Mobile NDN-Based Dynamic Adaptive Streaming over HTTP

Denny Stohr*, Fares Beji*, Rahul Dwarakanath†, Ralf Steinmetz† Wolfgang Effelsberg*
Technische Universität Darmstadt
*Distributed Multimedia Systems, {denny.stohr, wolfgang.effelsberg}@cs.tu-darmstadt.de, fares.beji@stud.tu-darmstadt.de
†Multimedia Communications Lab, {rahul.chini.dwarakanath, ralf.steinmetz}@kom.tu-darmstadt.de

***Abstract*—Named Data Networking (NDN) is a novel Internet architecture proposing a fundamental alternative to the current host-based routing. It addresses data objects instead of hosts—a concept promises to be especially useful for data-centric applications, e.g., dynamic adaptive streaming over HTTP (DASH). We demonstrate the implications of this networking architecture for the use case of DASH, based on a bare-bone wireless hardware testbed using a custom-build JavaScript player. In our demonstration, we show that NDN provides practical benefits with regard to resilience and for wireless handovers. Further, we propose the developed testbed platform for the practical evaluation of wireless handover and forwarding strategies in NDN networks, including new forms of mobile video production.**

I. Introduction

With the global video traffic estimated to reach 82 percent of all consumer Internet traffic by 2020 [1], along with the rising significance of mobile devices for video playback and production, flexible and efficient distribution of video content under challenging conditions is paramount.

Named Data Networking (NDN) is a novel Internet architecture which has its roots in Information-Centric Networking (ICN) [7] that promises to improve the efficiency for a wide range of applications that depend on the the Internet as a distribution network, such as Over-The-Top (OTT) video streaming e.g., dynamic adaptive streaming over HTTP (DASH).

In contrast to the traditional host-based Internet model, it features a data-centric approach where requests are based on file names rather than host addresses. In NDN there are two packet types: *interest packets* and *data packets*. Interest packets are issued to request a set of data. Data packets contain the actual data set. A client expresses an interest in a data set via an interest packet identifying a dataset name. The network answers with data packet containing the requested data. This not only promises a higher tolerance for client mobility and host failures, but also enables to use innovative routing and caching strategies essential for efficient video streaming architectures.

To demonstrate these characteristics and their practical advantages we introduce a mobile DASH streaming system over NDN. This project consists of three parts: 1) An NDN testbed that uses routers deployed as an NDN-enabled wireless network connected to multiple NDN File System (NDN-FS) nodes hosted on Raspberry Pi's. 2) A DASH client capable of video playback via the NDN testbed that has been implemented using JavaScript, fetching the Media Presentation Description (MPD) and video segments via the NDN rather than via

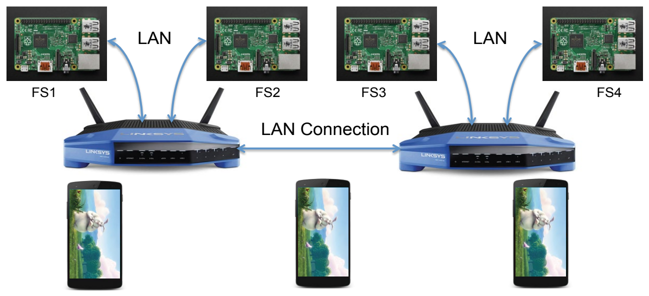


Fig. 1: Testbed architecture

the hypertext transfer protocol (HTTP). 3) A demonstration scenario for our given system, showing video playback on Nexus 5 mobile devices featuring fluent handover scenarios when switching between APs, as well as resilience to node failures. The remainder of this proposal is structured as follows: First, we give an overview of our system design in Section II. Next we depict how users interact with our application in Section III. In Section IV we describe related work, followed by the conclusions and planned future work in Section V.

II. System Design

We follow two major design goals for the demonstration: 1) Providing a mobile video streaming scenario featuring wireless routers as access points to allow for different data storages accessible via each router, and 2) achieving wide compatibility with consumer devices by employing web based development concepts.

To this end, the demonstration is built as a bare-bone NDN testbed composed of two Linksys WRT-1200-AC routers running a custom built OpenWRT, four Raspberry Pi 2 Model B, as well as one or more Nexus 5 smartphones acting as mobile clients, as depicted in Figure 1. In the following, we will introduce the main components of the demonstration, outlining key implementation and configuration aspects.

A. NDN-Enabled Routers

The routers act as the entry points to the network for the mobile clients. They host cross compiled versions of Named Data Networking Forwarding Daemon (NFD)[1] and WebSocket proxy[2] and provide an WebSocket-based interface

[1]https://github.com/named-data/NFD
[2]https://github.com/named-data/wsproxy-cpp

to NFD that allows for JavaScript based client compatibility (see Section II-C).

The 802.11ac WiFi is configured to allow seamless switching between the access points (client-based handovers), thus changing the routes to data hosts for connected clients while video content is streamed.

