Scalable Live TV Distribution with NetInf to Android Devices

[Demo submission]

Bengt Ahlgren
SICS
bengta@sics.se

Arndt Jonasson
SICS
arndt@sics.se

ABSTRACT

We demonstrate efficient live TV distribution over the Internet to Android devices that are modified to use the NetInf information-centric network service as transport. The adaptation of the standard HTTP Live Streaming format to the NetInf transport was straightforward, but required extending NetInf with names for dynamic content. The in-network caching and request aggregation of NetInf result in efficient multicast transport to the client devices.

1. INTRODUCTION

Information-centric networking (ICN) [1] is an approach for designing a future network infrastructure. Its main abstraction is *named data objects* (NDOs), on which all communication is based. Clients request NDOs by name, and publishers make NDOs available. The location of an NDO is secondary – any node holding a copy can satisfy a request for it, enabling ubiquitous in-network caching as part of the normal network service.

One strong motivation for ICN is that it is very suitable for the dominating traffic volume of large scale distribution of media content, both playback of stored content, as well as, distribution of live content. The in-network caching enables the scalability needed to distribute to many clients simultaneously.

In this demonstration, we show live video distribution to Android client devices. The video is distributed in the standard HTTP Live Streaming (HLS) [6] format adapted to NetInf ICN transport [2, 5]. The demo is a proof-of-concept prototype showing that it is straightforward to map HLS to NetInf transport, with some extensions, and that NetInf is able to fully utilise request aggregation and in-network caching to aggregate the load from many clients in the network, offloading the server and reducing network utilisation.

The rest of the paper is organised as follows. We briefly introduce the NetInf protocol and describe the mapping of HLS to NetInf in the next two sections. Section 4 describes the setup of the demonstrator, followed by some performance measurements in Section 5. The paper is then concluded in Section 6.

2. THE NETINF PROTOCOL

Network of Information (NetInf) [2] is an ICN architecture mainly developed in the SAIL EU FP7 project [4]. As part of the design of NetInf, the project defined the 'ni' naming scheme that has become IETF RFC 6920 [3]. The 'ni' naming scheme uses URI syntax and includes the message digest, or content hash, of the NDO as part of the name, as can be seen in the example in Figure 1. One benefit of including the message digest in the name is that it is straightforward to provide *name-data integrity*, that is, to verify that the data received is actually the data requested, a crucial function in an information-centric network.

```
ni://example.com/sha256;B_K97zTtFuOhug27fke4_Zgc4Myz4b_lZNgsQjy6fkc
```

Figure 1: Example 'ni' name for the named data object "Hello World!".

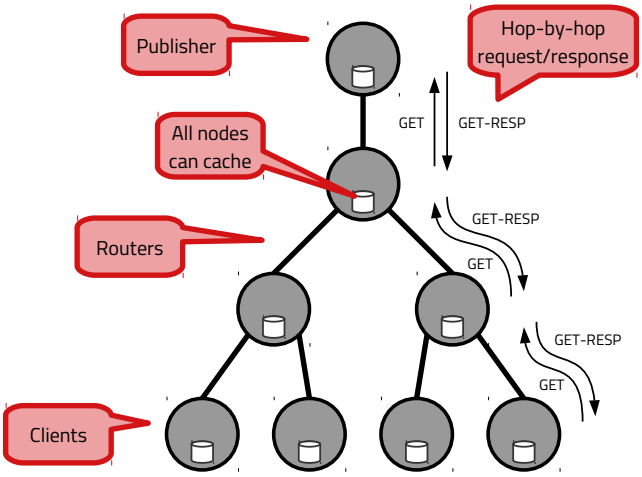

Figure 2: ICN messages and communication model.

NetInf defines three major protocol functions: `GET`, `PUBLISH` and `SEARCH`. Figure 2 illustrates the receiver-driven interaction model for an idealised small access network with clients at the bottom and a publisher/server at the top. `GET` messages are sent by the clients and forwarded hop-by-hop by the NetInf routers towards the publisher. Any intermediate

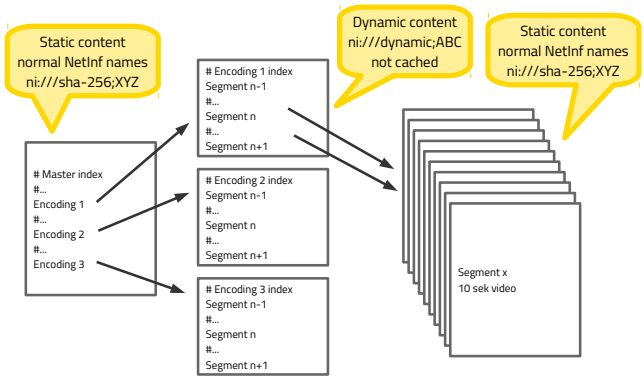

Figure 3: Live HLS video mapping to NetInf transport.

node that has a requested NDO, ultimately the publisher, responds with the corresponding `GET-RESP` message supplying the NDO.

