A Low-Cost NDN Testbed on Banana Pi Routers

Benjamin Rainer, Daniel Posch, Andreas Leibetseder, Sebastian Theuermann, and Hermann Hellwagner

ABSTRACT

The computer communication research community has a significant interest in the paradigm of ICN. Continuously, new proposals for ICN-related challenges (caching, forwarding, etc.) are published. However, due to a lack of a readily available testbed, the majority of these proposals are evaluated by either theoretical analysis and/or conducting network simulations, potentially masking further challenges that are not observable in synthetic environments. Therefore, this article presents a framework for an ICN testbed using low-budget physical hardware with little deployment and maintenance effort for the individual researcher; specifically, named data networking is considered. The employed hardware and software are powerful enough for most research projects, but extremely resource-intensive tasks may push both components toward their limits. The testbed framework is based on well established open source software and provides the tools to readily investigate important ICN characteristics on physical hardware emulating arbitrary network topologies. The article discusses the testbed architecture and provides first results obtained from emulations that investigate the performance of various forwarding strategies. The results indicate that further challenges have to be overcome when heading towards a real-world deployment of ICN-based communication.

INTRODUCTION

In recent years, the information-centric networking (ICN) research community has grown continuously, positioning data-oriented communication as promising architecture for the future Internet. Today, there are a variety of ICN candidates interpreting and implementing data-oriented communication in their own ways, as surveyed in [1, 2]. Among them is named data networking (NDN) [3], probably the most elaborated approach. As for all data-oriented communication approaches, data objects are addressed by name. NDN-based communication follows a strictly consumer-driven communication pattern. Consumers issue so-called *Interest* messages to request *Data* objects. The Interest carries the name of the requested object, and it is forwarded toward the content origin via longest prefix-based matching of the naming components. Due to the principle of data-oriented communication and a new security model (content-based security [4]), an Interest's final destination does not necessarily have to be the content origin. Any intermediate network node that may hold a content replica may also satisfy a given Interest. For more details on the principles of NDN, we kindly refer the reader to [3].

The NDN community provides an extensive software suite with the objective to further advance research in this field. This includes the network simulator ndnSIM [5], which is based on the well-known ns-3 framework. Furthermore, the software suite includes the Networking Forwarding Daemon (NFD) [6], an implementation of the NDN principle on Linux-based systems. Although the Linux-based implementation is freely available, evaluations on physical networks are rarely conducted in research articles. The majority of proposals have been evaluated in a synthetic environment, either relying on theoretical analysis or using network simulations. While both methods are valid and necessary, they should not be the final step and substitute experiments on physical networks. There are a number of impediments for experiments on physical networks/testbeds:

- Setting up a testbed is a tedious and work-intensive task.
- Building a testbed of significant size can be very costly.
- Conducting a large batch of evaluations using different parameters in a testbed is more time consuming and error-prone than using a flexible simulation environment.

In order to overcome these impediments and to support researchers in their daily work, the provision of a testbed is necessary. Currently, the NDN community provides a testbed [7] available to all community members. The testbed connects NDN nodes around the world via the Internet, providing the opportunity to evaluate NDN-based applications. Since this testbed is a shared resource and can be used by all members, the availability of the testbed for the individual researcher may be limited. Furthermore, researchers may not be able to experiment with the core functionality of the NFD (even if it is necessary for their research objectives), since the NDN testbed needs to stay functional at all times. Also, the NDN testbed is realized as an overlay network on

The computer communication research community has a significant interest in the paradigm of ICN. However, due to a lack of a readily available testbed, the majority of these proposals are evaluated by either theoretical analysis and/or conducting network simulations, potentially masking further challenges that are not observable in synthetic environments. The authors present a framework for an ICN testbed using low-budget physical hardware with little deployment and maintenance effort for the individual researcher.

The authors are with Alpen-Adria-Universität Klagenfurt.

