A hybrid NDN-IP Architecture for Live Video Streaming: A QoE Analysis

Ishita Dasgupta*, Susmit Shannigrahi†, Michael Zink*
*University of Massachusetts Amherst, †Tennessee Tech University
*ishitadg@cs.umass.edu, zink@ecs.umass.edu, †sshannigrahi@tntech.edu

2021 IEEE International Symposium on Multimedia (ISM) | 978-1-6654-3734-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISM52913.2021.00032

***Abstract*—With live video streaming becoming accessible in various applications on all client platforms, it is imperative to create a seamless and efficient distribution system that is flexible enough to choose from multiple Internet architectures best suited for video streaming (live, on-demand, AR). In this paper, we highlight the benefits of such a hybrid system for live video streaming as well as present a detailed analysis with the goal to provide a high quality of experience (QoE) for the viewer. For our hybrid architecture, video streaming is supported simultaneously over TCP/IP and Named Data Networking (NDN)-based architecture via operating system and networking virtualization techniques to design a flexible system that utilizes the benefits of these varying internet architectures. Also, to relieve users from the burden of installing a new protocol stack (in the case of NDN) on their devices, we developed a lightweight solution in the form of a container that includes the network stack as well as the streaming application. At the client, the required Internet architecture (TCP/IP versus NDN) can be selected in a transparent and adaptive manner.**

Based on a prototype we have designed and implemented maintaining efficient use of network resources, we demonstrate that in the case of live streaming, NDN achieves better QoE per client than IP and can also utilize higher than allocated bandwidth through in-network caching. Even without caching, our hybrid setup achieves better average bitrate over live video streaming services than its IP-only alternative. Furthermore, we present detailed analysis on ways adaptive video streaming with NDN can be further improved with respect to QoE.

***Index Terms*—Named Data Networking, Live Streaming, Software Defined Networking, quality of experience**

I. Introduction

With the advent of applications like Twitch, Facebook Live, live streaming has increased tremedously in popularity occupying upto 17% of the video traffic by 2022 [44]. Therefore mechanisms are required that deliver efficient QoE for live video streaming over existing internet infrastructures.

Apart from the traditional TCP/IP, multiple Internet architectures have been proposed for the future internet [1] [2] [39]. Information Centric Networking (ICN) [1] is one such future internet architecture that has the goal of replacing the host-centric approach of the current (TCP/IP-based) Internet with a content-centric approach. Named Data Networking (NDN) [48] is an instantiation of ICN that identifies and serves data by name instead of their location. In NDN, the communication is client-driven in the sense that it sends out "Interest packets" requesting content and receives a response as "Data packet" from the producer. Both these packets are stored in the NDN router for some time in Pending Interest Table (PIT) and Content Store cache, respectively. NDN routers forward packets using information from their Forwarding Information Base (FIB) and if an Interest arrives for a Data packet that is stored in the router's Content Store, the request is directly served from the cache. NDN can improve video streaming application performance due to its in-network caching and multi-path forwarding capabilities [36]. Obviously, the in-network caching characteristics of NDN are well-suited for live streaming events like the Superbowl or the FIFA Worldcup final, where the same content is requested by many viewers simultaneously (potentially in geographic proximity). Since NDN requires the replacement of the Layer 3 protocol, the immediate widespread adoption of this new approach is difficult. Experiences with the long and painful rollout of IPv6 and some new TCP flavors, which all require changes in the operating systems of virtually all nodes (routers and hosts) in the Internet, have clearly shown how cumbersome this process can be as well.

Motivated by these past experiences, we design, implement, and evaluate an approach that can transparently choose and adapt to multiple internet architectures (TCP/IP and/or NDN). This requires a portable, flexible and an easy to configure system that can be easily deployed on a variety of hardware platforms and dynamically adapt to application and network demands. To achieve this goal, our approach employs Software-Defined Networking (SDN) and Network Function Virtualization (NFV) [5] for network virtualization. Further, our approach is based on operating system virtualization as offered by Platform-as-a-Service container systems like Docker [3] or Singularity [17]. Such an approach bears the benefit that a particular application can run in a container with a completely pre-configured environment that does not require any user administration. Also, this bypasses the requirement for widespread replacement of the Layer 3 protocol in the case of NDN. Previous work [43] focused on the implementation of network elements that could support alternative Internet protocol stacks parallelly, allowing the clients to stream videos over both TCP/IP and NDN. However, our work presented in this paper focuses on the following new contributions:

1. Hybrid architecture with end-to-end approach serving miscellaneous viewing experiences over a multi-tier network infrastructure: We analyze our hybrid infrastructure over traditional IP with respect to live video streaming experience. This end-to-end approach supports transparent architecture selection all the way to the application running on end systems.

2. A QoE analysis on live streaming with NDN & IP: With

978-1-6654-3734-9/21/$31.00 ©2021 IEEE
DOI 10.1109/ISM52913.2021.00032

the focus to optimize user's QoE, we inspect the following:

- Average bitrate & bandwidth utilization by NDN and IP clients.
- The effect of caching on live streaming QoE.
- Existing drawbacks due to the way ABR algorithms work with NDN on user's QoE.
- Performance evaluation of NDN vs. IP clients accross server and client-side bottlenecks.

To summarize, this paper proposes a hybrid setup that is flexible to the network as well as the underlying physical layer such that it simultaneously supports multiple Internet architectures for better QoE in an ABR application. Furthermore, with our detailed QoE analysis, we justify using NDN along with our hybrid setup to improve live video streaming experience with minimal end-user involvement.

II. SDN and NFV Support for Parallel Live Video Streaming

The inherent caching properties of NDN have proven to support live streaming applications well [43]. However, to benefit from this improvement, significant changes throughout the network have to be performed to support NDN including installation at end systems. Hence, we propose a hybrid approach where clients can stream videos over both IP and NDN without needing NDN kernel support on hosts or fully replacing the TCP/IP protocol stack with NDN. We achieve this by utilizing virtualization techniques for standalone NDN-based container applications that can run on any client host supporting containerization. Further to enable the parallel support of IP and NDN protocol stacks, we combine SDN and NFV. Our primary goal is to build and evaluate an environment that can utilize the benefits of both these network architectures simultaneously in the domain of video-streaming while optimizing resource use and user experience. In that context, we present a brief overview on Live streaming with NDN and network components that enable flexible hybridity of our architecture.

