EXPERIENCES FROM A FIELD TEST USING ICN FOR LIVE VIDEO STREAMING

Adeel Mohammad Malik
Ericsson
adeel.mohammad.malik@ericsson.com

Bengt Ahlgren
SICS Swedish ICT
bengta@sics.se

Börje Ohlman
Ericsson
borje.ohlman@ericsson.com

Anders Lindgren
SICS Swedish ICT
andersl@sics.se

Edith Ngai
Uppsala University
edith.ngai@it.uu.se

Lukas Klingsbo
Uppsala University
lukas.klingsbo@gmail.com

Magnus Lång
Uppsala University
magnus.lang.7837@student.uu.se

ABSTRACT

Information Centric Networking (ICN) aims to evolve the Internet from a host-centric to a data-centric paradigm. In particular, it improves performance and resource efficiency in events with large crowds where many users in a local area want to generate and watch media content related to the event.

In this paper, we present the design of a live video streaming system built on the NetInf ICN architecture and how the architecture was adapted to support live streaming of media content. To evaluate the feasibility and performance of the system, extensive field tests were carried out over several days during a major sports event. We show that our system streams videos successfully with low delay and communication overhead compared with existing Internet streaming services, by scalability tests using emulated clients we also show that it could support several thousands of simultaneous users.

1. INTRODUCTION

Information Centric Networking (ICN) has attracted much attention from the Internet research community in recent years [1]. Its vision is to evolve the Internet infrastructure away from a host-centric paradigm to a network architecture based on named data objects. Data then becomes independent from its original location, and can be spread and made widely available through in-network caching. This approach can improve the network efficiency, scalability, and robustness.

A major benefit of ICN is that it enables caching of data objects in the network at the data level to reduce congestion and improve delivery speed. It can tackle the Flash Crowd problem on the Internet naturally and efficiently. Flash Crowds occurs when a large group of users access the same data source (e.g. web site or video) on the Internet, which cause traffic overload. Through caching of data objects at intermediate routers, queries and data streams of multiple clients can be aggregated along their paths to shorten the delay and reduce communication overhead. This applies in scenarios such as large crowds watching a game in an arena.

This paper presents a live video streaming system built on the NetInf ICN architecture [2, 3] and a field test conducted during the FIS Nordic Ski World Championship 2015[1] in Falun, Sweden. The system includes many ICN architectural elements such as naming, service discovery, aggregation and caching. The system functionality is implemented on a set of fixed NetInf routers together with a mobile streaming application developed for video recording and viewing. Experiments have been conducted with up to 20 mobile devices in the field capturing videos of athletics and streaming them in real-time from Falun to other clients at the event and at other sites. As the system currently runs as an overlay on the existing Internet protocols, and the NetInf routers have connectivity to the global Internet, the published video streams can be viewed by a client running our code anywhere on the Internet. However, the full benefits of the NetInf system such as aggregation and deaggregation of the streams, and on-path caching of data objects, are of course only available when connected to a network where there is a NetInf router present. Given the overlay structure and possibility to cross non-NetInf networks, it is possible to deploy a NetInf router anywhere in the Internet, making incremental deployment of the system possible. Our experimental results show that the system streams videos successfully with low delay and communication overhead compared with existing Internet streaming services Twitch and YouTube. We also made interesting observations on the system performance with different kind of network settings, including local WiFi, public WiFi, and 3G/4G connectivity. We also performed scalability tests that show that the routers we use can support more than 2000 clients each.

2. THE NETINF LIVE STREAMING SYSTEM ARCHITECTURE

Figure 1 shows the architecture of the NetInf live video streaming system. Users can record and publish video streams at a live event and at the same time other users can watch the streams live. The architecture also facilitates streaming to a device anywhere on the internet so that users not present at the venue can also publish or play streams.

Recording and playing clients at the event venue can connect to the system using local WiFi or mobile internet (3G/4G). Before a client can start publishing or playing, it first connects to a NetInf router. Consequently this router acts as the first hop NetInf node for the client. Clients on local WiFi at the event venue connect to a NetInf router in the local access network. Clients on the internet connect to a NetInf router in the NetInf core network.