0163-6804/16/$25.00

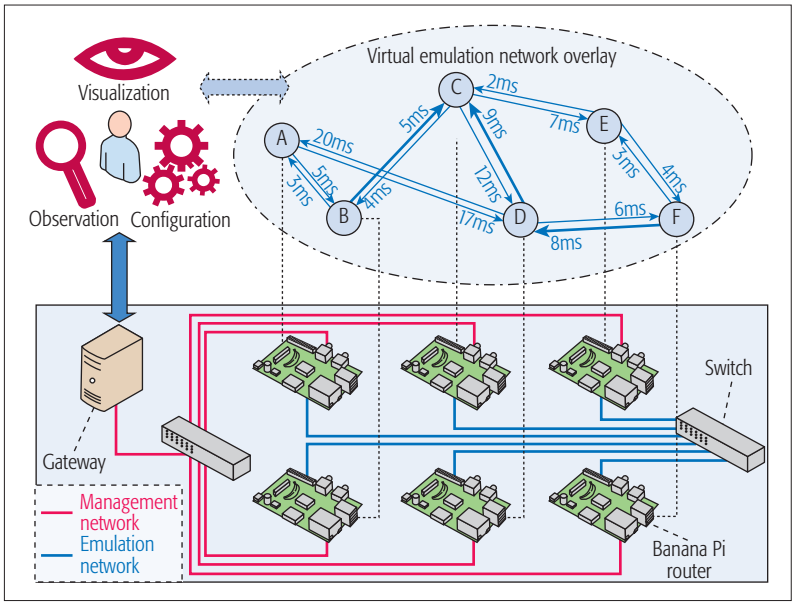

Figure 1. An overview of the testbed architecture. The testbed consists of two dedicated physical networks. The management network (red lines) is used to set up and control the testbed nodes, while the emulation network (blue lines) realizes the virtual overlay network. The gateway is used as a control server, and enables users to configure, observe, and visualize emulation tasks.

top of today's Internet using ordinary computers as software routers. Since the researchers do not have full control over the used infrastructure (devices, links, etc.), side effects may influence their evaluations, leading to distorted results and aggravating reproducibility.

To tackle the above mentioned challenges, this article presents a framework enabling researchers to readily deploy their own NDN testbed with low costs. We suggest a testbed framework that is based on a set of single-board devices, so-called Banana Pi routers.

While focusing on a low budget, the selected hardware is powerful enough to conduct significant evaluations in all research areas of interest regarding ICN. An exception is the use of computationally expensive cryptography; however, this would also challenge recent off-the-shelf CPUs such as used in ordinary desktop computers without cryptographic hardware support.

The costs for an NDN Banana Pi testbed of significant size (e.g., 20 nodes) is approximately US$3400. Costs scale linearly with approximately US$160 per added testbed node. We provide pre-configured disk images, scripts, and installation guidelines for download (http://icn.itec.aau.at), enabling researchers to easily realize their own NDN testbed with little effort. Once the images are copied to the required disks, and the testbed hardware is assembled and connected to a network, the testbed is ready for use. The proposed framework allows arbitrary network overlays to quickly be deployed on the physical topology. Researchers may specify every detail of the topology including link capacities, delays, queuing disciplines, queue sizes, and more. The framework further provides a web interface for observing the health status of the testbed and allows watching ongoing emulations in near real time.

Testbed: Architecture, Deployment, Emulation Environment, and Monitoring

This section discusses the testbed architecture. First, insights on the selected hardware are given. Second, the tools and applications necessary to realize an arbitrary network topology on the physical devices are introduced, and their configuration is discussed. Finally, the web-based monitoring possibilities for the testbed are presented.

Testbed Architecture and Hardware Requirements

Figure 1 depicts an overview of the testbed architecture. The testbed is based on an IP network and realizes a virtual NDN overlay. In general, the testbed consists of a number of single-board computers, at least two network switches, and a gateway. The testbed architecture requires two dedicated networks, each forming a star topology. The first network is denoted as the *management network* (MN), while the second one is denoted as the *emulation network* (EN). This clear separation ensures that traffic in the management network (setting up nodes, monitoring devices, deploying software, etc.) does not interfere with, or influence, active emulations. An ordinary PC provides enough resources to manage the tasks of the gateway. The gateway is connected to the MN only, providing external communication and control functionality.