B. NDN Repositories based on Raspberry Pi's

The Raspberry Pi's function as the networks' video data repositories, serving requests depending on the forwarding strategy and location of clients. Each node executes an instance of NFD for packet forwarding as well as NDN-FS, based on the NDN-CPP library, which serves as the repository for the media files. This allows the nodes to act as file repositories from which the MPD and DASH segments are requested by the clients. After initialization and data processing, NDN-FS stores a segmented version of the MPD and of the media files which can be served to the clients. For segmented data storage, NDN-FS uses an SQLite database that also contains auxiliary information necessary for file retrieval. This includes, among others, the version number, chunk numbers, the MIME-types of the data as well as the chunk length.

C. NDN-enabled DASH Video Player

To address the second design goal, a client-side DASH video player was implemented using JavaScript and HTML5 Media Source Extensions (HTML5-MSE), allowing to stream MPEG-DASH-compliant AVC-encoded content. The custom-build implementation can be executed in all modern browsers, including mobile versions. It can be accessed via HTTP or directly via NDN (the latter requires a plug-in which is available for Firefox). For the NDN support in JavaScript, required by the DASH Player, the client is developed based on NDN.js[3], which has been introduced by Shang et al. [6]. All communication between clients and routers uses the WebSocket protocol to communicate with a proxy present on the router, as described in Section II-A.

Along the lines of the DASH standard, initialization of the video player is done by requesting the MPD, in conformance with *isoff-live:2011* [3], which is extended to allow NDN-based requests (see Listing 1). First, the prefix field contains the NDN prefix by which the video segments can be fetched. Second, the `!RepresentationID!` tag is a variable pointing to the id field of the requested representation. For example, the video or initialization segments of the representation with id 2 must be subsequently accessible via the URL `Prefix/2/init.mp4`. In the same way, the tag §`Number`§ points to the segment number requested, and it must be included in the file name.

Listing 1: Sample MPD for the Video Player

```
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011"
  mediaPresentationDuration="PT12M14S" minBufferTime="PT3
  .00S" profiles="urn:mpeg:dash:profile:isoff−live:2011"
  type="static">
  <Period>
    <AdaptationSet mimeType="video/mp4"
      segmentAlignment="true" startWithSAP="1"
      numberOfSegments="30">
      <SegmentTemplate duration="3000"
        Prefix="/ndn/broadcast/ndnfs/ndnfs_files/tears_of_steel/
        cleartext/video/" initialization="!RepresentationID!/
        init.mp4/"media="!RepresentationID!/seg−Number.m4f/"
        startNumber="1" timescale="1000"/>
        <Representation bandwidth="686685" codecs="avc1
          .42C015" frameRate="24" height="214" id="3"
          scanType="progressive" width="512"/>
          ...
    </AdaptationSet>
  </Period>
</MPD>
```

[3]https://github.com/named-data/ndn-js

D. Playback

To start a playback session, first, a *face* (representing a logical connection point in NDN) is initialized with the IP address of an access point to the NDN network. The segment URL is then computed from the information obtained after parsing the MPD. An interest request is sent to determine the availability of the segment and to fetch the appropriate version number. As soon as the first interest is successful, a new URL is computed from the obtained version number and a *fetcher* is initialized with this URL. The fetcher is responsible for downloading all the segment chunks until an end-of-file block is received.

After the completion of the transfer the video data is added to the buffer, a new offset time to the next segment is calculated and the main loop continues by requesting the next segment when the time comes. The full request process of the player is illustrated in Figure 2. The offset time to next segment is computed from the difference between the playback time and the buffer length. The algorithm gets the next segment as fast as possible until an amount n of not yet played segments are buffered. If n segments are present in the buffer, downloading the next segment is delayed proportional to the current playback time. If the resolution is switched, the next segment is downloaded without delay. This avoids a possible waste of bandwidth if the client stops watching in the middle of the video. For now, a basic adaptation logic is implemented, which measures the downloading speed of each segment and computes an approximation of the available bandwidth by taking the average of the last three download rates.

III. Demonstration Interaction

The main goal of the demonstration is to show a functioning DASH streaming scenario using NDN concepts by providing

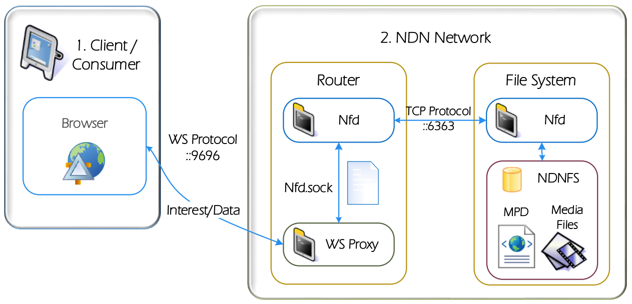
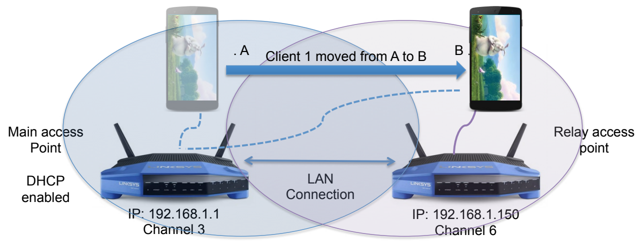