The NetInf core network hosts a Name Resolution Service (NRS). This service is responsible for resolving object names into

[1]http://falun2015.com/

locators. It also provides search function for the registered Named Data Objects (NDOs).

ICN employs ubiquitous caching. Therefore every NetInf router in the architecture is coupled with a local cache. These routers cache NDOs on-path and serve them to corresponding GET requests when there is a cache hit. This ensures that clients are served data from the local network (if the data is cached) and that the edge links (like the one between the NetInf access network and the NetInf core network as seen in Figure 1) are not choked with traffic.

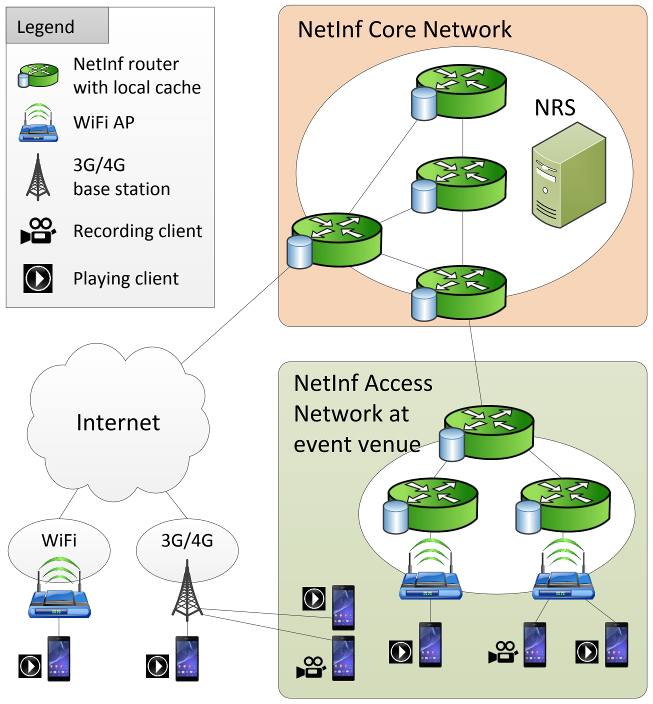


Fig. 1. System architecture

2.1. Stream Representation

In ICN content is abstracted in the form of Named Data Objects (NDOs). A video can hence be organized into several video chunks where each chunk is abstracted into an NDO.

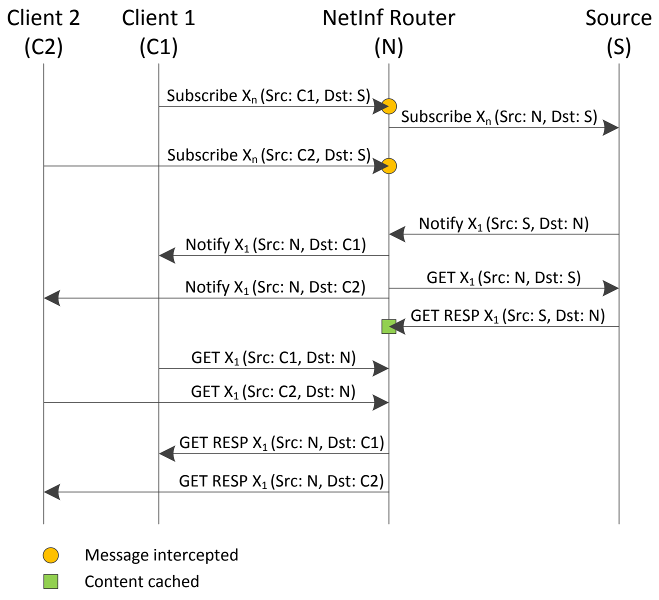
The entire video stream is represented by a single Header NDO. In other words, the Header NDO glues together all video chunks of a stream and presents itself as a single point of reference in order to request any subset of a video. The Header NDO contains the metadata for each video i.e. a description of the video and the geolocation of where the video is recorded. When subscribing to a live video stream, a client in fact sends a subscription request for the Header NDO.