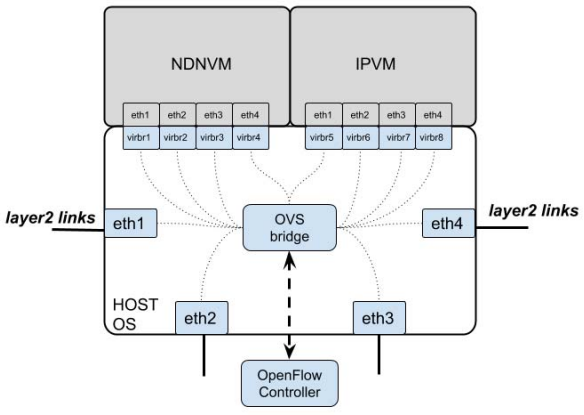

Fig. 1: SDN-NFV setup in an intermediate router

A. NDN and Live Streaming

NDN provides a set of features that are well-tailored for live streaming. Especially, the feature to cache content at each router (even if only small amounts can be cached compared to a CDN or web cache) supports live streaming where the same content is requested simultaneously by many viewers that might have a high degree of geographical locality. This essentially creates an efficient multicast mechanism. Thus, many user requests can be served from a router in the network instead of the origin server or a CDN edge server. Opposed to the traditional IP case where content sources are less diversified, in the case of NDN, DASH segments may come from various sources (a data publisher that is closest to the consumer, an in-network cache, and so on), when multiple sources are available. Additionally, each DASH segment will be chunked into multiple NDN Data packets that are 8800 Bytes in size (by default, the size of the packets is configurable). These Data packets may come from various sources as well. Further research is needed to create congestion control algorithms that can take into account this heterogeneity of Data sources in NDN [33].

The advantages of NDN for efficient delivery of video streams have been investigated in works such as [10] and [37], which make ICN-based protocols a likely candidate for the transport of live video. While the general feasibility of supporting live streaming with NDN has been demonstrated earlier [43], the approach in this paper focuses on providing transparent solutions that do not require users' involvement to configure the underlying network technology. Additionally, this paper provides a much more detailed evaluation of live video streaming based on our hybrid streaming architecture.

B. Network Devices

We have chosen an SDN supported NFV approach for the network device design that handles IP and NDN traffic in an isolated and adaptive fashion. This architecture enables flexible topology setup and efficient resource use. In addition, it can be easily extended to support other network layer protocols (e.g., IPv6 or IPSec). In our particular use case, this architecture is used to dynamically configure customized hybrid routers that simultaneously support IP(v4) and NDN. In this architecture, SDN is used to internally (within the device) direct traffic from the physical network interface to the respective virtual interface of the respective router. This is realized by running openvswitch [30] (OVS) on the Host OS of the node. An SDN controller implements these rules for isolating NDN from IP traffic in the OVS-based SDN switch as shown in Figure 1. While we use a centralized OpenFlow controller in our prototype, the architecture is designed such that any type of controller architecture (centralized or distributed) can be supported. Furthermore, our architecture reduces resource utilization in the case of live-streaming as we do not need client machines with NDN-supported kernels as we only need hosts that can run NDN-based container applications whenever live-streaming is requested.

III. Evaluation Setup

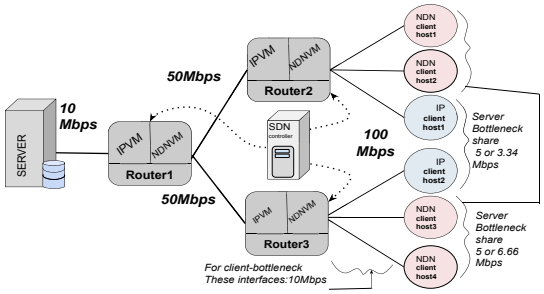

Fig. 2: Topology for our Evaluation setup

A. Experimental Setup

In this section, we describe the components of the architecture we employ for the evaluation of our approach.

Topology: Figure 2 shows the topology used for the evaluation experiments, which consists of a server, three intermediate routers (OpenFlow enabled), six Ubuntu clients, and a centralized SDN controller. The server node contains the videos being served for live streaming. The clients can be classified as an NDN client or IP client depending on which Internet architecture is being used by the client node for streaming. We chose this topology as we wanted to evaluate tiered cache effects amongst NDN clients in comparison with traditional IP clients.

We implemented this topology on the Cloudlab testbed [6] which offers bare metal servers providing flexibility to configure individual nodes exactly to our needs. CloudLab supports reproducibility, allowing us to share the complete setup and make the execution and outcome of our evaluation repeatable by other researchers. The decision to use a testbed for our evaluation instead of using a simulation or emulation environment like ndnSIM [21] or Mininet [12], respectively, came from the consideration that the latter would abstract several important factors that have an impact on the overall performance of our approach.

Network Setup: Figure 2 shows the available bandwidth at each link interface. We employ traffic control (tc) [7] to enforce the maximum bandwidth limits and the share of bandwidth that is available for NDN and IP traffic. Preliminary experiments revealed that TCP saturates the available bandwidth quickly due to highly optimized congestion control, slowing down NDN transfers. On the contrary, NDN is implemented in the application layer and currently lacks sophisticated congestion control mechanisms. If not mentioned otherwise, the link between Server and Router1 is shared equally between IP and NDN traffic. For the virtualized NDN routers (running on the physical router nodes 1-3 in Figure 2) the cache size was varied between 0MB, 250MB, and 500MB.

As we will see in the case of server-side and client-side bottleneck experiments in Sects. IV-B1 and IV-B4, the chosen bottlenecks were 5Mbps to serve upto 20 clients and 10Mbps to serve 5 clients, respectively. In the server-side scenario, this was done to maintain the proportionality of a limited bandwidth to multiple customers. In the case of client-side bottleneck,the highest quality segment in our dataset is 4.2Mbps. To serve 5 clients at highest quality, a bandwidth of 21 Mbps would be required, hence the bottleneck is set to 10 Mbps. In other words, these bottleneck bandwidth numbers were chosen keeping proportionality with real world in mind and not the absolute values themselves.

Virtualization: As introduced in Sect. II, network virtualization is implemented using a combined NFV-SDN setup, whereas containers are used for application-level virtualization in the case of NDN live streaming clients. The Kernel-based Virtual Machine (KVM) hypervisor is used at the routers to handle IP and NDN traffic separately and in an isolated fashion. Figure 1 shows the virtualized network configuration of Routers 1-3. The physical layer 2 links of the host are mapped to the virtual interfaces of the internal VMs via OpenVswitch [30], which is managed by the SDN OpenFlow controller. Internally in the router, the traffic is routed from the server to client nodes via these virtual VM interfaces at the intermediate routers.

Using this approach, the controller uses the EtherType field (0x8624 for NDN, 0x0800 for IPv4) of an incoming frame to determine which virtual interface it has to be forwarded to. As shown in Figure 1, an incoming IP packet on interface `eth1` would be forwarded to virbr5, while an outgoing NDN packet from virbr4 would be forwarded to `eth4`. More details on this configuration can be found in [43]. At the client side, Docker containers [23] are used for the NDN live streaming application. For the networking aspect of our containers, we used Docker macvlan networks [4] that connect each container's virtual interface to a host's physical interface. This also lets us monitor and regulate the traffic at Layer 2, which is needed for the NDN clients.

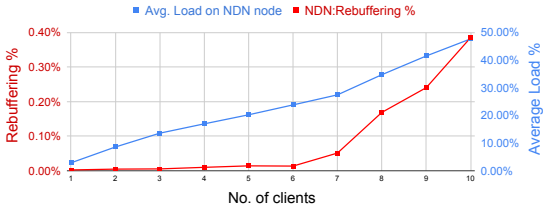

Fig. 3: Effect of increasing clients on CPU Load and Rebuffering percentage.

