Wireless Network Emulation for Research on Information-Centric Networking

Philipp Moll, Sebastian Theuermann, Hermann Hellwagner
Institute of Information Technology
Alpen-Adria-Universität Klagenfurt
firstname.lastname@aau.at

ABSTRACT

When developing new approaches in networking research, one of the most important requirements is to evaluate the degree of improvement of a new approach both realistically and cost-effectively. Wireless networks and their adequate emulation play an important role in evaluation, but emulation of wireless links and networks is still difficult to handle. In this paper, we present a low-cost, fixed-network testbed able to emulate the dynamically changing conditions of wireless links caused by client mobility and physical phenomena. We extend the existing fixed-network testbed for the purpose of wireless network emulation using the Linux tools *tc*, *iptables*, and *NetEm* in sophisticated ways. Convenient function blocks are provided to configure wireless network topologies as well as dynamic link and mobility conditions to be emulated with modest efforts. We utilize the testbed's capabilities to investigate the influence of different mobility models on streaming SVC-encoded videos in Named Data Networking (NDN), a novel Information-Centric Networking architecture. Furthermore, we evaluate the benefits of using early loss detection mechanisms for streaming in NDN, by implementing Wireless Loss Detection and Recovery (WLDR). Our results show that the extended fixed-network testbed can precisely emulate wireless network conditions and usage. For instance, the emulation revealed that both the choice of the mobility model and the use of WLDR have a substantial influence on the resulting SVC video streaming performance.

CCS CONCEPTS

• **Networks → Network performance evaluation**; *Network architectures*; Physical links; Wireless access networks;

KEYWORDS

Wireless Link Emulation; Networking Testbed; Information-Centric Networking

ACM Reference Format:
Philipp Moll, Sebastian Theuermann, Hermann Hellwagner. 2018. Wireless Network Emulation for Research on Information-Centric Networking. In *12th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization (WiNTECH '18), November 2, 2018, New Delhi, India.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3267204.3267211

1 INTRODUCTION

The global Internet traffic increases year by year and no end of this trend is foreseeable. This development triggers research on improvements of our current Internet architecture, but also on the development of new networking architectures. In both cases, it is important to analyze newly developed approaches and to check if they are suited to be used on the Internet.

For analyzing newly developed approaches, multiple options are available. One option is *theoretical analysis*, which can be used to verify the correctness of an approach or to calculate its theoretical performance. The results of such a theoretical analysis can be validated by event-based network *simulations*, which enrich the theoretical analysis by adding the basic characteristics and influences of computer networks. Nevertheless, event-based simulations only include abstractions of the real world and do not consider underlying physical phenomena or hardware restrictions. A network *emulation* is conducted with real devices and communication over real networks and is thereby a valuable addition to simulation.

Wireless links are an essential part of today's networks. According to Cisco's Visual Networking Index[1], 43% of all networked devices will be mobile-connected by 2021. Therefore, an option to emulate wireless links is required, but difficult to realize because the performance of wireless links depends on various factors, including, but not limited to, reflections, interference, and clients' movements. In this paper,

[1]https://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html, last visited: 2018-06-25

we extend an existing emulation testbed with the aim to be able to emulate dynamic link properties and thereby enable the testbed to be used for wireless network emulation.

Another ongoing trend is the change of the Internet's usage from host-to-host communication to content distribution, which is also reflected by current developments in networking research. Research on Information-Centric Networking (ICN) aims to better support this content-based communication. In Named Data Networking (NDN) [14], a promising ICN architecture, each piece of information is identifiable by a system-wide unique name. Data is requested by emitting an *Interest* packet, carrying the name of the desired information. When the Interest reaches a node which holds a copy of the information, the information is packaged into a *Data* packet and sent back to the requester on the reverse path of the Interest.

More and more real-world applications for NDN are developed, but a proper way to test these applications in wireless networks is still missing. Our testbed provides the possibility to run real-world NDN applications and to test them in arbitrary networks. From previous work [8, 10], we already know that NDN is well suited for adaptive video streaming in fixed network environments. We assume that mobility has a strong impact on streaming performance, which is why we now use the presented testbed to analyze the influence of mobility on video streaming quality.

The main contribution of this paper is the presentation of our open-source wireless network emulation testbed and a detailed overview of the function and usage of the testbed. We further analyze the capabilities of the Linux tools *tc* and *NetEm* and show their suitability to emulate wireless links. Finally, the influence of mobility patterns on the performance of video streaming in ICN networks is shown. In addition, we improve the streaming quality in lossy environments by including mechanisms such as *Wireless Loss Detection and Recovery* (WLDR) [4] and discuss the results.

The remainder of this paper is structured as follows. Section 2 discusses related testbeds. In Section 3 an overview of our testbed's architecture is presented. The approach used for emulating wireless links is described in Section 4. In Section 5 we use our testbed to evaluate the benefits of WLDR when considering different movement patterns of multimedia streaming clients in wireless networks. Finally, we conclude with the findings of the paper in Section 6.

2 RELATED WORK

The evaluation of new networking approaches is generally done by employing one or multiple of the following methods:

Simulations using event-based simulators are rather general tools and allow for testing new ideas without side effects resulting from physical phenomena or hardware restrictions. The ns3/ndnSIM network simulator [6] is an example of a commonly used simulation tool for NDN-related research.

