1.1	Twitch . . . . .	2
2.1.2	Youtube . . . . .	2
2.1.3	Steam . . . . .	2
2.2	Experiment Set-Up . . . . .	3
2.3	Results . . . . .	3
2.4	Discussion . . . . .	4
<b>3</b>	<b>Conclusion</b>	<b>4</b>

# 1 Cache effectiveness evaluation

In this section, we show the experiments and results to measure the performance of our backend and effectiveness of our caching solution.

## 1.1 Experiment setup

Three EENR instances are set up on a single subnet, one of them hosting the NRS. Eight phones are connected to the same subnet via WiFi. One of the phones was selected to perform the streaming role, and the remaining seven phones were tuned in to that stream. The stream used a target bitrate of 1000 kbit/s and chunk size of 2 seconds. After a delay of approximately eight minutes, letting all the client phones stabilize their streams, a period of three minutes was chosen during which statistics from the routers were collected.

All phones were allowed to randomly pick a router to connect to.

## 1.2 Results

The results can be found in in Table 1. A total of 104 chunks were published during the 3 minute sample period.

Router	Clients	Streamers	Hits	Misses	Network RX	Network TX
Toaster	3	0	425	104	1.125 Mbit/s	4.915 Mbit/s
Bowser	3	1	698	0	2.083 Mbit/s	6.552 Mbit/s
Luigi	1	0	104	105	1.039 Mbit/s	1.967 Mbit/s

Table 1: Cache effectiveness experiment results

## 1.3 Discussion

We believe that the results of this experiment are quite curious. One might note that the total number of GET requests (1436) is significantly higher than the expected number ( $104 \times (2+7) = 936$ ). We speculate that this is due to a bug in the application which is caused by the prefetch of chunks mentioned in NOTIFY messages not being cached properly, and thus being re-requested when they are buffered in the player. This speculation seems to indicate that each chunk would be downloaded twice, which implies there should be a total of  $(104 \times (2 + 7 \times 2) = 1664)$  requests. The fact that the measured total is less than this number can easily be explained: it is due to a known deficiency in the code that fills the player buffer. When it receives chunks out of order, it will not insert the newly found chunks at the appropriate place in the buffer, but rather just drops them silently. As the chunks that were not downloaded twice arrived out of order, they were skipped during playback.

Other than this anomaly, these stats show that the caching scheme works as designed. None of the servers had to provide the full bandwidth requirements of 7 clients.

## 2 Experimental latency comparison

In this section we compare the streaming latency of our solution to three other streaming solutions:

- Twitch.tv
- Youtube
- Steam Broadcasting

### 2.1 General information on other streaming solutions

#### 2.1.1 Twitch

Twitch is a platform that provides video game live streaming at a very large scale range (approximately 5 million viewers), with a large number of concurrent streams. They use Apple HLS for clients and RTMP for streamers. The typical chunk size of a Twitch streams is 4 seconds, and they advertise a typical latency of 12 to 40 seconds.<sup>1</sup>

#### 2.1.2 Youtube

Youtube contains a service called Youtube live that provides small to medium scale general purpose live streaming by Youtube content producers, as well as large scale professional streaming of sports events, concerts and news. They use Apple HLS and MPEG-DASH for clients, depending on the device type the client uses and RTMP for streamers. The typical chunk size of a Youtube stream is 5 seconds, and they advertise a typical latency of up to 60 seconds.<sup>2</sup>

#### 2.1.3 Steam

Steam is a software delivery and DRM platform, primarily intended for video games. It recently announced[1] a streaming service, called Steam Broadcasting, primarily intended for video game live streaming. Currently, the feature is in beta testing, with restrictive limits on the number of viewers.

---

<sup>1</sup><http://blog.twitch.tv/2013/12/new-video-system-update-after-one-week-in-service/>

<sup>2</sup>[https://support.google.com/youtube/answer/2853700?hl=en&ref\\_topic=2853712](https://support.google.com/youtube/answer/2853700?hl=en&ref_topic=2853712)

## 2.2 Experiment Set-Up

We measure the latency of each solution three times. The average latency is taken.

The following computers were used for the testing:

**A** Desktop, Intel Core2 Quad Q9400 (2.66GHz) CPU, running Linux 3.17.

**B** Desktop, Intel Core2 Quad Q9400 (2.66GHz) CPU, running Linux 3.13.

**C** Laptop, Intel i5-3317U (1.7GHz) CPU, running Windows 8.1.

**D** Laptop, Intel i5-M430 (2.27GHz) CPU, running Windows 7.

The test of Twitch.tv and Youtube was measured using machine A with OBS-Studio<sup>3</sup> on the streaming side and machine B with Chrome 39 on the client side. The video stream was h264 at 720p resolution and 2500 kbit/s bitrate. The three samples were taken using three different client computers.

The test of Steam Broadcasting used two different Windows laptops (since the Linux client of Steam was not capable of streaming at the time of the experiment) for the streaming role. All machines were running Steam package version 1420770381. The three samples were taken using machine pairs C to D, D to C, and D to A.

The following phone models were used for the testing of our solution.

$\alpha$  Motorola Moto G X1032, 4-core Cortex-A7 (1.2GHz), running Android 4.4.4.

$\beta$  HTC One X+, 4-core Nvidia Tegra 3 (1.7GHz), running Android 4.4.4.

$\gamma$  Google/Samsung Galaxy Nexus, 2-core Cortex-A9 (1.2Ghz), running Android 4.4.4.

The measurements of our solution were taken while varying the chunk size from 1 second to 5 seconds. 6 seconds and above was not be tested as the chunk size exceeded the allowed size of IPC packets in Android. The streaming came from a phone of model  $\alpha$ . The samples were taken simultaneously on one phone of each model. The latency was noted both just after a stream was joined, as well as after 15 minutes of continuous playback, as the player does not attempt to catch up if the playback is temporarily paused by a transient slowdown. When measuring with a 1 second chunk size, the second sample was taken after only 12 minutes, as a memory leak in the MP4 library crashes the app with the streaming role after that point.

All clients and streamers were connected to the university network.

## 2.3 Results

The results can be found in Table 2 and Table 3.

---

<sup>3</sup><https://github.com/jp9000/obs-studio/commit/428a7def16c0d63d19842b61b89ce4fd285bd861>

Chunk Size	Initial Latency	After 15min
1s	10.7s	18s
2s	8.7s	18.7s
3s	10s	22.3s
4s	9s	11.3s
5s	12s	13s

Table 2: Latency experiment results for our solution

Solution	Latency
Twitch	17.3s
Youtube	35.7s
Steam	9.7s

Table 3: Latency experiment results for other solutions

## 2.4 Discussion

As can be seen in Table 2, the average latency of our solution actually decreases when the chunk size is increased up to 4 seconds, especially in the “after 15min” sample. We speculate this is due to the retry logic in the streaming phone getting sufficient time to recover from failed publish attempts, something that happens surprisingly often. We think the failures is caused by a bug rather than transmission errors in the network, and it is probable that fixing this bug will improve the latency when using smaller chunks.

Even so, the latency of our solution is very close to that of the best other solution. This is a good sign considering our solution has not yet been optimized much for latency.

## 3 Conclusion

These experiments show that the approach of using NetInf for live video streaming is possible and has promise of being comparable to traditional methods. However, they also showed that our solution require some more polishing to provide adequate stability and robustness.

## References

- [1] Valve Corporation. Introducing steam broadcasting. <http://store.steampowered.com/news/15117/>, 2014. Accessed: 2015-01-14.