2.2. Naming and the Name Resolution Service (NRS)

NetInf employs hash-based names as described in RFC6920 [4]. In this scheme, SHA-256 is used to derive the name for an NDO. A Name Resolution Service (NRS) stores name-locator bindings of the published NDOs. These bindings are used to resolve object names into locators. According to Figure 1 the NRS is located in the NetInf core network but it may be located elsewhere e.g. in the NetInf access network. The NRS also stores metadata for each NDO registered in it. The metadata is an important part of the system as it facilitates some important functions in the system such as populating the Event Browser as described in Section 2.9 and the seek function in video playback.

2.3. Subscribe-Notify Protocol and Content Retrieval

The NetInf live video streaming system uses a hop-by-hop Subscribe-Notify protocol between the requesting client and the data source. Figure 2 illustrates a signaling sequence of the Subscribe-Notify protocol between two clients, a NetInf router and a data source.



Client 2 (C2)
Client 1 (C1)
NetInf Router (N)
Source (S)
Subscribe X_n (Src: C1, Dst: S)
Subscribe X_n (Src: N, Dst: S)
Subscribe X_n (Src: C2, Dst: S)
Notify X_1 (Src: S, Dst: N)
Notify X_1 (Src: N, Dst: C1)
GET X_1 (Src: N, Dst: S)
Notify X_1 (Src: N, Dst: C2)
GET RESP X_1 (Src: S, Dst: N)
GET X_1 (Src: C1, Dst: N)
GET X_1 (Src: C2, Dst: N)
GET RESP X_1 (Src: N, Dst: C1)
GET RESP X_1 (Src: N, Dst: C2)
Message intercepted
Content cached

Fig. 2. Subscribe-Notify protocol and Content Retrieval

When a user selects a live stream for viewing the client sends out a subscribe request for the Header NDO of the video. This can be seen in Figure 2 where $C1$ and $C2$ send subscribe requests for X_n towards the data source. NetInf routers along the path taken by the request intercept the request and establish state for the requesting client. Subsequently they initiate a request of their own towards the data source. Any more subscription requests for the same video stream are aggregated. Requests from $C1$ and $C2$ are intercepted and aggregated by N into a single request towards the source. Note that in this case $C1$ and $C2$ will be subscribers to N while N will be a subscriber to S.

Notifications are triggered at the data source every time a new video chunk for the requested video stream is published at the data source by the publisher. Notifications are deaggregated in a similar fashion as the requests are aggregated. Notifications contain the name of the published NDO that contains the new video chunk. On receiving the notification the clients send a NetInf GET request for the published chunk. The GET request is destined for the next-hop NetInf node with which the client has established a Subscribe-Notify relationship. Figure 2 shows that S sends out a notification for NDO X_1 to N. Subsequently N sends out notifications for NDO X_1 to $C1$ and $C2$. N then sends a GET request to S for the notified NDO and receives it in a GET RESP from S. Also $C1$ and $C2$ send a

GET request to N for the notified NDO and receive it in a GET RESP from N.

The Subscribe and Notify messages employ the NetInf UDP convergence layer. These messages have a simple reliability mechanism built-in to ensure that the messages are retransmitted if lost in transit. The NetInf GET and GET RESP messages, however, employ the NetInf HTTP convergence layer [3].

2.4. NetInf Service Discovery

As mentioned in Section 2 a client should connect to a NetInf router before it can start publishing or playing a video stream. Clients connected to the local access network at the event venue via WiFi use Multicast DNS (mDNS) to discover the NetInf router that they should connect to. Clients on the internet connect to a NetInf router in the NetInf core network. For this they use DNS resolution to resolve a fixed name to the IP address of a NetInf router in the NetInf core network. In order to load balance between different NetInf routers in the core network the DNS name can be resolved to more than one IP address using several DNS A records.

2.5. Caching

Each NetInf router has a local cache to cache NDOs on-path. Every router also runs a local NRS exclusively for retrieving NDOs from the cache. In this case the NRS only serves as a database to check if an NDO is cached locally or not. GET requests originating from the clients are intercepted by the NetInf routers. The NetInf routers then check if the NDO is cached locally. If yes, the request is served with the requested NDO. Otherwise the request is forwarded towards the data source.