Emulation using *virtualization* is a more practical method, where multiple sandboxed runtime environments, either realized as virtualized operating systems or Linux containers, are interconnected by an emulated network topology. Depending on the computational requirements, these environments can be started either on a single machine or on a server cluster, and can be used to test sample applications on arbitrary network topologies. The vICN toolchain [11] for research on Community ICN (CICN) as well as Mini-NDN[2] for NDN-based research are representatives of virtualization approaches. Although these approaches are closer to realistic scenarios than simulation, they suffer from the absence of a real network and real parallelization.

Emulation using a dedicated *testbed* is the most realistic evaluation method. This method utilizes real hardware as well as real network connections. The most prominent representative of an ICN testbed is the NDN testbed[3], where research institutions from all around the world participate in order to allow for experiments over the public Internet. When conducting experiments on this testbed, the ever-changing environment, resulting from changing traffic conditions on the underlying IP network, has to be kept in mind and complicates achieving reproducible results. A more controlled environment is ensured by the low-cost NDN testbed [9], which is a local testbed for network emulation purposes. It utilizes low-performance devices as network nodes, which can be interconnected by an easily configurable overlay network topology. Resulting from the use of real hardware, this testbed allows, besides the analysis of network behavior, insights into the computational complexity of evaluated algorithms and applications by CPU load and power consumption observations.

For the evaluation of wireless links, various testbeds utilizing real wireless channels are available. The NITOS facility[4], the Orbit testbed[5], and the PhantomNet infrastructure[6] are prominent examples of testbeds which allow to connect testbed nodes via real wireless links and thereby allow the optimization of the wireless channels proper, but also of applications using those channels. Although real wireless link properties can be studied, the access to these testbeds is limited in terms of time restrictions and limited control of testbed nodes.

Analyzing and optimizing applications' performance based on the characteristics of a wireless network does not inevitably require the use of real wireless channels and can

[2]https://github.com/named-data/mini-ndn, last accessed: 2018-05-23
[3]https://named-data.net/ndn-testbed/, last accessed: 2018-05-23
[4]https://nitlab.inf.uth.gr/NITlab/nitos, last accessed: 2018-08-07
[5]http://www.orbit-lab.org/, last accessed: 2018-08-07
[6]https://www.phantomnet.org/, last accessed: 2018-08-07

be replaced by more flexible emulated wireless channels in many cases. Although virtualization approaches provide solutions for wireless link emulation, currently no possibilities to emulate mobility are supported. This is why we aim at realizing wireless network emulation, including the effects of mobility, on real hardware. Besides providing a highly customizable and flexible testbed, we focus on a low-cost solution, allowing other researchers to easily set up their own testbed.

3 TESTBED ARCHITECTURE

The testbed's architecture, as visualized in Figure 1, can be structured into two main components, namely hardware components and the execution engine. The hardware of the testbed actually runs the experiments, while the execution engine allows for the easy configuration of experiments. In the following, details on the main components and their constituting parts are presented.

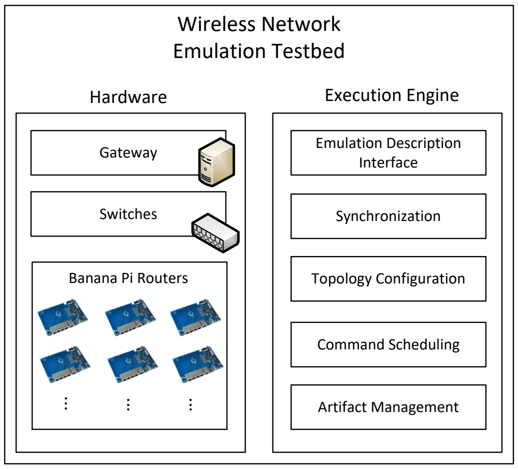

Figure 1: Conceptual design and main components of the wireless network emulation testbed.

3.1 Testbed Hardware

The testbed's hardware is based on the low-cost NDN testbed of [9]; that paper presents the structure and the configuration options of the testbed in detail. The purpose of the testbed is to provide a low-budget option to validate theoretical concepts and evaluate the performance of ICN on physical hardware. Besides upgraded software components (Section 3.2), the low-cost testbed is used as described in [9]. This section provides an overview of the testbed's hardware and explains the interplay of its components.

The testbed is based on an IP network interconnecting 20 network nodes, realized as *Banana Pi Routers* (BPI-R1). These low-performance devices are used to run the actual experiments and therefore operate the NDN network stack and all applications required for a specific experiment.

Each node is connected to two physically separated networks. First, the *management network*, which is mainly used for logging, artifact management and command scheduling. Second, the *emulation network*, which is used for the emulation traffic itself. This separation is needed in order to ensure that management activities do not influence the emulation activity. The emulation network, physically set up as a star topology, is converted into an arbitrary emulation network topology by using Linux tools, such as *traffic control* (tc), *iptables* and *Network Emulator* (NetEm). These tools allow to modify the emulation topology without changing the physical links between network nodes.

The *gateway server* is used for managing emulations. It provides all required artifacts, but also manages the execution of emulations and collects log files produced during the emulations. Furthermore, it provides a Web interface for observing the current state of the testbed nodes and network links.

3.2 Execution Engine

Besides the hardware components, the newly developed *Execution Engine* (EE) is the second important component of the testbed. It is used for the automated execution of network emulations, including all tasks for preparing the testbed and its nodes for the emulation process. In the following, a description of all components of the EE, including the interplay of the components is presented.