Deciding the number of NDN containers per host: For the large scale experiment scenarios we describe in Sect.IV, using one physical node per client would have required a total of 65 physical nodes. Due to resource limitations of the CloudLab testbed, using such a large number of physical nodes is not feasible. Therefore, we had to resort to run several NDN containers on the same physical nodes. We ran experiments to decide how many clients we could safely run on each physical node. We ran a series of experiments where we constantly increase the number of containers (NDN) running on a physical node. We start with a single client streaming over NDN and monitor the CPU load during the process as

well as the resulting Rebuffering percentage (see Sect. III-C for the definition of the metric). This experiment is then scaled up to ten clients and we observe an increase in the Rebuffering percentage along with CPU load, as shown in Figure 3. While both metrics increase with an increase in the number of clients, the Rebuffering percentage observed for the case of five NDN container-based clients per physical node is only 0.01%. The other QoE metrics (e.g., Spectrum and QS) show the same behavior though we do not include them for brevity. Since five clients do not significantly affect the QoE, we decided to run five NDN containers per physical host for most of our large-scale experiments.

Video: For our evaluation, we made use of the Big Buck Bunny video [13], which is encoded in the DASH/AVC format and supports up to 14 different quality bitrates. We use this well-known encoding format for our basic dataset that has been used in many works in the past. More recent encoding formats like HVEC would also work with our approach since it is encoding format agnostic, as long as there is MPD support for the latter. It should also be noted that the same data is compared across IP and NDN, hence the data itself or its encoding type, is of less importance with the respect to the comparative performance evaluation.

Video Server: Since DASH is employed as the streaming technique in this work, we use a regular web server (vanilla Apache2) for IP-based streaming. For NDN, the live-streaming content needs to be named down to the granularity of a DASH segment's quality, which can be achieved with minimal effort. For a given video "*v*" whose DASH segment "*n*" is of quality "*q*", the segment is named as "*<ndn_prefix>/<v>_<q>_<n>.m4s*." This content is served using *ndn-python-repo* [16] which creates a repository at the server containing the DASH video segments in their respective quality levels. Combined with *ndncatchunks* [29] on the client-side, it delivers data based on content names. *ndncatchunks* implements a TCP CUBIC-like [11] congestion control algorithm that can adjust the data transfer rate based on the observed network conditions (e.g., congestion, packet loss).

Video Client: For the streaming client, we use a python-based video streamer, *AStream* [14] that supports multiple DASH adaptation algorithms. Here, *AStream* uses HTTP libraries and *ndn-python-repo* combined with *ndncatchunks* to download video segments over IP and NDN, respectively. We selected BOLA [42] as the bitrate adaptation algorithm. We choose BOLA because it is a near-optimal state-of-the-art adaptive bitrate streaming algorithm for our player at the client and is also a part of the DASH reference player [41]. As long as the same ABR algorithm is used by NDN and IP for comparative analysis of the QoE, we can always use alternate ABR algorithms with this setup in the future.

B. Live Streaming

Since our focus is only on the streaming part, we ignore the production process of live content and only focus on the content delivery. For the evaluation of our approach, we use the Big Buck Bunny video as described in Sect. III-A, where only the very first client starts requesting segments from the very beginning of the video. We specify that this first request occurs at t_0. We log this time at the video server and once a request from a new client arrives (e.g. at $t = t_1$), we determine the starting segment for that client as $(t_1 - t_0)/segmentlength$. For example, if the first client starts requesting $t_0 = 0$ seconds and the next client request is received at $t_1 = 10$ seconds and we assume a video segment length of 2 seconds, then the second client is served the video from segment 5 onwards. To allow the client to determine the correct starting segment, this information is transmitted from the server to the client in the dynamic MPD file.

C. Metrics

Since one of our goals is to optimize QoE , we use the following metrics that are widely accepted as good representations for viewers' perceived quality. **(a) Average Quality Bitrate (AQB):** One of the objectives of quality adaptation algorithms is to maximize the average quality bitrate of the streamed video. For a comprehensive QoE representation, we need to combine this metric with the *Number of Quality Switches*. **(b) Number of Quality Switches ($\#QS$):** This metric is used together with AQB to draw quantitative conclusions about the perceived quality (QoE). For example, for two streaming sessions having the same AQB, the session with the lower $\#QS$ will be perceived better by the viewer. **(c) Spectrum (H) [49]:** This metric is a centralized measure for the variation of the video quality bitrate around the average bitrate. A lower H indicates better QoE. **(d) Rebuffering percentage (RB):** The average rebuffering percentage is given by $RB = \mathsf{E}\left[\frac{t_a - t_e}{t_e}\right]\%$, where t_a is the actual playback time and t_e is the entire video length in seconds.

IV. Evaluation Results

The aim of this paper is to present the feasibility and benefits of our hybrid network model over a traditional IP-based one with respect to live video streaming. Additionally, we also make a case as to why NDN should be favored over IP in live streaming scenarios. In this light, the results show that our hybrid model achieves higher bitrate as NDN achieves overall higher bandwidth utilization than traditional TCP/IP. Followed by this, we observe that improvement in certain QoE metrics for NDN live streaming clients increases with increasing cache size but not indefinitely. Finally, we identify drawbacks of implementing adaptive streaming with NDN leading to oscillation effects as well as suggest improvements to overcome them and make a case for improved QoE by NDN over IP even with client-side bottleneck. In order to make these evaluations, we carried out small as well as large-scale experiments on the hybrid model that also tests the scalability and robustness of our approach. It should be noted that for each experiment, all clients start streaming at the same time and the reported results were accumulated from an average of 10 streaming sessions.

Experiment I: Small-Scale This scenario serves as a baseline in observing the impact of increasing clients (running on a

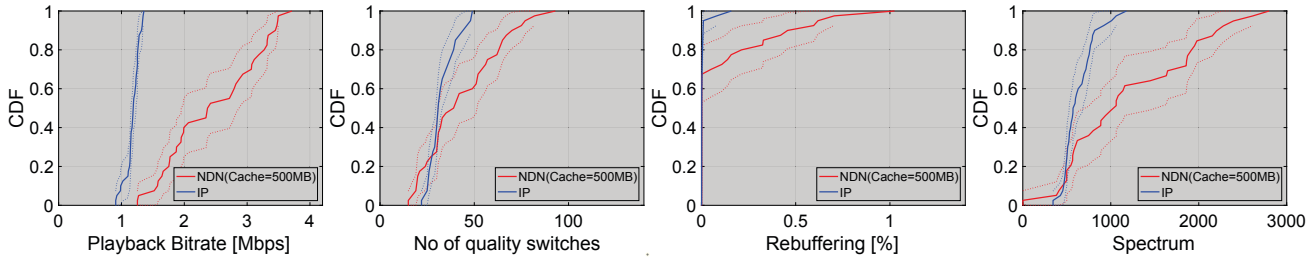

Fig. 4: Cumulative Distribution Functions for QoE metrics for the case of 4 NDN and 4 IP clients.