Fig. 2: DASH Player request pipeline



. A
Client 1 moved from A to B
B
Main access Point
DHCP enabled
LAN Connection
Relay access point
IP: 192.168.1.1 Channel 3
IP: 192.168.1.150 Channel 6

Fig. 3: Client based wireless handover

Nexus 5 devices as mobile video players to participants. This allows viewers to understand the underlying networking concepts of NDN and experience their potential benefits. These benefits will be further emphasized in two scenarios:

First, by changing the location of the mobile clients, thus registering to different access points, routes to the Raspberry Pi data hosts will change their costs (e.g. in terms of Round Trip Time (RTT))during runtime (see Figure 3). In a regular streaming scenario, the requests for the DASH playback would continue unaltered and the routes to data hosts with lower costs are neglected. In NDN, depending on the forwarding strategy used, the *best* route is dynamically selected for each request, providing a better quality of service (QoS).

Next, we will induce manual node failures. Here, a regular DASH streaming system would not provide continuous playback, without reinitializing the MPD providing an updated data host IP-address. In contrast to this, the playback session in the demonstrated scenario will provide a continuous playback as the identification of hosts is handled by the NFD which abstracts the knowledge of hosts from the client.

Further, a central monitoring User Interface (UI) will show live information regarding the currently used routes, based on the selectable forwarding strategy, as well as node utilization in real time. The user can configure two different routing strategies: `best-route` and `broadcast`. In the first case, the interest is forwarded to the lowest-cost next hop, which reduces the overall load on the system. In the second case, the interest is forwarded to all eligible next hops, allowing for a higher resilience in the case of node failure or handovers. The implications of the changes will be evident on the overall system utilization and performance as shown on the client devices and the central monitoring UI.

IV. Related Work

Muto et al. [5] introduced the first implementation of a JavaScript-based NDN DASH Player. They evaluated a train commuting scenario, where video elements are prefetched using NDN to servers located in railway stations. The work is implemented on the virtual LXC Testbed, and it is unclear if the NDN-based DASH player is being deployed on the client devices. In contrast to this work, we focus on showcasing actual handover for individual playback based on NDN requests within DASH on mobile devices in a physical testbed.

In a follow up work by Kanai et al. [4] the system was evaluated in a real-world scenario by deploying the proposed prefetching infrastructure in train stations and on trains. The developed DASH player was tested on mobile devices. Building on the same concept of a mobile JavaScript-based NDN DASH player, we focus on providing a live demonstration to users, focusing on client handover and resilience during streaming sessions instead of prefetching in a commuting scenario.

Another *ndn.js*-based work by Ishizu et al. [2] focuses on improving the efficiency of NDN requests by implementing interest aggregation and evaluating the influence of different playback buffer sizes for a DASH streaming scenario. The authors implemented functionality for fetching DASH segments based on *ndn.js*, however, they did not integrate an adaptive bitrate streaming.

V. Conclusions and Future Work

In this demonstration, we propose a mobile DASH streaming scenario based on NDN networking concepts practically showing improvements with respect to handover and routing efficiency.

In future work, we will extend the developed testbed architecture for the evaluation of new forwarding strategies and their influence on QoS and quality of experience (QoE) factors in DASH streaming. Further, we plan to minimize the usage of WebSocket connections to full NDN based communication from client to server by running a local instance of NFD on the Android devices themselves,[4] accompanied by a wider choice and evaluation of handover strategies, i.e. extended to network based handovers. Lastly, the novel scenario of User-generated Video (UGV) and network assisted composition can potentially significantly benefit from using NDN concepts given the simplified addressing of mobile video sources, and is therefore planned to be integrated in future work.

VI. Acknowledgment

This work has been funded by the German Research Foundation (DFG) as part of the project C03 in the Collaborative Research Center (SFB) 1053 MAKI.

VII. References

[1] *Cisco Visual Networking Index, 2014-2019 White Paper*. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf.

[2] Y. Ishizu, K. Kanai, et al. "Energy-Efficient Video Streaming over Named Data Networking Using Interest Aggregation and Playout Buffer Control". In: *2015 IEEE DSDIS*.

[3] *ISO/IEC 23009-1:2014*. http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=65274.

[4] K. Kanai, T. Muto, et al. "Performance Evaluation of Proactive Content Caching for Mobile Video through 50-User Field Experiment". In: *2015 IEEE Globecom Workshop*. 12/2015.

[5] T. Muto, K. Kanai, and J. Katto. "Implementation evaluation of proactive content caching using DASH-NDN-JS". In: *2015 IEEE WCNC*, pp. 2239–2244.

[6] W. Shang, J. Thompson, et al. "NDN.JS: A javascript client library for named data networking". In: *2013 IEEE INFOCOM*.

[7] L. Zhang, A. Afanasyev, et al. "Named data networking". In: *ACM SIGCOMM* 44.3 (07/2014), pp. 66–73.

[4] https://github.com/named-data-mobile/NFD-android