Emulation Description Interface. The Emulation Description Interface (EDI) provides an easy way to describe network emulations using the programming language Python. A network emulation is conducted by executing low-level system commands at predefined points in time on different nodes in the network. This includes configuring a network topology, but also starting applications and managing files.
The concept of the EDI's *function blocks* combines such lower-level tasks in self-contained containers, similar to the class concept in object-oriented programming. Thereby it becomes easily possible to define high-level tasks, such as setting up an emulated WiFi connection, with only a few lines of code.

Synchronization. In order to time emulation events, such as the start of an emulation run, but also to be able to calculate the delay between sending a packet on the sender side and receiving it on the receiver side, the system clocks of all devices have to be synchronized. Therefore, the *Network Time Protocol* (NTP) implementation *chrony*[7] is used. Investigations on the

[7]https://chrony.tuxfamily.org/, last accessed 2018-05-23

testbed showed that *chrony* is capable of keeping the maximum clock offset between two devices on the order of tens of microseconds.

Topology Configuration. The topology configuration component of the EE allows the easy deployment of arbitrary network topologies on top of the emulation network. In combination with function blocks of the EDI, it becomes possible to configure, e.g., random networks or predefined topologies, such as the Abilene Core topology [1]. In the background, this component uses the Linux tools *tc* and *iptables* to configure the desired network behavior.

Command Scheduling. The command scheduling module allows to execute commands at predefined times before and during emulation runs. Basically, command scheduling can be used to schedule two types of commands. First, *initialization commands* are executed to prepare the testbed for an emulation run. This preparation includes commands needed for the topology configuration as well as the distribution of configuration files required by various applications. Second, *emulation commands* are used to schedule commands during an emulation. These could effect starting or stopping an application, but also changing parameters of a link, e.g., initiating scheduled link failures.

Technically, the command scheduler works as follows. The EDI structures commands for each node into initialization and emulation commands and passes them to the corresponding nodes. The nodes then execute the initialization commands and thereby prepare the testbed for the next emulation run. After completion of the initialization phase, an emulation start time is set by the EE, which is then used as the reference time for all scheduled emulation commands.

Artifact Management. Applications, configuration files and log files can be seen as different artifacts which need to be managed during an emulation run. These various artifacts can be managed by using function blocks of the EDI. The idea of artifact management is that all required applications and configuration files are distributed to the testbed nodes in the initialization phase and automatically cleaned up after gathering log files, when the emulation run is completed.

4 WIRELESS LINK EMULATION

This section presents how the testbed is used to emulate wireless links. The goal of our work is to not only emulate static properties, such as a constant loss rate or link delay, but also to emulate changing network conditions, resulting, e.g., from moving network nodes. In reality, the characteristics of wireless links change continuously over time. For emulation purposes, this continuity is approximated by periodic adjustments of link properties.

The continuous change of a wireless link can be traced in two different ways. The first way is to model the properties of the wireless link by using well-established fading and shadowing models, such as Nakagami-lognormal channels [13], and by combining these models with mobility using models, such as presented in [3]. The second possibility is to use real-world data obtained in controlled experiments, such as provided by [7]. Independent of the researcher's preference, the flexibility of our system allows to use both possibilities. In the following, we model link characteristics by using the ns-3[8] implementation of Nakagami-lognormal channels in order to create network traces, which are then used as input for our emulation. Nevertheless, real-world traces can form the basis of an emulation as well.

The testbed uses the Linux tools *tc* and *NetEm*[9] to control the properties of emulated wireless links. In order to approximate the real continuous link changes as accurate as possible, the intervals between periodic adjustments need to be as small as possible, but larger as the time needed for configuring the adjustment. When making the intervals too small, we run the risk of overwhelming *tc* and *NetEm*, which might result in unwanted behavior. To prevent this, the capabilities of those Linux tools are analyzed; a representative part of the results from this analysis is presented in the following.

Figure 2 visualizes the evaluation results of the *switching lag* when switching the link delay of a single link between two testbed nodes. Switching lag is defined as the time elapsing between initiating a change of a parameter and the change becoming effective. In the case of modifying a link's delay, we define "becoming effective" such that the delays of five subsequent packets do not deviate more than 5% from the targeted delay. For evaluation, we used the delay values of the DASH Industry Forum (DASH-IF) test vector NP2j [2]. In the upper chart of Figure 2, the blue line (mostly covered) indicates the test vector; the green line visualizes the measured link delay between the testbed nodes. The results indicate that *NetEm* is capable of adjusting the link delay values accurately with only small anomalies. The lower chart of Figure 2 visualizes the average switching lag for adapting the link delay as defined by the test vector, including 95% confidence intervals (CI) resulting from 30 emulation runs. It can be seen that the lag for applying the changes lies consistently below 50 ms and fluctuates only little. In addition, evaluations for bandwidth shaping and packet loss variation

[8]https://www.nsnam.org/, last accessed 2018-05-25

[9]The *NetEm* tools are used for controlling link delay and packet loss characteristics. For bandwidth shaping, the *tc* token bucket filter is used.