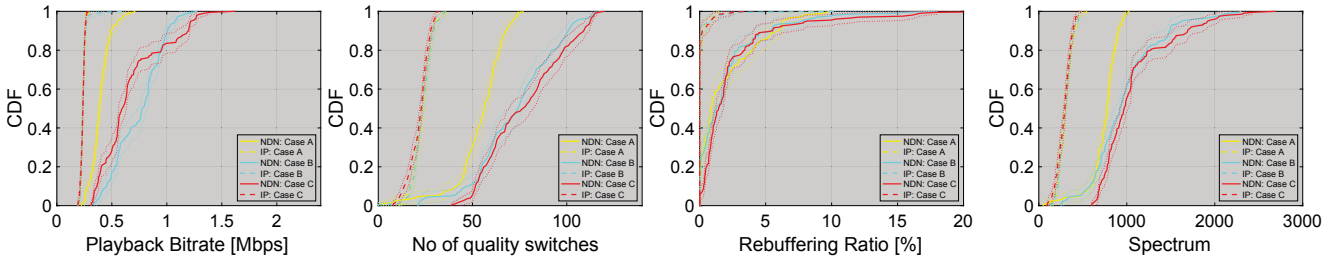

Fig. 5: Cumulative Distribution Functions for QoE metrics of 20 NDN client cases with varying cache sizes. Case A: NDN cache size 0MB; Case B: NDN cache size 250MB; Case C: NDN cache size 500MB

single physical node) on the overall QoE. For this initial small-scale evaluation of our approach, we run one NDN streaming client application in a docker container on each of the four physical client nodes (one container per node as shown in Fig. 2), while two IP-based streaming applications runs on each of the two physical client nodes (two per node).

Experiment II: Large-Scale For this set of large-scale experiments, we evaluate the following setups:

a) 20 NDN & 20 IP: Here, we increase the number of clients to 20 . Since we cannot increase the number of physical nodes in the topology, we have to run 20 IP clients on two physical nodes (10 on each) and 5 NDN containers on four physical nodes, each (see Fig.2). Apart from the QoE analysis, we chose this setting to also study **i)** the effect of varying cache sizes (0, 250MB, 500MB) on the performance of NDN clients and **ii)** compare the performance of NDN and IP clients when the bottleneck is on the client-side instead of server-side.

b) 40 NDN & 20 IP: Further testing the scalability, the number of NDN clients is increased to 40. This increase is motivated by the assumption that popular live streaming events will be watched by many viewers (almost in a flash crowd style) putting additional stress on the system. To adjust for the imbalance between the number of NDN and IP clients, we adjust the bandwidth allocation on the `10Mbps` link between the server and Router 1. As explained in III-A, this number was chosen to study the effect of limited server-side bandwidth on midgress traffic. For this experiment, we allocate $2/3rd$ of the bandwidth to NDN sessions and $1/3rd$ to IP sessions (proportional to the number of clients of each type).

The average QoE metrics for all experiments are shown in Table I. In this table, we present the Rebuffering percentage, #QS, average bitrate and spectrum reported across 10 streaming sessions for experiments I, IIa & IIb. In optimizing the viewer's QoE, a higher avg. bitrate and lower Rebuffering percentage, #QS and spectrum is preferred. More detailed analysis on these results are discussed in the next sections. The CDFs for QoE metrics for Experiments I, IIa & IIb are presented in Fig. 4, 5 and 6 respectively.

With respect to the end-to-end video-streaming architecture, this work focuses on the subset of this architecture from post video-processing to the distribution to the viewers. Henceforth, we do not regard the latency from video capture (camera or camera set at a live event) to making segments available on the video server in our evaluation. To estimate latency differences between IP and NDN clients between the central server and end-client, we compare the download time differences for identical content with Experiment IIa 's 500MB cache setup. The average difference amounts to less than 3% where NDN clients take 40ms more time than IP to download the same content. This difference can be further reduced with more cache hits and further code optimization (in the case of NDN) leading to reduced download times and thus reduced latency difference.

A. Hybrid Streaming Architecture Analysis

Our hybrid model has the flexibility to stream videos over multiple network types without affecting each other adversely. On top of providing this benefit, the overall impact on QoE also needs to be investigated. This evaluation allows the comparison between an IP-only scenario and a mixed NDN/IP scenario (as presented in experiments I & II). For IP-only case, we run 40 IP live streaming clients. Similar to NDN clients in the hybrid setup, half of the 40 IP live streaming clients also run in a container. Table II shows the average QoE metrics for the IP-only case and the NDN/IP case. For a fair comparison with cache-less IP, the cache size at the NDN routers was set to `0MB` in this experiment. As can be seen from the table,

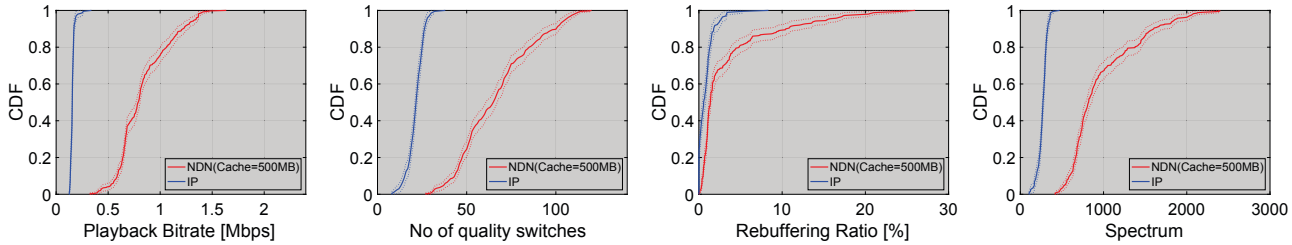

Fig. 6: Cumulative Distribution Functions for QoE metrics for the case of 40 NDN clients and 20 IP clients.

NDN live streaming performs better with respect to average bitrate, even if no caching is performed. NDN uses the Pending Interest Table (PIT) to aggregate all duplicate requests that are temporally close. When data comes back, all requester receives the same copy of the data, creating an inherent multicast mechanism. We attribute the improved performance to this inherent multicast characteristics that complement live streaming scenarios well. However, this bitrate enhancement is accompanied with increased #QS, Spectrum & Rebuffering Ratio. We address the cause and solution to this drawback in the next section.

B. Live Streaming with NDN vs. IP

This section first analyzes the benefits of using NDN for live video streaming over IP due to its in-network caching capabilities. We further study the effect of increasing the cache size of NDN clients with respect to QoE. After discussing the benefits, we analyze the drawbacks of using NDN in correlation to live video streaming and suggest possible solutions. Finally, we show that with our suggested improvements, NDN outperforms IP across all QoE metrics with server-side as well as client-side bottlenecks.

1) Higher bitrate & bandwidth utilization by NDN

NDN clients report higher average playback bitrate across all experiments as can be observed from the corresponding column in Table I and their respective CDFs (Figures 4, 5, 6). While throughput for NDN traffic is limited to `5Mbps` (50% of the `10Mbps` link between server and Router 1) in experiments I & IIa, the combined average bitrate is almost double (4*2.48=`10Mbps`) or higher (20 * 0.67=`13.4Mbps`), respectively. Similar conclusions can be drawn from experiment IIb results. As cumulative avg. bitrate increases with increasing clients, this confirms our hypothesis that NDN benefits live streaming scenarios due to its inherent in-network caching and improved handling of potential NDN segment retransmissions. In comparison, the cumulative average bandwidth for IP clients for both experiments stays slightly below (4*1.18=`4.72Mbps`, 20*0.24=`4.8Mbps`) the allotted `5Mbps` on the link between server and Router 1.