It acts as the control server for conducting emulations and is therefore the entry point for the users to the testbed. Furthermore, it hosts several (optional, but very useful) services, which allow the monitoring of the testbed (discussed in detail later). The single-board computers should be able to access the Internet via the gateway, so software updates can be retrieved easily.

Since the testbed architecture foresees two dedicated networks, the testbed nodes (single-board computers) are required to provide at least two network interfaces. We have chosen Banana Pi routers (BPI-R1) because they are equipped with a four-port LAN switch including a dedicated switching circuit. Moreover, these devices provide a WLAN interface (supporting IEEE 802.11 b/g/n), which additionally gives the opportunity to support mobility scenarios. Another advantage of the BPI-R1 is that it offers a SATA 2.0 port. We consider this an important feature, especially for conducting emulations in the area of ICN. Users may want to conduct experiments requiring large in-network caches. When using only ordinary Micro SD cards (default disk storage for embedded devices and single-board computers), such experiments could not be performed since the cache size would be constrained by the main memory (1 GB DDR3-SDRAM). Paging memory to the Micro SD cards with rather low read/write data rates would drastically decrease the performance below the required line speed. Therefore, we strongly suggest equipping the Pis with a solid state disk (SSD). We summarize the most important hardware specifications of a BPI-R1 in Table 1. Furthermore, Table 2 lists the required hardware components (including approximate costs

Component	Description (http://bananapi.com)
CPU	ARM-Cortex-A7 (2x1.0 GHz)
GPU	Mali-400MP2
Memory	1 GB DDR3-SDRAM
Storage	1x Micro SD, 1x SATA 2.0
Network	1x Ethernet RJ45, 4-Port-Switch, WLAN 802.11b/g/n
Power source	5 V/ 2 A via Micro USB

Table 1. Specification of a Banana Pi router (BPI-R1).

per unit) for a testbed with 20 nodes. Note that we did not include hardware for the control server because any available PC should be able to handle the required workload and does not burden one's budget.

Deployment of the Virtual Network Topology

Once the Pis are assembled (they are usually shipped without a case) and connected to the switches, the next step is to deploy the desired virtual network overlay on top of the testbed. The BPI-R1 is shipped with its own operating system called Bananian, a customized Debian Linux composed by the hardware manufacturers. Thus, basically all networking tools available for Linux, for example, *iptables* and *traffic control* (*tc*), can be used to realize the desired overlay network. However, in order to use some more advanced features of the *tc*, the Linux Kernel has to be re-configured and re-compiled as the default disk image is optimized with respect to space constraints rather than for networking functionality. Therefore, we provide a modified Bananian image for download (http://icn.itec.aau.at).

We continue the discussion by providing the technical details to deploy a virtual overlay on the testbed. All discussed steps are implemented in script(s) that can be downloaded at the previously indicated web page. Researchers using the framework may specify arbitrary overlay networks (in an ordinary text file, or using the provided random network topology generator) and apply them on the physical devices instantaneously. Note that only the EN will be modified. The MN remains always unchanged as it is only used for configuration and monitoring tasks.

Let us assume that the overlay network illustrated in Fig. 1 shall be deployed. The directed edges define the virtual links connecting the individual nodes (thicker lines indicate higher link capacities), and the link delays are indicated next to directed edges' heads in milliseconds. The first step is to model the virtual connection(s). This can be achieved with the application *iptables*, which enables the configuration of the tables provided by the Linux kernel firewall. Initially, we configure the tables to block all IP packets received or transmitted over the EN. Then, for each virtual link in the overlay network, an exception is inserted allowing the forwarding/receiving

Component	Advice/comment	≈ Cost/unit
20x BPI-R1		US$80
20x Case for BPI-R1	Optional, but handy	US$15
20x SSD	Size of at least 120 GB	US$50
20x Micro SD	Size of at least 8 GB	US$4
20x USB power cable	Dimensioned for 2 A	US$3
4x USB power hub	At least 6 ports	US$20
2x Gigabit switch	At least 24 ports	US$100
40x Ethernet cable	CAT6, two colors, 5 ft	US$2
	Total:	**US$3400**

Table 2. Components for a testbed with 20 nodes.

of IP packets. For instance, in the case of node A these are the upstream and downstream connections to *B* and *D* (Fig. 1). Figure 2a depicts the required rules to realize the connections for node A. Once the *iptables* are configured on all nodes, the basic topology of the overlay network is reflected by the EN on the IP layer.