The NetInf live video streaming system also allows the users to play recorded videos. This is where caching is primarily useful. Note that while playing a live video stream the benefits of caching are less frequently used because all the subscribers requests for the recently published video chunks arrive within a very short time intervall. For live video streaming it is request aggregation, as described in Section 2.6, that provides the needed benefits.

2.6. Request Aggregation

Request aggregation is an characteristic feature of ICN. In this system the NetInf routers aggregate subscription requests and GET requests of users for video streams and chunks. Placing NetInf routers at network edges would allow for traffic aggregation on the transit links. In this case traffic aggregation on the link between the NetInf core network and the NetInf access network implies that only one flow per video stream will traverse the link at a given time.

Subscription requests are aggregated at each NetInf hop along the path from the requesting clients to the source. This eventually results in a tree of hop-by-hop point-to-multipoint subscription relationships where each node along the path from the requesting client to the data source only interacts with its next-hop NetInf node. Notifications generated by the data source when video chunks are published are deaggregated along the path from the data source to the clients.

2.7. Searching and Metadata

The NRS allows for search queries based on metadata of the registered NDOs. The registered NDOs have well-defined JSON schemas for the metadata which are used to parse the different fields in it.

There are two types of NDOs registered in the NRS. First the Header NDO and second the NDOs for video chunks. The two types of NDOs have the following key fields in the metadata.

Table 1. NDO metadata

NDO type	Metadata fields
Header NDO	Video name, Video description, Video geolocation
Video chunk NDO	Header NDO name, Timestamp, Sequence number

NRS search is used for a number of functions e.g. populating the Event Browser, retrieving video chunks when a seek is performed during video playback and retrieving video chunks for which notifications are lost.

2.8. Video Encoding and Chunking

The video generated by the recording device is encoded using H.264. The video data rate is 1 Mbps and the resolution is roughly 864×480. The video is split into chunks so that each chunk fits into a NDO. Chunking is done at I-frames in order to ensure that video chunks are independent of each other. The H.264 encoded chunks are packaged into MP4 containers. Each chunk corresponds to a video playout of 2 seconds. This parameter can be tweaked to achieve different tradeoffs between the playback latency and the NDO header overhead. The playing application buffers video data of roughly 10 seconds.

2.9. Event Browser

An Android application enables the clients to record and view live and recorded video streams. The application also provides an integrated Event Browser. The Event Browser is used for video stream selection. It provides a List view and a Map view. The List view shows a list of all the live and recorded streams while the Map view plots icons over a map of the arena corresponding to the geolocation of where the video streams are recorded.

The application polls the NRS periodically to populate and keep the Event Browser updated with the list/map of video streams. In order to do this a NRS search is performed for all the Header NDOs registered in it. To overcome the disadvantages of the update latency associated with polling, the information about changes in the list of published videos can be its own NDO to which the application can subscribe.

2.10. Security - Signing and Hash-Based Names

NetInf inherently provides content integrity through the use of hash-based names to name objects. In this system every time a data object is produced it is hashed to derive a name for it. The NDO is additionally signed by the publisher to ensure the authenticity of the NDO. The signature is added to the metadata of the the NDO. This signature is also included in the notifications for published video chunk NDOs so that the receiving clients can verify that the notification was not forged by an attacker.

3. FIELD TEST AND EXPERIMENTAL RESULTS

To test the NetInf video streaming system in a real-life scenario, a prototype of the system was deployed and tested during the 2015 FIS Nordic World Ski Championship in Falun, Sweden.