Motivated by NDN consistently reporting higher avg. bitrate, we also report bandwidth utilization by NDN versus IP streaming clients. If we define bandwidth utilization as the percentage of average playback bitrate per client for bottleneck bandwidth allotted per client, then NDN utilizes up to 505% of the available bandwidth (40*0.84=33.6Mbps out of 6.67Mbps) whereas IP only utilizes up to 96% (20*0.16=3.2Mbps out of 3.33Mbps) of it in the best case scenario (computed from experiment IIb results). Thus, indicating NDN's superior bandwidth utilization over IP across all scales. This clearly demonstrates the scalability and the benefits of using NDN for live streaming applications.

TABLE I: Average QoE metric results from streaming sessions accross different experiment setup

Scenario	Rebuf %	#QS	Avg. bitrate	Spectrum
Exp I: 4 NDN, 4 IP, 50/50 BW split, 500MB Cache				
NDN	0.1	43.7	2.48	1220.0
IP	0.0	33.3	1.18	637.9
Exp IIa: 20 NDN, 20 IP, 50/50 BW split, 0MB Cache				
NDN	1.98	54.6	0.40	745.8
IP	0.1	24.0	0.24	304.1
20 NDN, 20 IP, 50/50 BW split, 250MB Cache				
NDN	2.07	73.3	0.75	977.6
IP	0.15	23.8	0.24	302.2
20 NDN, 20 IP, 50/50 BW split, 500MB Cache				
NDN	2.5	78.0	0.67	1077.5
IP	0.12	22.2	0.24	287.4
Exp IIb: 40 NDN, 20 IP, 66/33 BW split, 500MB Cache				
NDN	3.47	66.8	0.84	979.1
IP	0.3	21.8	0.16	268.6

TABLE II: Average QoE for IP-only and NDN/IP case.

Scenario	Rebuf %	#QS	Avg. bitrate	Spectrum
20 IP, 20 IP, 50/50BW split				
IP	0.37	22.62	0.24	292.16
20 NDN, 20 IP, 50/50 BW split, 0MB Cache				
NDN	1.04	39.31	0.32	524.97

2) Effect of caching

In this section, we present the effect of varying cache sizes for NDN clients in large-scale scenario (experiment IIa). Comparing the results for the three different cache-sizes, we make three major observations. First, increasing the cache size beyond `500MB` does not lead to a further increase in QoE. Even the increase from `250MB` to `500MB` results in marginal improvement of the QoE metrics. Second, the rebuffering percentage is almost identical for all three cases (see Figure 5), but slightly increasing #QS indicating interdependence between these QoE metrics. Third, as shown in Figure 5, the #QS metric is lowest for the 0MB cache and increases for the `250MB` and `500MB` cache cases (with negligible differences between the two cases). Compared to IP, the #QS metric is higher for all three caching scenarios

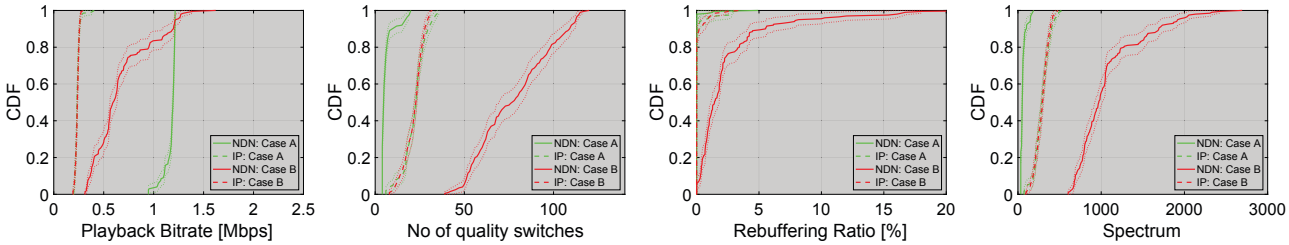

Fig. 7: Cumulative Distribution Functions for QoE metrics for the case of limited playback bitrates. Case A: NDN chooses from 4 qualities & IP from 14. Case B: Both NDN & IP choose from 14 qualities

Layer 3 protocol	Rebuff%	#QS	ABR	Spectrum	Hit-Rate at Router1	Hit-Rate at Router2	Hit-Rate at Router3
NDN (14-qualities)	2.5	78.0	0.67	1077.5	0.05	0.12	0.11
IP	0.12	22.2	0.24	287.4	NA	NA	NA
NDN (4-qualities)	0.05	6.5	1.17	62.26	0.19	0.30	0.36
IP	0.13	22.62	0.24	288.98	NA	NA	NA

TABLE III: Avg. QoE and hit-miss ratio at respective caches for Livestreaming on NDN with 14 qualities vs. 4 qualities

and the difference to NDN is larger than in the small-scale scenario. From the large-scale scenarios in Table I we observe that #QS goes up in NDN and down in IP. In the next section, we explain the reason behind this with a hypothesis as well as proposed solutions.

3) Effect of Oscillation effect on Rebuffering, #QS & Spectrum: Causes & Solution

Examining the additional QoE parameters shows that the increased playback bitrate in the NDN case comes with the trade-off of increases in quality switches and spectrum. While the rebuffering percentage stays comparable to the IP scenario in small-scale results (experiment I), it increases in comparison with IP for large-scale cases (experiment IIa & b). This increase might have a negative impact on the viewer's QoE that can outweigh the positive impact of an increased playback bitrate. Hence, in this section we try to find the underlying cause and plausible solutions to this observed problem.

Cause/Hypothesis: First, higher #QS in NDN as opposed to IP can in part be explained by examining the average bandwidth available to each IP client on the bottleneck link. In the case of 20 IP clients the average bandwidth is `0.25Mbps` per client. This bandwidth is only sufficient to stream the lowest 4 out of 14 playback bitrates (`0.08Mbps, 0.13Mbps, 0.17Mbps, and 0.22Mbps`). For NDN live streaming with `250MB` cache size, the average bitrate is `0.75Mbps`, which is sufficient to stream the 7 lower playback bitrates. This clearly demonstrates higher opportunities for bitrate quality changes in NDN as opposed to IP. Second, we conjecture that some of the decreases in QoE are caused by the interplay of ABR streaming and NDN. As presented by Grandl et al. [9], the potentially random placement of video segment on either the server or the caches can lead to so-called "oscillation effects". For example, if a client receives a video segment in low quality from a cache, the measured download rate might be high. Based on this observation, the ABR algorithm at the client (BOLA [42]) might decide to request the next segment in a higher quality. If this segment is currently not stored at the cache, the client has to retrieve the segment from the server and most likely experiences a lower download rate. This results in the client requesting the next segment at a (much) lower quality. This alternate retrieval of segments from server or cache can happen several times during a streaming session leading to increased #QS and spectrum in the case of NDN. Furthermore, these suspected oscillation effects caused by potential low hit rates on the caches lead to lower download rates and hence the higher rebuffering percentages. This led us to further investigate the effect of available bitrates on cache hit-ratios causing potential "oscillation effects" affecting QoE.