The next step is to enforce the link capacities and delays as indicated by the overlay description. For this task the framework relies on *tc*, an application to configure and control the Linux kernel's network scheduler as sketched in Fig. 2b. The figure illustrates the employed traffic control classification for packet scheduling by the example of node E. We use hierarchical token bucket (HTB) filters, with multiple child classes (one class per up- or downstream connection). The bandwidth capacities are not limited at the HTB level, but at the individual HTB's child classes. Each child is equipped with an additional queuing discipline, a classical token bucket filter (TBF) controlling the link capacities and queue sizes. Furthermore, each child class is equipped with a *netem* discipline realizing the individual link delays. One can go even further and use *netem* to introduce artificial packet loss or packet corruption facilities if desired; however, we do not discuss this further and refer to [8, 9] for details.

Installing the NDN Communication Layer

The final step before emulations can be conducted is to set up the NFD [6] accordingly on the Pis. This application implements the NDN-based communication layer. To make this task as comfortable as possible, the framework also takes over this part. The user may specify the configuration of the NFDs (caching strategies, cache sizes, forwarding strategies and entries, etc.) and deploy it on the individual Pis using a deployment script. The NFD supports two baseline caching strategies, least recently used (LRU) and first-in first-out (FIFO), and several Interest forwarding strategies (BroadCast, BestRoute, and NCC) [6]. Nevertheless, one may be interested in implementing more sophisticated approaches. To

The testbed architecture requires two dedicated networks, each forming a star topology. The first network is denoted as the management network, while the second one is denoted as the emulation network. This clear separation ensures that traffic in the management network does not interfere with, or influence, active emulations.

There are basically two levels of monitoring that can be accessed via the web interface. The first level provides a very rough overview that allows monitoring the health state of the testbed. The second monitoring level is more detailed and is active once emulations are conducted.

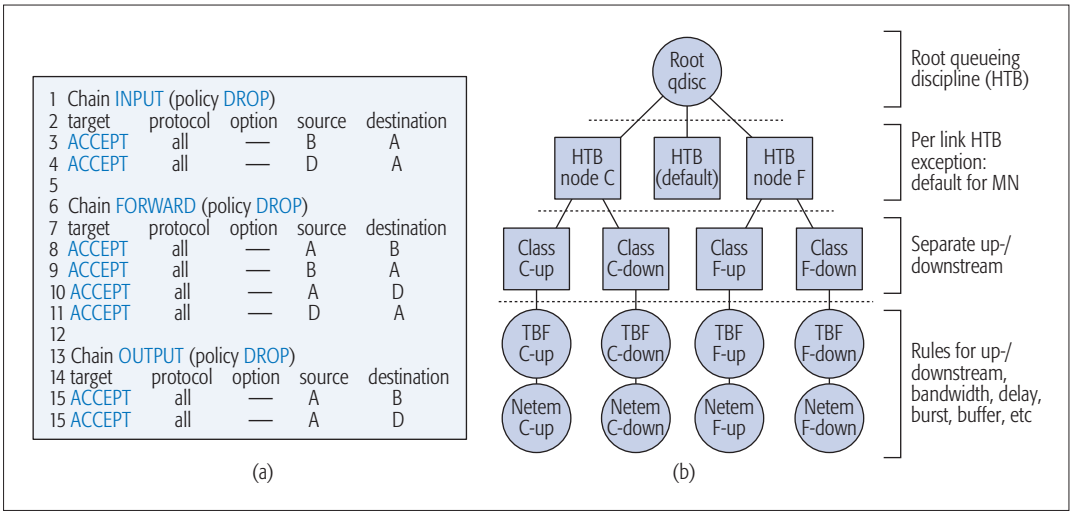

Figure 2. The left side sketches the configuration of the Linux iptables by the example of node A (Fig. 1). The right side illustrates the Linux traffic control classification for packet scheduling by the example of node E.

that end, references [10, 11] provide good surveys of recent caching strategies, while the related work in [12] gives a good overview of existing forwarding strategies.