3.1. Network Setup

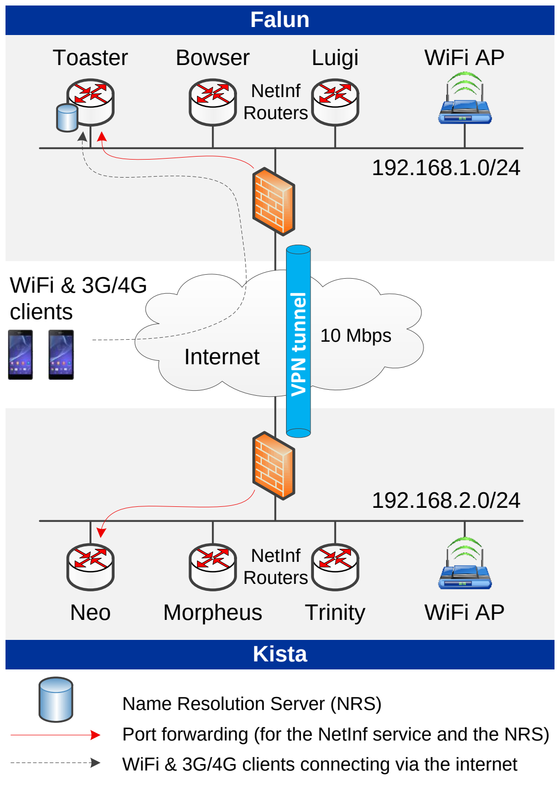


Fig. 3. Network setup

Figure 3 shows the network setup that was used to carry out the field test. The network setup spans two sites in Sweden, Falun and Kista. Identical setups were used in the two sites with three NetInf routers installed at each site (Toaster, Bowser and Luigi in Falun and Neo, Morpheus and Trinity in Kista) and a local WiFi access point connected on the same subnet as the routers to facilitate client connections. Besides the local WiFi that we set up in Falun, a commercial telco operator had provisioned free internet access on a separate WiFi which was also used to run the field tests.

Firewalls are installed as gateways in Falun and Kista. A VPN tunnel is configured between the two firewalls and any traffic exchanged between the nodes in Falun and Kista is routed through this tunnel. The networks in Falun and Kista are separate subnets. The separation was needed to ensure that clients connecting to the WiFi access point in Falun only discover the NetInf routers in Falun (via mDNS) while clients connecting to the WiFi access point in Kista only discover the NetInf routers in Kista. Traffic exchanged between Falun and Kista over the VPN tunnel is aggregated as described in Section 2.6. The network in Falun is limited by a 10 Mbps (uplink and downlink) internet backbone.

Clients on the Internet connect to Toaster in Falun via DNS resolution as described in Section 2.4. Toaster also hosts the NRS used by all the nodes in Falun and Kista. Port forwarding is configured at the firewall in Falun towards Toaster for ports used by the NetInf service and the NRS.

The server hardware used for the NetInf routers has a quad-core Intel Xeon E5-2407 v2 2.4 GHz processor, 24 GB of memory and a 480 GB read intensive Solid State Drive. The software for the NetInf routers is written in Erlang and the streaming application is Android-based.

3.2. Tests and Measurements

During the field test in Falun, we performed end-to-end playback delay measurements for the NetInf live video streaming system. These tests were performed using different connections (local WiFi in Falun, Telco WiFi in Falun, Kista WiFi, 3G/4G) for the recording and playing clients. For comparison with commercial streaming services we also performed end-to-end playback delay measurements for live streaming on Twitch and YouTube.

The field tests were performed with up to about 20 Android mobile devices. To get a measure of how the system scales with a very large number of clients we also performed tests with emulated clients. More specifically, here we measured the aggregation efficiency of a NetInf router across the 10 Mbps bottleneck link between Kista and Falun (the VPN tunnel) when a huge number of clients access the same live video stream.

The NetInf routers can be configured to generate several different kinds of logs. During the field tests they logged the transmit and receive throughput and the CPU load. We also performed qualitative testing in Falun to check the robustness of the system. Here we tested the system with several recording clients publishing content at the same time and several playing clients accessing a live stream at the same time.

3.3. Results

Table 2 and Table 3 show the playback delay measurements recorded for NetInf and Twitch/YouTube live video streaming, respectively. The tests were performed such that the recording client and the playing clients are not geographically co-located (the recording device was in Falun while the playing devices were in Kista). The measurements show that NetInf live video streaming achieves playback delays similar to what the commercial streaming services offer today. We do note however that both NetInf and commercial streaming services can be configured to achieve lower delays if needed.