Confirmation: To gain more detailed correlation between higher available bitrates (in the case of NDN), the oscillation effect caused by low cache hit-rates and the resulting QoE (#QS, spectrum & Rebuffering percentage), we first analyze the hit rate (per NDN segment) on the three routers (1-3) used in the topology for these experiments. In an additional experiment for the 20NDN/20IP client case, we set the cache sizes on all three routers to 2GB (large enough to cache all DASH segment in all playback bitrates of the video which totals to 1.8GB). We observed that the hit rates, surprisingly, do not increase with an increase in cache size and are consistently low. Reported hit-rates were 0.07,0.11 and 0.14 for `250MB` caches, 0.05,0.12 and 0.11 for `500MB` caches and 0.06,0.15 and 0.15 for `2GB` caches in Router1, Router2, and Router3, respectively. The most probable reason behind low hit rates would be that all 20 NDN clients request a very disjunct set of qualities for individual DASH video segments (keep in mind that every segment is available in fourteen different bitrates (see Sect. III-A)). Our hypothesis is further strengthened by the observation that with a maximum of fourteen qualities available, NDN clients requested up to seven different qualities for a given DASH segment and an average of five different qualities for all DASH segments. This seemingly high variation in requested qualities results in low hit rates. With ten clients connected to Routers 2 and 3 each, almost every client requested a different quality per DASH segment.

Solution: Based on these observations, we reduced the avail-

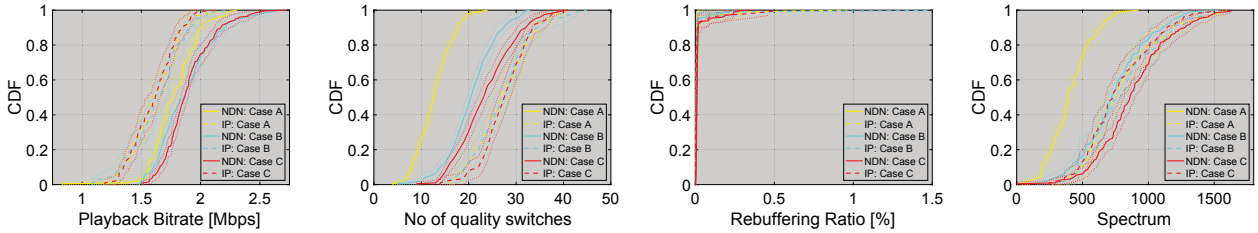

Fig. 8: Cumulative Distribution Functions for QoE metrics with client-side bottleneck. Case A: NDN chooses from 4 qualities & IP from 14. Case B: Both NDN & IP choose from 14 qualities. Case C: Case B with 1% packet loss

able playback bitrates from fourteen to four (`0.08Mbps, 0.22Mbps, 0.79Mbps, and 1.24Mbps`) for NDN live streaming and conducted an experiment with 20 NDN and 20 IP clients (similar to Experiment IIa in Sect. IV) and cache sizes set to `500MB` at the routers. The IP clients are still able to select from all fourteen playback bitrates. We observe that the number of distinct qualities were much lower. Previously with fourteen bitrates, *mode* for the distinct qualities requested per DASH segment was five and a *maximum* of seven distinct qualities. But with four available bitrates, the *mode* reduces to one with a *maximum* of three distinct qualities requested by all NDN clients. More importantly, this resulted in higher hit rates as reported in Table III. Figures 7 and Table III further confirm our hypothesis that the reduction of available qualities has a significant impact on QoE. From the CDF graphs in Figure 7, we observe that when NDN chooses from 4 qualities, it results in higher playback bitrate, lower quality switches, lower rebuffering percentage and spectrum as compared to when NDN chooses from more available qualities(In our case, 14). Furthermore, when NDN clients choose from lower available bitrates, it outperforms IP clients (as can be seen from the CDF grpah both in Cases A & B) in terms of QoE. Additionally from Table III, the comparison of NDN with fourteen available qualities to the case with only four available qualities shows that #QS is significantly reduced while the playback bitrate is increased for the four-quality case. A similar improvement can be observed for the rebuffering ratio and the spectrum as an effect of reduction in the #QS. These results conclusively show that with suggested modifications, NDN outperforms IP across all QoE metrics.

Clearly, the selection of the playback bitrate for NDN live streaming was informed by the results presented in Sect. IV-B1 and Table I. With an average bitrate of `0.67Mbps` in the case of `500MB` cache size, selecting playback bitrates was straight-forward. To make such an approach feasible for an actual system, an approach could be implemented that collects the average client bandwidth and adapts the playback bitrates that are advertised in the MPD file once a sufficient amount of data has been collected.

4) NDN vs. IP with client-side bottleneck

Our motivation to enforce a server-side bottleneck so far was to observe how the architecture responds to reduced midgress traffic. Obviously, the last hop to the client can also be a bottleneck (especially in mobile scenarios). Hence, a comparative QoE analysis of NDN and IP clients streaming over a network with client-side bottleneck is also required to give a more wholesome idea about NDN's superior performance with live-streaming. To further study such a scenario, we increased the bandwidth of the server side link (see Figure 2) to 1Gbps and added a client-side bottleneck of 10Mbps. For equal distribution of resources and a fair comparison, 5 clients were run on each node (both IP and NDN) and all nodes were connected to routers 2 & 3 with 10 Mbps links. As Figure 8 shows, NDN outperforms IP across all QoE metrics with 4 as well as 14 qualities. Even though the caching is limited with this client-side bottleneck, NDN still benefits from it as well as from its inherent nature of multicast and retransmissions to the nearest cache. It is also interesting to note that contrary to the results shown in Figure 7, QoE for 14 qualities is more comparable to the one with 4 qualities (0.08Mbps, 0.79Mbps, 1.55Mbps, and 2.48 Mbps) around an ideal average of 2Mbps. In this case, the bottleneck at the last hop dampens the oscillation effect. The bitrate is slightly increased in the case of 14 qualities because of more available qualities which in turn slightly worsens #QS, spectrum, and rebuffering percentage. In the event of loss, it is intuitive that the QoE will be lower than in the lossless cases. However, Figure 8 shows that NDN still outperforms IP with respect to all QoE metrics besides the spectrum due to increased magnitude of variability with 14 qualities.

V. Related Work

A. CDNs and NDN

To accomplish large-scale live streaming, CDN providers need to *work around* several limitations of the TCP/IP architecture such as lack of native IP multicast support, lack of support for caching at the network layer, and more. NDN, on the other hand, provides several desirable properties for live streaming such as in-network caching, multicast, and failure resiliency. These properties can be utilized for live-streaming - both inside a CDN infrastructure and user-based live streaming. A CDN can certainly deploy several NDN based live-streaming servers inside its infrastructure that will work in parallel with existing IP-based streaming mechanisms. Previous work by Ghasemi et al. [8] compared an NDN based video streaming solution deployed on the NDN Testbed with

two well-known CDNs, Akamai and Fastly. While this work did not attempt a hybrid solution, it showed NDN can reduce origin workload compared to traditional CDNs. Another work by Thelagathoti et al. [45] demonstrated that NDN deployment inside a CDN infrastructure will help with efficient data retrieval and improve QoE.