To complete the configuration, the installation of the routing and forwarding entries in the routing and forwarding information base of the NFD [6] is required. The framework provides two choices when users decide to automatically deploy routes, similar as implemented in ndnSIM [5]. The user may choose to use either all possible routes, or only the shortest paths between each pair of nodes. The framework implements the installation of the entries as follows. Each node is given a unique identifier. Then for each forwarding strategy and for each other node, one forwarding entry of the following structure is installed: */fw-strategy/source-node-id*. If using all routes has been selected, the entry may point to multiple outgoing faces. Hence, any user-defined application within the testbed can easily choose the source from which it may consume data by specifying the producer's node identifier. One further chooses the forwarding strategy by indicating the strategy name in the emitted request message.

Conducting and Observing an Experiment

To conduct a batch of emulations/experiments, the user needs to specify the emulation parameters in a script. Basically, the framework foresees the following behavior. The user specifies two kinds of abstract applications: *consumers* and *producers*. Which application is executed on which node has to be specified in the previously mentioned topology file. First, a script starts the logging functionality on all Pis, and then all producer/consumer applications are executed. The gateway monitors the Pis and stops the emulation once all consumer applications have finished. Then the logging functionality is stopped, and the corresponding logfiles are gathered and stored on the gateway for later processing. The logfiles provide information about the Pis including CPU load, power consumption, and memory usage, and may also contain application-specific data provided by the specified consumer/producer applications.

Testbed Monitoring

Testbed monitoring is coordinated by the gateway (Fig. 1). There are basically two levels of monitoring that can be accessed via the web interface. The first level provides a very rough overview that allows monitoring the health state of the testbed. To that end, each Banana Pi uses *cron* (a job scheduler for Linux) to trigger the logging of important data periodically (e.g., on an hourly basis). The logged data is provided in JSON format and is collected by the gateway using *wget* (a program to fetch data via HTTP-/FTP connections). This data includes important information about the Pis including CPU load, power consumption, memory usage, network traffic, disk usage, and so on. A history of about one week is stored at the gateway, providing a quick overview of whether or not the testbed has been operating as expected. The second monitoring level is more detailed and is active once emulations are conducted. During an experiment, every few seconds the Pis actively push logfiles to a virtual RAM disk on the gateway (through the MN) using the Network File System (NFS). These files are then accessible via a web server, and users may monitor the ongoing experiment in near real time using the web interface, as illustrated in Fig. 3. The second monitoring level should be disabled for experiments demanding highly accurate measurements of the CPU load or power consumption, as the provision of this real-time log data requires little but noticeable resource consumption.

Testbed Evaluation and Comparison to Network Simulations

This section presents the design and results of an exemplary experiment conducted on the testbed. The selected experiment investigates the influence of different forwarding strategies on the data delivery performance of NDN. Furthermore, the obtained results are compared to the same experi-