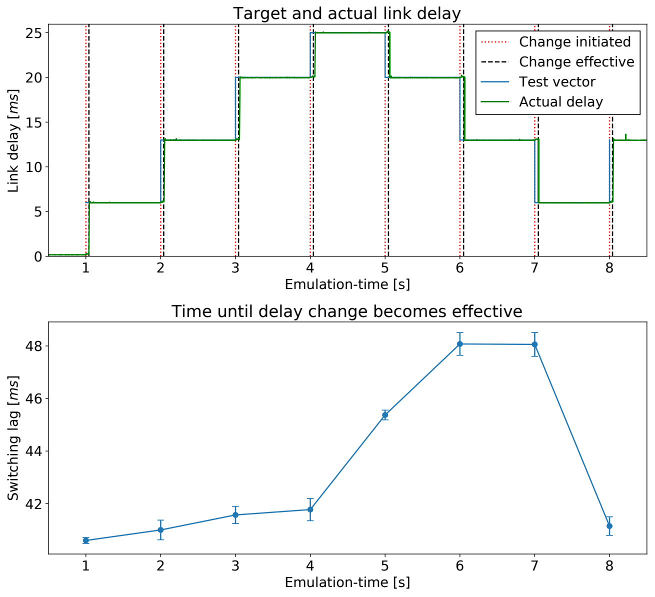

Figure 2: Evaluation of the switching lag for link delay variation using a DASH-IF test vector.

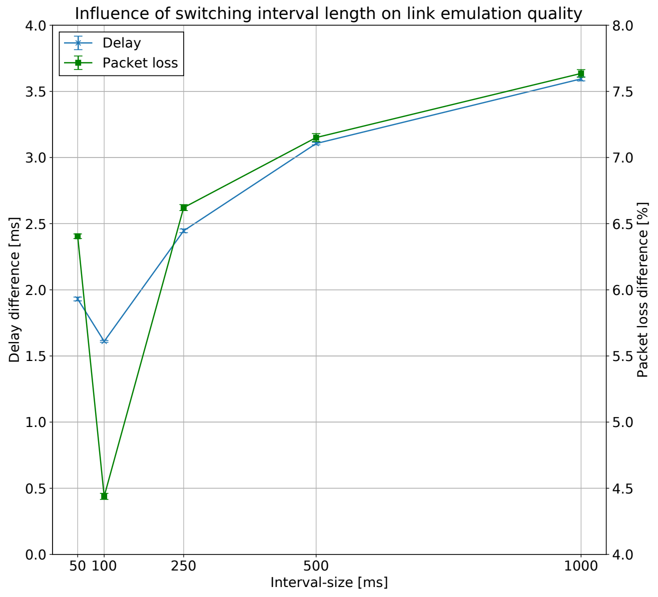

Figure 3: Visualization of emulation accuracy when emulating wireless links with varying switching intervals.

were conducted. The results of these evaluations are very similar to delay variation; results are published along with the source code of the testbed.

Having established that the switching lag for adjusting link conditions is around 50 ms, we are able to search for the optimal *switching interval* for emulating wireless links. As switching interval, we define the time between consecutive adjustments of link properties. The basis for this evaluation are network traces generated by the network simulator ns-3. The simulation scenario includes two nodes connected via a 802.11g WiFi link, one walking straight away from the other and back, while consuming the available bandwidth by sending UDP packets in both directions. Based on the simulation traces, values for packet loss, bandwidth and link delay are extracted and used as input for the wireless link emulation. Changes of a link's characteristics are applied once per switching interval. The single target value for a interval is calculated by averaging all measured delay, loss and bandwidth values in the corresponding switching interval.

Figure 3 visualizes the influence of the switching interval on delay and packet loss deviation. The deviation is calculated between the WiFi network trace from simulation and the traces generated by sending traffic over the emulated WiFi testbed link. The blue line visualizes the absolute delay difference in milliseconds, the green line the packet loss difference in percent. In general, we can see that shorter switching intervals lead to smaller deviations, as long as the switching interval is well above the switching lag. With a switching interval of 100 ms, a delay difference of less than 2 ms can be achieved. Regarding packet loss, a deviation of less than 5% is measured when setting the switching interval to 100 ms. Based on these results, we assume that the optimal switching interval for emulating wireless links is 100 ms, which is why this interval is used for further evaluations. Configuring a switching interval of less than 100 ms means that the adjusted property is effective for just a few milliseconds before it is adjusted again, which leads to fewer delivered packets with correct parameters and a higher deviation, as visible when focusing on the 50 ms switching intervals. Depicted error bars represent 95% CIs after 40 emulation runs.

Besides the calculated deviations visualized in Figure 3, we can examine the emulation precision visually in Figure 4. The emulation of the wireless link includes adjustments of the link delay and the packet loss rate. When packet loss is high, fewer packets arrive; when the delay is high, the transmitted packets arrive later at the receiver. Combining these characteristics already leads to a reduced bandwidth, without the need of shaping it separately. Figure 4 visualizes target and actual delay (upper chart) and packet loss (lower chart) values, measured during a single emulation run based on the previously explained simulated network trace. The emulated distance between the two nodes increases until second 50, where hardly any packets are transmitted successfully. From second 50 onwards, the distance between the nodes is reduced until second 100, which leads to reduced packet loss and delay values.