B. SDN, NFV, and NDN

Several works have highlighted the benefits of an architecture that integrates SDN with NFV [5], [22]. For instance, the agility this integration provides to infrastructure and network service design can be very desirable for any dynamic and scalable architectural framework [28]. NFV and SDN combined have revolutionized network architectures that are able to cope with the continuous growth in data-traffic [26]. They provide the ability to virtualize any network infrastructure based on its requirements. Hence, the decision to use our SDN-NFV setup. There has been work on SDN-NFV infrastructures that handle heterogeneous network technologies (e.g., [20], [46], [47]) but none of them compare multiple network stacks and their performance. In [19], Mai et al. have implemented NDN technologies with SDN-NFV support but the novelty of our work lies in the heterogeneity of the network protocol stacks as implemented in [43]. Performance analysis over IP versus non-IP protocols [15] or specifically IP versus NDN protocols [36], [38] have been executed before but not with the design flexibility that comes with the benefits of SDN programmability [43]. Kanada et al. [15] use virtual link tunnels to encapsulate IP and non-IP packets whereas Satria et al. [38] evaluate them separately and not in the context of video streaming in a non-virtualized setup.

Several works have proposed translation between TCP/IP and NDN so that they can coexist. Moiseenko et al. proposed TCP over ICN where TCP traffic is converted into ICN traffic [25]. Shannigrahi et al. proposed IPoC [40] where TCP/IP traffic is encapsulated into NDN packets for transport. Refaei et al. proposed an IP-ICN gateway that allows IP client-server communication to operate seamlessly through an NDN cloud [32]. Nour et al. [27] propose an approach that uses NFV for ICN/IP hybrid routers that require predefining a set of regions. On the contrary, we utilize layer 2 header information to indicate different types of traffic that allows us to differentiate between IP and NDN traffic in real-time and decide whether to forward it to IP or NDN data source.

Several papers [18], [24], [34], [35] motivate the idea behind using the NDN protocol for live video streaming. Mohammed et al. [24] and Li et al. [18] realize NDN's superiority as compared to IP in live-streaming over wireless networks using WiFi direct and WiFi's built-in broadcast mechanism, respectively. Samain et al. [36] and Rainer et al. [31] make similar comparison of the dynamic adaptive streaming performance on IP over NDN as shown in [43]. While Samain et al. [36] experiments with different modes of NDN under wireless loss detection recovery and load balancing state, our work aims to provide the best user viewing experience leveraging native NDN. Although on a different hybrid setup that runs both the protocol stacks simultaneously, we also compare the performance metrics of IP and NDN with respect to video streaming. Our main motivation is to use this comparison to build an architecture directed towards an optimal hybrid video-streaming experience not explored before.

VI. Conclusion & Future Work

In this paper, we propose and show the benefits of a hybrid and flexible streaming approach which supports multiple internet architectures over a traditional IP approach for improved QoE in live adaptive bitrate video streaming applications. To achieve this goal our approach employs network and containerization techniques. We present a detailed description to demonstrate how SDN, NFV, kernel virtualization and containerization can be orchestrated to provide a hybrid and highly scalable streaming architecture. We implement this setup in the CloudLab testbed and perform an extensive performance evaluation of our hybrid streaming approach. The evaluation results demonstrate the profit in terms of average bitrate and bandwidth utilization from our approach and reveal that live streaming can be performed efficiently, is scalable, and provides good QoE with the help of NDN. Using containers for the NDN streaming clients provides a method that can activate such clients without end user involvement. Counter-intuitive to experience gained in the case of ABR streaming over TCP/IP, we show that a reduced set of available playback bitrates leads to better performance in the case of NDN-based live streaming and outperforms IP-based live streaming under both server and client-bottleneck conditions. We also show that NDN-based live streaming behaves fairly to IP-based session and does not negatively impact the QoE of these sessions. In future work we plan to to develop new ABR algorithms that are cognizant that NDN provides in-network caching. We will also study if the approach of caching at the level of NDN Segments is appropriate in the case of ABR streaming.

Acknowledgment

This work was funded by National Science Foundation (NSF) grants CNS 19-01137, OAC-2019163, OAC-2126148, and OAC-2019012. All opinions and statements in the above publication are of the authors and do not represent NSF positions.

References

[1] S. H. Ahmed, S. H. Bouk, and D. Kim. *Content-Centric Networks: An Overview, Applications and Research Challenges*. Springer Publishing Company, Incorporated, 1st edition, 2016.

[2] T. Anderson, K. Birman, R. Broberg, M. Caesar, D. Comer, C. Cotton, M. J. Freedman, A. Haeberlen, Z. G. Ives, A. Krishnamurthy, et al. The nebula future internet architecture. In *The Future Internet Assembly*, pages 16–26. Springer, 2013.

[3] Docker. Docker. "https://www.docker.com/", 2020.

[4] Docker. Macvlan networks. "https://docs.docker.com/network/macvlan/", 2020.

[5] Q. Duan, N. Ansari, and M. Toy. Software-defined network virtualization: an architectural framework for integrating sdn and nfv for service provisioning in future networks. *IEEE Network*, 30(5):10–16, 2016.

[6] D. Duplyakin, R. Ricci, A. Maricq, G. Wong, J. Duerig, E. Eide, L. Stoller, M. Hibler, D. Johnson, K. Webb, A. Akella, K. Wang, G. Ricart, L. Landweber, C. Elliott, M. Zink, E. Cecchet, S. Kar, and