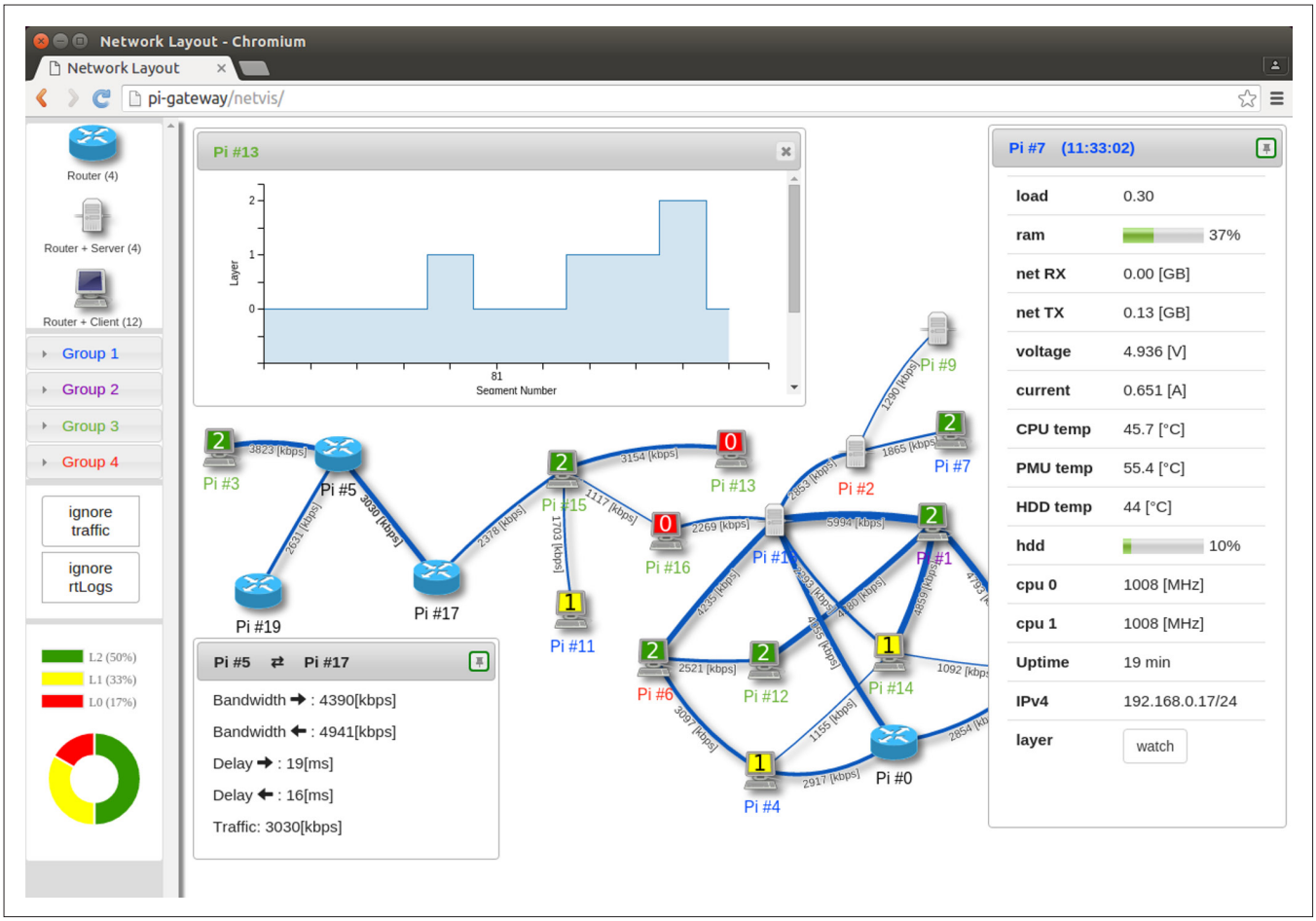

Figure 3. Snapshot of the web interface that can be used to monitor ongoing emulations in near real time.

ment conducted using ndnSIM 2.0 [5]. Finally, we push the Pis to their limits to assess the maximum work load these devices can handle.

Evaluation Scenario

In the experiment we want to investigate the performance of the following forwarding strategies: Broadcast [6], BestRoute [6], NCC [6], RFA [13], and SAF [12]. Due to the limited number of available testbed nodes, we decided to model a typical peer-to-peer overlay network. We generate network topologies consisting of $n = 20$ nodes using the Erdős Rényi model [14]. The probability of creating a link between two nodes was set to $p = 0.15$, and the link delays $d \in [5$ ms, 20 ms] are drawn from a uniform distribution. We ensure that the generated graph is connected by omitting topologies not satisfying this condition. For the bandwidth capacity of a link we consider three different settings: LowBW, that is, from the interval [2000 kb/s, 3000 kb/s]; MediumBW, that is, from [3000 kb/s, 4000 kb/s]; and HighBW, that is, from [4000 kb/s, 5000 kb/s]. Each node maintains a 250 MB large cache using the FIFO replacement strategy. For each simulation/emulation, 4 nodes are randomly assigned as servers providing unique content to 12 clients. Each client requests content with a constant bit rate of approximately 2000 kb/s from a single server, with a Data packet size of 4 kB. The pairing between client and server is randomized for each run. In total, 40 runs are performed for each configuration simulating/emulating 30 min of network traffic.