3. MAPPING HLS TO NETINF

HLS defines three types of data objects, or files, that the clients retrieve in order to play the video stream. There is a master index file, illustrated at the left in Figure 3, that lists the available encodings of the video, including different bitrates and thus with different network capacity needs. The master index file can directly be named with standard 'ni' names, since its content does not change (not very often, at least). The content of the master index file has to be changed to use NetInf 'ni' names for specifying the encoding-specific segment index files (in the middle of the figure).

These segment index files, however, change each time a new video segment is created for the live content. Typically, the three most current video segments are listed. We can therefore not use 'ni' names based on content hashes, but instead needed to define a new name type for NDOs with dynamic content. For NDOs using these names, we need to complement the NDO with publisher signatures in order to be able to provide name-data integrity. This is however not yet implemented in the demo.

The clients regularly have to re-request the segment index file it has chosen to get the names of the video segment files. Since the latter do not change after they are created, regular 'ni' names works well. Like for the master index file, the segment index files have to be changed to list the video segments by their 'ni' names. When a new video segment appears in this index, the client can request that segment.

4. THE DEMO SETUP

The setup of the demo is illustrated in Figure 4. The live video can be taken from two sources. One is the regular, public, servers for TV4's live and stored TV content. The other is our own video camera. The TV streams from either can be published in real time in the NetInf system on the 'NetInf server' in the figure.

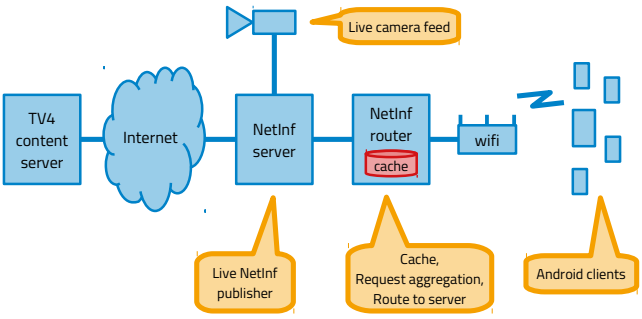

Figure 4: Setup of NetInf live streaming demo.

At the right hand side of the figure we have a set of Android clients running a modified version of the TV4 app that can use the NetInf network service for retrieving the HLS video. In between there is a NetInf router that cache all data, except for the dynamic segment index files. The router also implements request aggregation, so that requests from clients that are in sync with each other will only result in one single request to the server.

The demo setup makes use of simple default routing for directing the `GET` requests towards the NetInf server. In a larger setup, there is a clear need for more advanced routing schemes.

The Android clients run an Android NetInf implementation from Ericsson, and the NetInf server and router runs the open source NetInf implementation in the Python language that was developed as part of the SAIL EU project[1] and that has been extended with default forwarding, request aggregation, and rudimentary names for dynamic NDOs. The NetInf server and router runs on regular laptops with Ubuntu Linux.

5. MEASUREMENTS

To quantify the operation of NetInf in the demo we have carried out a few measurements on the NetInf router in the demo setup. Figure 5 shows the intensity of the network traffic at the NetInf router when five clients request the same live video stream. We can clearly see that the incoming traffic from the server (lower curve) is much less than the outgoing traffic to the clients (upper curve). The difference is in the order of five times, clearly showing that the router aggregates the traffic to the clients as we intended, relieving the server and upstream network from transferring multiple copies of the same video stream.

Figures 6 and 7 show histograms of the latency on the router for the time taken to complete two types of NetInf requests (`GET`/`GET-RESP` transaction). The x-axis is the time in milliseconds, and the y-axis is the number of occurrences in the measurement dataset for the respective latency interval.

The first of the figures shows the time to complete requests for video segment files. These files are around 600-800 KB in size and the total number in the analysed log is 19006.

[1]See `http://www.netinf.org`, and `http://sourceforge.net/projects/netinf/`.

Figure 5: Network traffic at the router.