P. Mishra. The design and operation of cloudlab. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 1–14, Renton, WA, July 2019. USENIX Association.
[7] L. Foundation. tc. "https://linux.die.net/man/8/tc", 2020.
[8] C. Ghasemi, H. Yousefi, and B. Zhang. Far cry: Will cdns hear ndn's call? In *Proceedings of the 7th ACM Conference on Information-Centric Networking*, ICN '20, page 89–98, New York, NY, USA, 2020. Association for Computing Machinery.
[9] R. Grandl, K. Su, and C. Westphal. On the interaction of adaptive video streaming with content-centric networking. In *2013 20th International Packet Video Workshop*, pages 1–8, 2013.
[10] P. Gusev, Z. Wang, J. Burke, L. Zhang, T. Yoneda, R. Ohnishi, and E. Muramoto. Real-time streaming data delivery over named data networking. *IEICE Transactions on Communications*, 99(5):974–991, 2016.
[11] S. Ha, I. Rhee, and L. Xu. Cubic: A new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74, July 2008.
[12] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown. Reproducible network experiments using container-based emulation. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, page 253–264, New York, NY, USA, 2012. Association for Computing Machinery.
[13] ITEC. Big buck bunny video. "http://www-itec.uni-klu.ac.at/ftp/datasets/DASHDataset2014/BigBuckBunny/2sec/", 2020.
[14] P. Juluri. Astream. "https://github.com/pari685/AStrea", 2020.
[15] Y. Kanada and A. Nakao. Development of a scalable non-ip/non-ethernet protocol with learning-based forwarding method. In *2012 World Telecommunications Congress*, pages 1–6, 2012.
[16] Z. Kong, X. Ma, Y. Zhang, Z. Zhang, D. Pesavento, S. Shannigrahi, S. Dulal, and J. Shi. ndn-python-repo. "https://github.com/UCLA-IRL/ndn-python-repo", 2020.
[17] LBL. Singularity. "https://sylabs.io/docs/", 2020.
[18] M. Li, D. Pei, X. Zhang, B. Zhang, and K. Xu. Ndn live video broadcasting over wireless lan. In *2015 24th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–7, 2015.
[19] H. L. Mai, M. Aouadj, G. Doyen, W. Mallouli, E. M. de Oca, and O. Festor. Toward content-oriented orchestration: Sdn and nfv as enabling technologies for ndn. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 594–598, 2019.
[20] R. Martínez, A. Mayoral, R. Vilalta, R. Casellas, R. Muñoz, S. Pachnicke, T. Szyrkowiec, and A. Autenrieth. Integrated sdn/nfv orchestration for the dynamic deployment of mobile virtual backhaul networks over a multilayer (packet/optical) aggregation infrastructure. *IEEE/OSA Journal of Optical Communications and Networking*, 9(2):A135–A142, 2017.
[21] S. Mastorakis, A. Afanasyev, and L. Zhang. On the evolution of ndnsim: An open-source simulator for ndn experimentation. *SIGCOMM Comput. Commun. Rev.*, 47(3):19–33, Sept. 2017.
[22] J. Matias, J. Garay, N. Toledo, J. Unzilla, and E. Jacob. Toward an sdn-enabled nfv architecture. *IEEE Communications Magazine*, 53(4):187–193, 2015.
[23] D. Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.
[24] S. Mohammed and M. Xie. A measurement study on media streaming over wi-fi in named data networking. In *2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems*, pages 543–548, 2015.
[25] I. Moiseenko and D. Oran. Tcp/icn: Carrying tcp over content centric and named data networks. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, ACM-ICN '16, page 112–121, New York, NY, USA, 2016. Association for Computing Machinery.
[26] V. Nguyen, A. Brunstrom, K. Grinnemo, and J. Taheri. Sdn/nfv-based mobile packet core network architectures: A survey. *IEEE Communications Surveys Tutorials*, 19(3):1567–1602, 2017.
[27] B. Nour, F. Li, H. Khelifi, H. Moungla, and A. Ksentini. Coexistence of icn and ip networks: an nfv as a service approach. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2019.
[28] N. Omnes, M. Bouillon, G. Fromentoux, and O. L. Grand. A programmable and virtualized network it infrastructure for the internet of things: How can nfv sdn help for facing the upcoming challenges. In *2015 18th International Conference on Intelligence in Next Generation Networks*, pages 64–69, 2015.
[29] R. Pauly, C. Ogle, C. Mcknight, D. Reddick, J. Presley, S. Shannigrahi, and A. Feltus. NDN-TR68: Utilizing NDN for Domain Science Applications - a Genomics Example - Named Data Networking (NDN), Mar 2021. [Online; accessed 8. Mar. 2021].
[30] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado. The design and implementation of open vswitch. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, NSDI'15, page 117–130, USA, 2015. USENIX Association.
[31] B. Rainer, D. Posch, and H. Hellwagner. Investigating the performance of pull-based dynamic adaptive streaming in ndn. *IEEE Journal on Selected Areas in Communications*, 34(8):2130–2140, 2016.
[32] T. Refaei, J. Ma, S. Ha, and S. Liu. Integrating ip and ndn through an extensible ip-ndn gateway. In *Proceedings of the 4th ACM conference on information-centric networking*, pages 224–225, 2017.
[33] Y. Ren, J. Li, S. Shi, L. Li, G. Wang, and B. Zhang. Congestion control in named data networking – a survey. *Computer Communications*, 86:1–11, 2016.
[34] F. B. Rukmana and R. F. Sari. Performance evaluation of video streaming application via named data network (ndn). In *2019 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)*, pages 141–144, 2019.
[35] J. Saltarin, E. Bourtsoulatze, N. Thomos, and T. Braun. Adaptive video streaming with network coding enabled named data networking. *IEEE Transactions on Multimedia*, 19(10):2182–2196, 2017.
[36] J. Samain, G. Carofiglio, L. Muscariello, M. Papalini, M. Sardara, M. Tortelli, and D. Rossi. Dynamic adaptive video streaming: Towards a systematic comparison of icn and tcp/ip. *IEEE Transactions on Multimedia*, 19(10):2166–2181, 2017.
[37] J. Samain, G. Carofiglio, L. Muscariello, M. Papalini, M. Sardara, M. Tortelli, and D. Rossi. Dynamic adaptive video streaming: Towards a systematic comparison of icn and tcp/ip. *IEEE Transactions on Multimedia*, 19(10):2166–2181, 2017.
[38] M. N. D. Satria, F. H. Ilma, and N. R. Syambas. Performance comparison of named data networking and ip-based networking in palapa ring network. In *2017 3rd International Conference on Wireless and Telematics (ICWT)*, pages 43–48, 2017.
[39] I. Seskar, K. Nagaraja, S. Nelson, and D. Raychaudhuri. Mobilityfirst future internet architecture project. In *Proceedings of the 7th Asian Internet Engineering Conference*, pages 1–3, 2011.
[40] S. Shannigrahi, C. Fan, and G. White. Bridging the icn deployment gap with ipoc: An ip-over-icn protocol for 5g networks. In *Proceedings of the 2018 Workshop on Networking for Emerging Applications and Technologies*, pages 1–7, 2018.
[41] K. Spiteri, R. Sitaraman, and D. Sparacio. From theory to practice: Improving bitrate adaptation in the dash reference player. *ACM Trans. Multimedia Comput. Commun. Appl.*, 15(2s), July 2019.
[42] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. Bola: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, 2016.
[43] B. Suresh, D. Bhat, and M. Zink. An evaluation of sdn and nfv support for parallel, alternative protocol stack operations. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7, 2018.
[44] I. Technical Report. Cisco Systems. Cisco. 2020. cisco visual networking index (vni) complete forecast update. 2017 - 2022.
[45] R. K. Thelagathoti, S. Mastorakis, A. Shah, H. Bedi, and S. Shannigrahi. Named data networking for content delivery network workflows. In *2020 IEEE 9th International Conference on Cloud Networking (CloudNet)*, pages 1–7. 10.1109/CloudNet51028.2020.9335806, 2020.
[46] R. Vilalta, A. Mayoral, R. Casellas, R. Martínez, and R. Muñoz. Experimental demonstration of distributed multi-tenant cloud/fog and heterogeneous sdn/nfv orchestration for 5g services. In *2016 European Conference on Networks and Communications (EuCNC)*, pages 52–56, 2016.
[47] R. Vilalta, A. Mayoral, R. Muñoz, R. Casellas, and R. Martínez. Multitenant transport networks with sdn/nfv. *Journal of Lightwave Technology*, 34(6):1509–1515, 2016.
[48] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang. Named data networking. *SIGCOMM Comput. Commun. Rev.*, 44(3):66–73, July 2014.
[49] M. Zink, J. Schmitt, and R. Steinmetz. Layer-encoded video in scalable adaptive streaming. *IEEE Transactions on Multimedia*, 7(1):75–84, 2005.