Please note that this evaluation scenario is exemplary to compare results obtained from the testbed with results obtained from simulations. The evaluations are not sufficiently extensive to assess the real performance of the investigated strategies. Nevertheless, this experiment may indicate interesting trends.

Results Obtained from the Testbed

Figures 4a–d illustrate the results of the experiment when conducted on the testbed. Figure 4a shows the *Interest satisfaction ratio*, which provides the ratio of received Data packets and generated Interests per client. The figure indicates that SAF outperforms its competitors in scenarios with limited bandwidth; however, as bandwidth resources are increased, the other algorithms start to catch up, particularly BestRoute. Figure 4b depicts the average *cache hit ratio* per network node. Here one can observe that Broadcast and NCC obtain the highest cache hit ratio. However, one has to consider that those strategies tend to replicate Interests, especially Broadcast, and therefore, the cache hit ratio always has to be seen in relation to the caused traffic and the achieved Interest satisfaction ratio. In Figs. 4c and 4d the average *power consumption* and the average *CPU load* per node are illustrated. These results can only be obtained from the testbed. This makes them particularly interesting for investigating the real-world

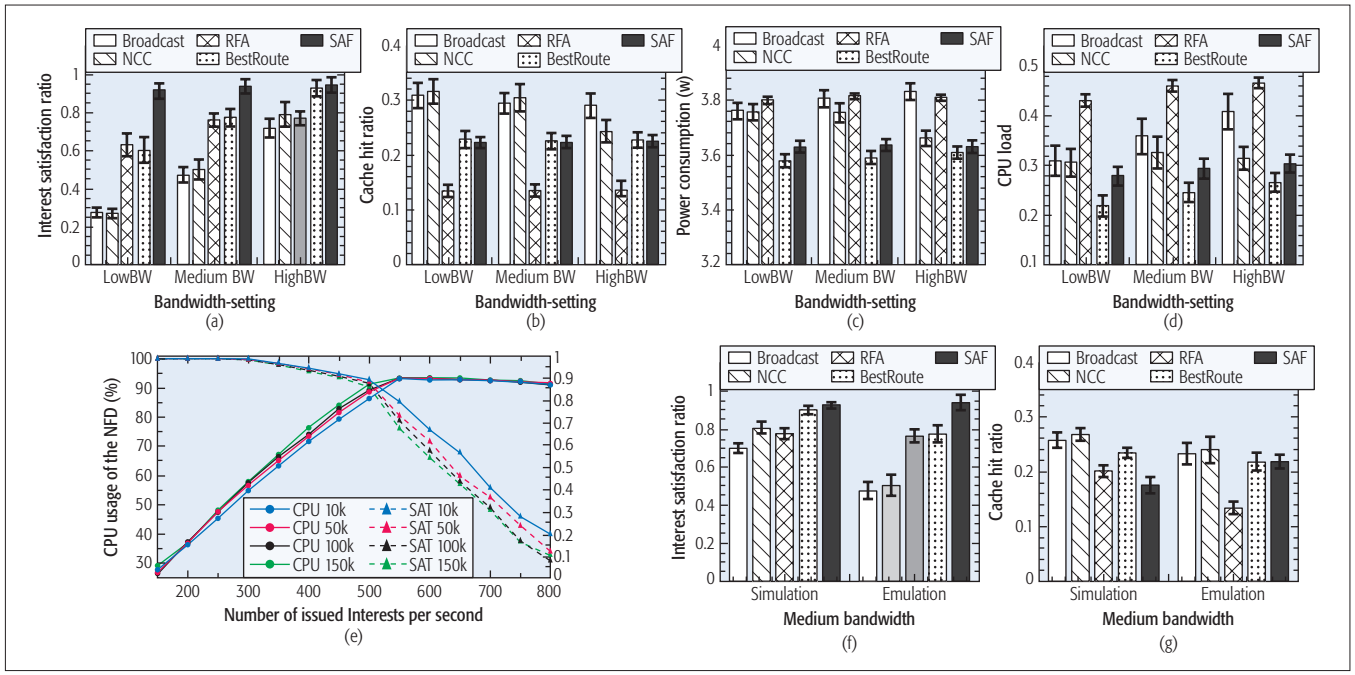

Figure 4. The figure depicts the findings of the conducted experiments. Subfigures 4a–d illustrate the results obtained from the testbed. Subfigures 4f and 4g compare the results obtained by emulations and simulations for the setting "MediumBW" also. Subfigure 4e indicates the packet processing capabilities of a BPI-R1 with respect to varying cache sizes (10,000–150,000 entries).

requirements of algorithms. It is evident that both SAF and BestRoute consume less power on the single-board computers than the other algorithms. For an environment with scarce energy resources, these algorithms may thus be better suited. Regarding CPU load, we can observe that RFA requires the most CPU resources, which may lead to problems on constrained, embedded devices. Also Broadcast tends to use a lot of resources, as the relentless duplication of Interests and especially their later processing, requires a lot of resources.

To assess the capabilities of the BPI-R1 we conducted another experiment measuring the CPU usage of the NFD with respect to the processed packets. Figure 4e illustrates the result of this experiment, where we varied the number of issued Interests from 100 to 800 packets/s (one satisfied Interest results in two processed packets, i.e., an additional Data packet). Furthermore, we conducted this experiment with different cache sizes. The