Table 2. Playback delay (in seconds) of NetInf live video streaming

Recording client connected to:	**Streaming client connected to:**			
	Local WiFi	**Kista WiFi**	**HSPA+**	**4G**
Local WiFi	13.37	12.40	12.92	21.50
4G	17.18	14.18	15.10	16.40
Telco WiFi	13.60	16.70	22.20	18.10

Table 3. Playback delay (in seconds) of Twitch/YouTube live video streaming

	Test run #					
	1	**2**	**3**	**4**	**5**	**6**
Twitch	14.28	13.20	12.48	16.79	12.59	12.39
YouTube	22.38	21.92	16.17	17.93	16.21	24.63

Figure 4 shows the results of measurements performed with emulated clients to measure the aggregation efficiency of the NetInf router. Here a single recording device published content to Toaster in Falun and the emulated clients were run in Kista. The results show that with the hardware used as mentioned in Section 3.1, the NetInf router can aggregate video traffic (roughly 1 Mbps for a video stream) from 2000 clients with a CPU utilization of around 50%. Through extrapolation it can be deduced that the Netinf router can aggregate video traffic from roughly 4000 clients when completely loaded.

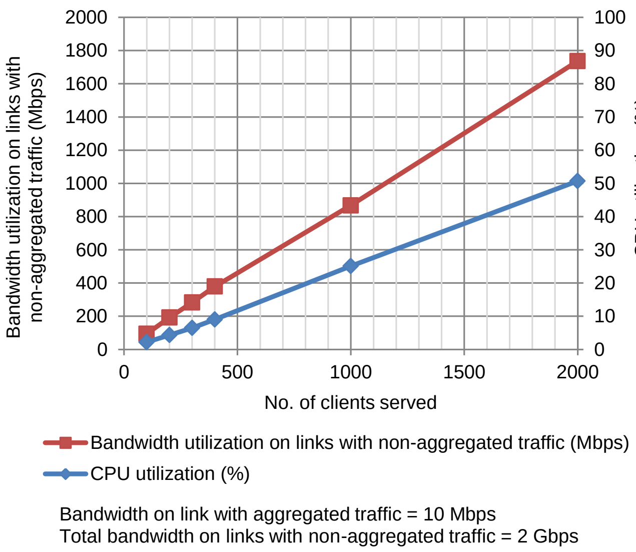


Fig. 4. Aggregation efficiency of the NetInf router

In the first of the qualitative tests, checking the general functionality and performance of the system, we let about ten of the Android phones simultaneously publish live video streams. Figure 5 shows the network and CPU performance during this test for ‘Toaster’, the main NetInf router. During this time interval, the average NetInf PUBLISH rate was about 2 per second. We can see from the performance graph that this test poses a quite low load on the NetInf router. The average CPU load (in total for all four cores) for this time period is about 4 %, and the average network receive load (RX, including the video sent by the publishers) is about 7.8 Mbit/s.

In the second qualitative test, we let one Android client publish a live stream, while most of the other watched that stream. Figure 6 shows as expected that the network transmit load dominates (TX, with average 9.4 Mbit/s, including the video sent to the clients) over receive load (RX, with average 2.2 Mbit/s) for the same router as

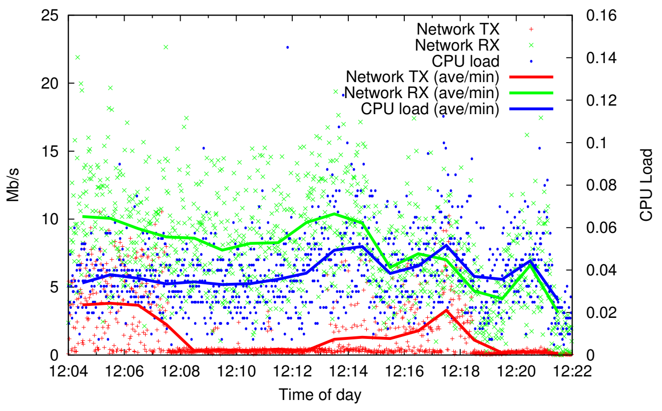


Fig. 5. Network and CPU load with many publishers.