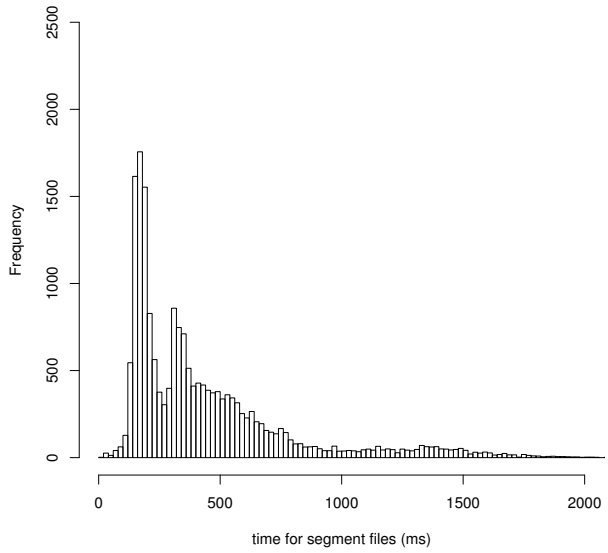

Figure 6: Latency of requests for video segment files.

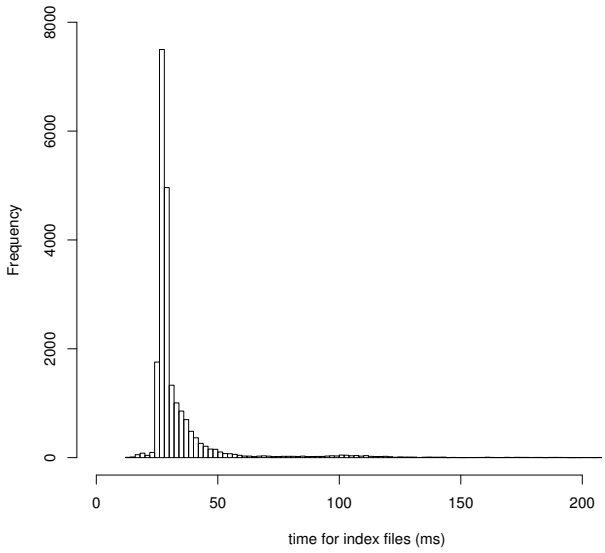

Figure 7: Latency of requests for index files.

We see two peaks and a fairly long tail. The first peak at about 180 ms most likely corresponds to files served from the router cache. The second peak at about 340 ms most likely corresponds to requests that were forwarded to the server. The long tail suggests that there are quite some variability in the service time. Some of the variability comes from the difference in size between the video segments.

The second figure shows the time to complete requests for the segment index files. These files are small, some three hundred bytes, and the total number in the analysed log is 21572. We only see one peak, since all these requests are forwarded to the server. Due to the much smaller size, the service times are also much lower with the peak at about 30 ms.

We should also point out that the CPU load of the NetInf router is quite low during these measurements, so the latency from processing should also be quite low.

6. CONCLUSIONS

The demonstration shows the feasibility and performance of live video distribution using the HTTP Live Streaming format adapted for the NetInf ICN transport. The caching and request aggregation of the NetInf transport result in efficient multicast to many clients. The caching removes the need for synchronisation between the clients, in contrast to the synchronous nature of IP multicast.

The adaptation of HTTP Live Streaming to use NetInf ICN transport was straightforward. A naming scheme for the dynamic segment index files had to be designed and implemented for the demo.

7. ACKNOWLEDGEMENTS

This work has been supported by the EFRAIM project funded by Vinnova in the challenge-driven innovation pro-

gramme, and by EIT ICT Labs. Many colleagues from EFRAIM and ICT Labs have considerably contributed to this work, especially Börje Ohlman and Linus Sunde at Ericsson Research and Anders Lindgren at SICS.

8. REFERENCES

[1] Bengt Ahlgren, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, and Börje Ohlman. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7):26–36, July 2012.

[2] Christian Dannewitz, Dirk Kutscher, Börje Ohlman, Stephen Farrell, Bengt Ahlgren, and Holger Karl. Network of information (NetInf) – an information-centric networking architecture. *Computer Communications*, 36(7):721–735, April 2013.

[3] S. Farrell, D. Kutscher, C. Dannewitz, B. Ohlman, A. Keranen, and P. Hallam-Baker. Naming Things with Hashes. RFC 6920 (Proposed Standard), April 2013.

[4] B. Kauffmann, J.-F. Peltier, et al. D.B.3 (D-3.3) final NetInf architecture. Deliverable D-3.3, version 1.1, SAIL EU FP7 Project 257448, January 2013. FP7-ICT-2009-5-257448/D.B.3.

[5] D. Kutscher, S. Farrell, and E. Davies. The NetInf Protocol. Internet-Draft draft-kutscher-icnrg-netinf-proto-01, Internet Engineering Task Force, February 2013. Work in progress.

[6] R. Pantos and W. May. HTTP live streaming. Internet-Draft draft-pantos-http-live-streaming-12, IETF Secretariat, October 2013. Work in Progress.