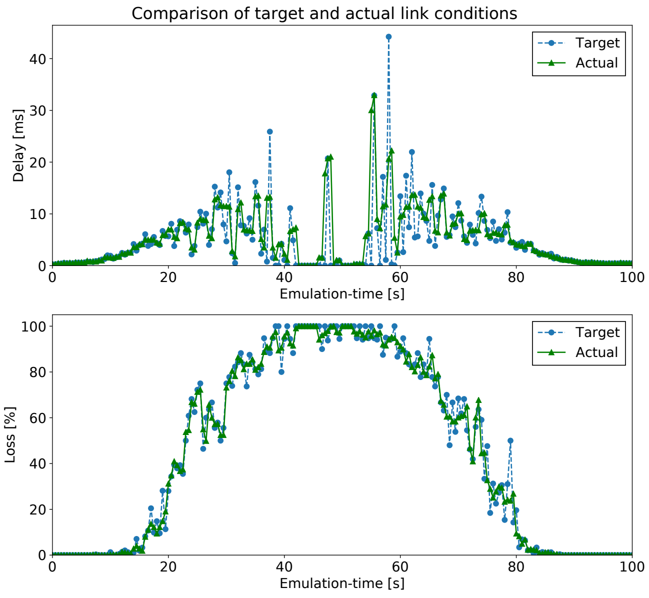

Figure 4: Progression of link delay and packet loss on an emulated WiFi link including a trivial mobility pattern.

The underlying mobility of the simulated network trace used for the presented evaluations may seem artificial, but besides the good suitability for demonstration purposes, the characteristics of this trace regarding the progression of packet loss and delay values follow the well-established model for Nakagami-lognormal channels [13]. Therefore, the results of this evaluation, showing that the testbed is capable of emulating wireless links accurately, become valid for other scenarios as well. Changing the emulation scenario is trivial, the only required step is to create a network trace by simulation or by conducting real-world measurements. Besides the emulation of channel-specific properties, the flexibility of this trace-driven approach also allows to emulate different types of handovers as well as interferences, provided that a given network trace includes such events.

5 ADAPTIVE VIDEO STREAMING OVER ICN IN WIRELESS NETWORKS

After having explained the structure of our testbed and the process of wireless link emulation in detail, in this section we demonstrate the testbed in action. We investigate the influence of mobility when streaming MPEG-DASH-compliant multimedia content [12] encoded with the scalable video codec (SVC). SVC allows to encode a video into multiple layers, including a base layer and multiple enhancement layers. The base layer provides a low-quality representation of the video. Enhancement layers can be stacked upon the base layer in order to improve the quality in terms of higher temporal resolution, spatial resolution, or visual quality (SNR).

Furthermore, we investigate the improvements achievable by the Wireless Loss Detection and Recovery (WLDR) technique [4] in this streaming scenario. WLDR is a technique used for the early detection of packet loss on wireless links. When requesting video segments in NDN, a video segment is split up into multiple Data packets, having the same name prefix and increasing sequence numbers. If e.g. Interest packets for sequence numbers 0–10 and an Interest packet for sequence number 12 are received on a WiFi access point, it is very likely that the Interest for sequence number 11 was lost. When using WLDR, the access point recognizes the gap in sequence numbers and forces the consumer to re-issue an Interest packet for the lost sequence number. The same principle is used for lost Data packets: when a gap in the sequence numbers of incoming Data packets is recognized, a retransmission of the lost Data packet is initiated. The benefits of WLDR for dynamic adaptive streaming without scalable codecs and static client positions was already identified by Samain et al. [10]. We now use the wireless network emulation testbed to analyze whether these benefits also hold true for scalable video codecs and analyze how different mobility models influence the performance of WLDR.

In Section 5.1, we show how the testbed, especially its execution engine, can be used to set up an evaluation scenario using predefined function blocks. Details on the evaluation scenario and results of our evaluation are given in Section 5.2.

5.1 Using the Testbed's Function Blocks

In this section, we describe how the testbed can be used to define an emulation scenario and to conduct evaluations. We use the video streaming scenario described in the previous section to showcase the testbed's capabilities.

The Emulation Description Interface (EDI) allows to use function blocks which combine logically connected low-level tasks to form a larger task. One example of a function block is the configuration of an emulated WiFi connection between two nodes. An emulated WiFi connection consists of two physical nodes, including an access point (AP), and a mobile client. The function block applies changes in the wireless link's performance caused by client mobility, by periodic adjustments of connection properties. Pre-recorded connection properties resulting from different mobility scenarios can be used to configure the wireless connection.

Besides the WiFi block, other function blocks are available, which together allow to define a complete emulation scenario. Figure 5 depicts the interplay of individual function blocks to define the structure of our video streaming scenario. In the following, this interplay is explained in greater detail.

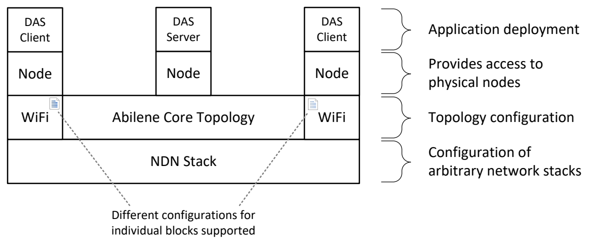
DAS Client
DAS Server
DAS Client
Node
Node
Node
WiFi
Abilene Core Topology
WiFi
NDN Stack
Application deployment
Provides access to physical nodes
Topology configuration
Configuration of arbitrary network stacks
Different configurations for individual blocks supported

Figure 5: Structure of an emulation topology for video streaming, created by combining function blocks.