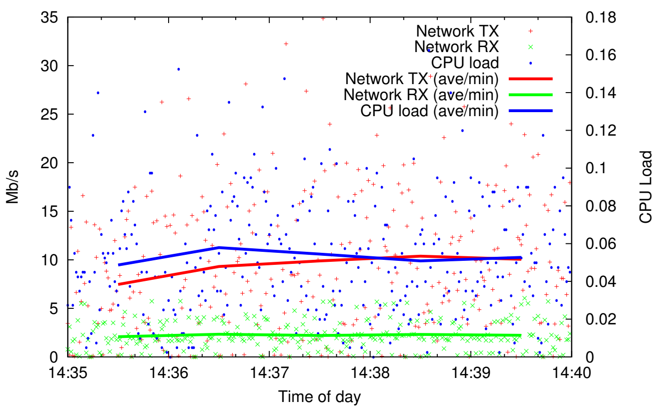


Fig. 6. Network and CPU load with many viewers.

above. The average CPU load for the time interval is about 5 %, and the average NetInf GET count is 5.6 per second, implying at most 11 active clients receiving from this particular router.

4. DISCUSSION

In this section we discuss practical experiences we gained from doing the field test in Falun and technical issues such as the scalability of ICN technology.

4.1. Experiences from doing the field test

In order to successfully perform this field test at a World Championship event, two major challenges had to be addressed. The first one was to get approval to install our test equipment at the event venue, which was solved through good internal contacts with event organizers and network providers serving the event. The second one was to ensure development of sufficiently stable software prototypes for the test. For the needed software development, we partnered with a student project course at a well-renowned university involving 13 undergraduate students for a full semester. We acted as customer and provided technical supervision for them.

After these thorough preparations, the field test was carried out without major issues. The system worked very well; for example, the NetInf routers have been running for months and the Android clients are very easy to distribute and install via a download link on a web page.

It is important that feedback on if things have been published is provided by the application to the user. When publishing or viewing didn’t work it was in most of the cases due to bad connectivity (usually WiFi problems). The user would benefit from having more information directly in the app about the current connectivity status.

4.2. ICN and global flash crowds

Current CDNs work well for provisioned media distribution, e.g. TV and VoD streaming services, but do not work very well when the demand for the services is not known. Even large corporations like Apple and BMW have had to see their webcasts of live events break down due to unexpected large crowds wanting to see the event. This has happened even though the events were planned for long in advance and using state of the art CDN providers. For true flash crowds like when a video from a single user's Android phone goes viral there are no solutions that scale to global audiences.

While we have only performed a limited field test with the live video streaming NetInf technology, by combining those results with those we have from the scalablity and the Twitch/YouTube tests we think the scalability properties of ICN technology in general and NetInf in particular look very promising. With the field test we have shown that the technology, e.g. aggregation and caching, works as expected.

We have also shown that our simple unoptimized prototype performs at least as well as current publicly available services, e.g. Twitch or YouTube Live. With the hardware we used, as described in Section 3.1, we can aggregate more than 2000 clients on one router. This means that if we have a three level hierarchy we can, with the prototype boxes we have today, stream to a global audience of 8 billion users from one Android phone without having to do any preconfiguration of the network. That can be compared with the servers needed for today's upload services. In an ICN network no pre-subscriptions are needed.

5. CONCLUSIONS

We presented a live video steaming system developed based on the NetInf architecture. It includes ICN features including naming, caching, request aggregation, searching on metadata, together with an Android mobile application for recording and viewing the video streams in real time. We conducted a field test in the Nordic Ski World Championship 2015 in Falun, Sweden. Experimental results show that our system can achieve live video streaming delay comparable with popular public services, e.g. Twitch and YouTube, even though optimization has not yet been enforced. We also demonstrated that our system is able to aggregate requests from large amount of clients efficiently without any additional network configuration or pre-subscriptions. In the future, we plan to further improve the system performance by optimizing caching and scale up the experiments to include more number of users.

6. ACKNOWLEDGMENTS

We would especially like to acknowledge the student project at Uppsala University that developed most of the software that was used for these experiments. We are also very grateful to the people at the Lugnet Ski Stadium in Falun, Sweden, that made this field test possible and helped us in many ways. This research has in part been funded by the EIT ICT Labs activities "14082 Networks for Future Media Distribution" and "14326 MOSES – Mobile Opportunistic Services for Experience Sharing".

7. REFERENCES

[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *Communications Magazine, IEEE*, vol. 50, no. 7, pp. 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*, vol. 36, no. 7, pp. 721–735, 2013.

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

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