The network topology is set up by combining two different types of function blocks. The *Abilene Core Topology* block is used to form the Abilene Core network [1] between eleven physical devices. The *WiFi* block connects two nodes via an emulated WiFi link. One of those nodes acts as an AP and gateway to the Abilene network. The second node represents a mobile client connected to the network via WiFi.

The *NDN Stack* function block can be used to manage the NDN stack on the devices of the emulation topology. Besides starting the required NDN services, forwarding information is disseminated to all nodes. NDN is only one example of an ICN or even a general network architecture that is deployable on the testbed.

The *Node* function block allows to access a single physical device. The block can be used to copy files or to schedule commands to be executed before the emulation starts, or at predefined points in time during the emulation.

Management of application life-cycle can also be performed by using specialized function blocks. Besides starting the streaming application, the *DAS Client* and *DAS Server* blocks configure the applications and collect log files after the completion of the emulation. Thereby, consumer logs containing information about Quality of Experience (QoE) are collected on a single place on the gateway server.

5.2 Influence of Consumer Mobility on Video Streaming Quality

We now use the wireless network emulation capabilities of our testbed to evaluate the influence of different mobility patterns on video streaming performance as well as the benefits of early loss detection provided by WLDR. Our findings are based on the streaming quality reported by the client applications, as experienced in various settings involving different mobility patterns and deployments of WLDR.

5.2.1 Emulation Scenario. The underlying network topology used in the evaluation consists of the Abilene Core topology [1] which is extended by wireless access networks hosting the video consumer applications. The wireless access networks are realized as emulated 802.11g WiFi networks containing a mobile consumer (client) and a static access point which is a member of the Abilene network. The position of the access networks on the Abilene network as well as the position of the video producer (server) is selected randomly. Wireless links are modeled as Nakagami-lognormal channels [13]. A link's performance is obtained by simulation using ns-3[10] and depends on consumer mobility, following three well-established mobility models: *Constant Position*, *Random Direction* and *Random Walk* [3]. In case of Constant Position, the consumer is positioned directly below the access point and does not move during the emulation. In case of Random Direction and Random Walk, the consumer walks around the access point in a pattern similar to those depicted in Figure 6 at a constant walking speed of 1.4 meters per second. Although these mobility models do not model human behavior accurately and have been criticized [3], they can be used to demonstrate the testbed's capability to emulate various mobile scenarios.

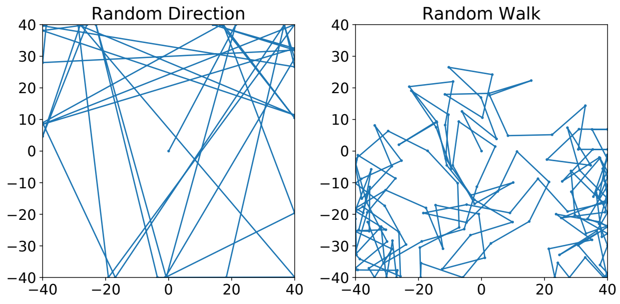

Figure 6: Consumer mobility patterns based on the random direction and random walk mobility models.

The video consumer and producer applications use the NDN stack for the video streaming process. NDN routing information in the core network is based on the shortest path; Interests are forwarded using the *Best Route* forwarding strategy [6].

During an emulation run, two clients stream the same video published by a single producer. The video itself is a 21 minute long concatenation of two animation videos, encoded in three representations with bitrates ranging from 640 kbps for the base layer, 1 Mbps for medium quality and 1.4 Mbps for the best quality, taken from the scalable video coding dataset of [5]. The consumer and producer applications are based on the work done by Rainer et al. [8] and use the therein mentioned buffer-based adaptation logic with an Interest lifetime of 500 milliseconds. For transmission, video segments are divided into several Data packets, each carrying 2048 bytes.

[10]https://www.nsnam.org/, last accessed 2018-05-25

WLDR [4] is implemented by an additional service observing all incoming packets logged by the NDN Forwarding Daemon (NFD). Once detected, packet loss is reported to the sender via a notification using UDP. In emulations using WLDR, the WLDR service is only running on the pairs of nodes in the WiFi access networks.

5.2.2 *Streaming Performance on WiFi Links.* In order to rate the streaming performance, we focus on QoE characteristics which are perceivable for users watching the multimedia streams. Besides the average played out representation, the number and length of stalling events as well as the number of quality variations are of particular interest. A stalling event occurs if the video player does not receive the base layer of a video segment in time and the playout of the video gets suspended. We define a quality variation as change of the consumed video quality between two consecutive segments.

When focusing on the influence of mobility, we see that the consumer's mobility has a strong impact on the streaming performance, as visualized in Figure 7. The left part of the figure displays the average segment's representation after 15 emulation runs. The solid orange lines in the boxplots represent the median values, the dashed green lines illustrate the average over all emulation runs. The upper and lower borders of a box depict 25% and 75% quartiles, whiskers show the variability outside the quartiles, individual points illustrate outliers. The available representations range from *Representation 0*, which requires the SVC base layer only, up to the highest available bitrate with *Representation 2*, requiring the base layer and both enhancement layers. In the case of Constant Position scenarios, which do not involve packet loss on the WiFi channel, the highest average played out representation is achieved. The average consumed representation is about 1.5, which means that most segments were played out in one of the two higher representations. With moving consumers, the average representation drops significantly to 0.6 in case of Random Direction mobility, and 0.8 in case of Random Walk mobility.

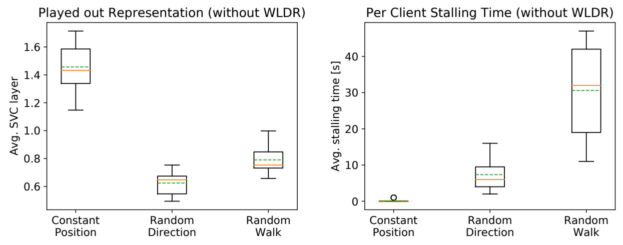

Figure 7: Average played out video representation and stalling time under different mobility models.

The right part of Figure 7 displays the average stalling time of consumers, where again the best results were achieved in the Constant Position scenario. In this case, virtually no stalls were observed at all. In case of Random Walk mobility, where it could happen that clients stay at the edge of the AP's coverage area for a longer period of time, the stalling time during playing the video sums up to a total of more than 30 seconds per client in some cases. Consumers moving according to the Random Direction model perform significantly better. In this case, a maximum of 16 seconds stalling time was observed. Nevertheless, none of the clients was able to retrieve the video without playout interruption.

Figure 8 shows the progression of the consumed representations and the number of encountered stalling events per segment for different mobility scenarios. The greyscale intensity indicates the average streamed representation of a segment, while the number of clients stalling at a certain segment is illustrated by the height of the red bars in the lower part of the charts.

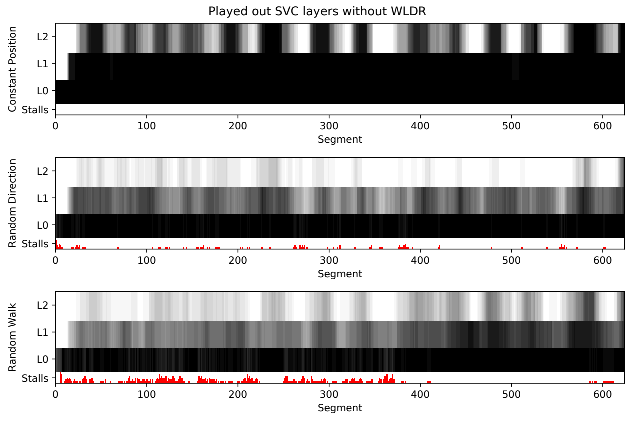

Figure 8: Progression of retrieved SVC layers and number of stalls per segment.

In the Constant Position scenario, all clients are able to retrieve Layer 0 and Layer 1 after a ramp-up phase at the start of the stream, which is caused by the clients' adaptation logic[11]. The adaptation logic tries to fetch Layer 2, if the playout buffers for the lower layers are sufficiently filled. This results in oscillations of Layer 2, which indicates a too low bandwidth for fetching the best provided representation. Regarding the scenarios involving movement, we can see small bursts of stalls in case of Random Direction mobility. This differs from Random Walk mobility, where longer stalls can be noticed, which is because clients stay longer on the edge of the AP's coverage area, where packet loss can exceed 40%. At Random Direction, only short time intervals are spent at the edge of the AP's coverage area.

[11]The adaptation logic is a client-side component deciding which layers to retrieve when streaming multimedia content.

5.2.3 Streaming Performance Benefits of using WLDR. To quantify the benefits of WLDR, we enhance the wireless access networks by adding the WLDR service and repeat the evaluations from the previous section. Comparing the results without WLDR in Figure 7 to the results including WLDR in Figure 9, no significant improvement in the average consumed representation is measured, although the number of stalls decreased significantly. The difference in the average consumed representation and stalling time between the different mobility models persists. Examining the worst case setting regarding stalls, Random Walk, we quickly see that the use of WLDR reduced the maximum stalling time from almost 50 seconds to only 31 seconds. Besides the improvements of the maximum stalling time in both scenarios considering mobility, the average stalling times were significantly reduced as well. In the case of Random Direction, both clients are in some emulation runs even capable of streaming the video without stalls when using WLDR.

Keeping the characteristics of the mobility models in mind, we find the use of WLDR particularly beneficial in situations of prolonged higher packet loss such as in the Random Walk scenario. Short-term spikes in packet loss as prevalent in the Random Direction scenario can be compensated using the client's playout buffer, diminishing the impact of WLDR.

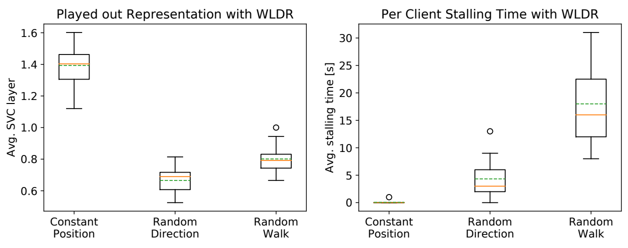

Figure 9: Average played out video quality and stalling time under different mobility models and with WLDR enabled.

Comparing the results of this evaluation to the findings of a previous evaluation [10], where the influence of WLDR on the streaming quality for non-scalable codecs was investigated, we noticed slightly different results. Although WLDR improved the quality of experience in both evaluations, [10] observed improvements in the average streaming quality in terms of bitrate, while we observed an improvement in respect to the stalling behavior. One possible reason for this difference is the layered quality improvement concept utilized by scalable codecs. While low quality representations with non-scalable codecs require a low bitrate, the base layer in SVC already requires a relatively high bitrate compared to the enhancement layers. In lossy environments, this could result in situations where not even the base layer can be successfully retrieved, which makes it impossible to play out a higher quality. In these situations, WLDR helps to retrieve the base layer and thereby reduces stalling events.

6 CONCLUSION

The main part of this paper focuses on the presentation of a testbed for wireless network emulation. We extended an existing low-cost testbed for fixed networks with the functionality to change link properties dynamically. This allows to emulate wireless networks, as well as link failures and recoveries or other events concerning links. In addition, we developed the testbed's *Execution Engine* (EE), which allows to perform network emulations as conveniently as with event-based simulators.

We demonstrated the capabilities of the testbed by investigating the influence of different mobility models on MPEG-DASH SVC video streaming performance and the benefits of explicit loss detection and notification (WLDR) in lossy wireless networks. Our results show the importance of considering client mobility in investigations involving wireless networks. Furthermore, the results underline the benefits of early loss detection in NDN, realized by WLDR.

For future research, we plan to use the testbed for evaluations involving real-world network traces featuring various mobile scenarios, such as vehicular mobility as published in [7], complementary to evaluations with synthetic mobility models. Furthermore, we plan to use the testbed's execution engine without ICN deployment for investigations on fog computing. From a more technical point of view, the support of containerized applications, abstracting the underlying software stack to a more general environment is planned. Besides providing a clean-slate environment, containerization would ease the development and deployment of apps applicable on a broad range of different testbed environments.

The ease of use of the testbed's EE for setting up emulations makes us believe that other researchers could benefit from using such a tool as well. Generic concepts, not focusing on specific hardware platforms, allow the use of the EE for arbitrary testbed platforms. To allow the community to adopt the EE for their purposes, we published the open-source licensed source code and provide installation instructions[12]. Along with the source code, we additionally publish the function blocks and the emulation scenarios, which allows for the reproduction of the presented results.

ACKNOWLEDGMENTS

This work was supported in part by the Austrian Science Fund (FWF) under the CHIST-ERA project CONCERT (project no. I1402).

[12]http://icn.itec.aau.at/ndn-network/

REFERENCES

[1] 2005. *Abilene Network*. Technical Report. 2 pages. https://web.archive.org/web/20120324103518/http://www.internet2.edu/pubs/200502-IS-AN.pdf

[2] 2017. *Guidelines for Implementation: DASH - AVC/264 Test cases and Vectors*. Technical Report Version 0.9. DASH Industry Forum. https://dashif.org/wp-content/uploads/2015/04/DASH-AVC-264-Test-Vectors-v09-CommunityReview.pdf

[3] T. Camp, J. Boleng, and V. Davies. 2002. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing* 2, 5 (Aug. 2002), 483–502.

[4] G. Carofiglio, L. Muscariello, M. Papalini, N. Rozhnova, and X. Zeng. 2016. Leveraging ICN In-network Control for Loss Detection and Recovery in Wireless Mobile networks. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking - ACM-ICN '16*. ACM, 50–59.

[5] C. Kreuzberger, D. Posch, and H. Hellwagner. 2015. A scalable video coding dataset and toolchain for dynamic adaptive streaming over HTTP. In *Proceedings of the 6th ACM Multimedia Systems Conference*. ACM, 213–218.

[6] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang. 2016. *ndnSIM 2: An updated NDN simulator for NS-3*. Technical Report NDN-0028, Revision 2. NDN.

[7] C. Mueller, S. Lederer, and C. Timmerer. 2012. An Evaluation of Dynamic Adaptive Streaming over HTTP in Vehicular Environments. In *Proceedings of the Fourth Annual ACM SIGMM Workshop on Mobile Video (MoVid12)*. ACM, 37–42.

[8] B. Rainer, D. Posch, and H. Hellwagner. 2016. Investigating the Performance of Pull-based Dynamic Adaptive Streaming in NDN. *Journal on Selected Areas in Communications* (Aug. 2016), 11.

[9] B. Rainer, D. Posch, A. Leibetseder, S. Theuermann, and H. Hellwagner. 2016. A Low-Cost NDN Testbed on Banana Pi Routers. *IEEE Communications Magazine* 54, 9 (Sept. 2016), 105 – 111.

[10] J. Samain, G. Carofiglio, L. Muscariello, M. Papalini, M. Sardara, M. Tortelli, and D. Rossi. 2017. Dynamic Adaptive Video Streaming: Towards a Systematic Comparison of ICN and TCP/IP. *IEEE Transactions on Multimedia* 19, 10 (Oct. 2017), 2166–2181.

[11] M. Sardara, L. Muscariello, J. Augé, M. Enguehard, A. Compagno, and G. Carofiglio. 2017. Virtualized ICN (vICN): towards a unified network virtualization framework for ICN experimentation. In *Proceedings of the 4th ACM Conference on Information-Centric Networking - ICN '17*. ACM, 109–115.

[12] I. Sodagar. 2011. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE Multimedia* 18, 4 (Apr. 2011), 62–67.

[13] T.T. Tjhung and C.C. Chai. 1999. Fade statistics in Nakagami-lognormal channels. *IEEE Transactions on Communications* 47, 12 (Dec. 1999), 1769–1772.

[14] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, kc Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang. 2014. Named Data Networking. *ACM SIGCOMM Comput. Commun. Rev.* 44, 3 (July 2014), 66–73.