employed cache sizes are 10,000, 50,000, 100,000, and 150,000 entries. We always start the individual runs with full caches to force extensive entry replacement to assess its influence on the CPU load. Figure 4e shows that the CPU load of the NFD increases proportionally with the number of processed Interests. Also, larger cache sizes increase the CPU load, as the replacement of entries becomes more expensive. With 500 issued Interests, the BPI-R1 reaches its capacity limits, leading to drastic packet loss if more packets have to be processed. Although the BPI-R1 maintains a dual-core CPU, the NFD [6] is not capable of using more than one core, limiting the Pi's capabilities. One can translate this result in a possible data rate a BPI-R1 may handle using the NFD on a single CPU core. Assuming a Data packet size of 8 kB (maximum currently supported by the NFD), a Pi may handle at most a data rate of $500 \cdot 8 \cdot 8192 = 32{,}768$ kb/s. If researchers require more than 500 forwarded Interest/Data pairs per second and/or a higher data rate for their experiments, there are two options:

- Use more powerful devices. We expect that there will be a successor for BPI-R1 available shortly, with similar hardware as the recently released BPI-M3 (octa-core CPU with 8 × 1.8 GHz).
- Update the NFD to make use of multiple CPU cores.

Please note that for all our experiments the digital signing of packets as foreseen by [3] was disabled. Using public-key cryptography [15] for real-time packet signing without dedicated cryptographic hardware is not feasible even with powerful desktop CPUs.

Comparing the Results of ndnSIM and the Testbed

We anticipate that the observed performance of the algorithms in the simulator is better than on the testbed. We expect this because ndnSIM [5] does not consider the overhead caused by the underlying protocols (UDP: 8 bytes, IPv4: 20 bytes, Ethernet: 18 bytes) that are required for communication on the testbed. Instead, the simulator uses a virtual channel to transmit NDN packets with nearly no overhead (2 bytes for a protocol flag).

As already mentioned in the previous section, using ndnSIM we are not able to obtain figures for the power consumption or the CPU load required by the algorithms. Therefore, Figs. 4f and 4g only depict the results obtained for the Interest satisfaction ratio and the cache hit ratio. Please note that we compare only the results for the setting *MediumBW* to ease readability; the results for the other two settings are similar. The left set of bars in Figs. 4f and 4g depict the results obtained by conducting simulations, while the right set of bars depict the

results obtained conducting emulations on the testbed. It is evident from Fig. 4f that especially Broadcast and NCC suffer from severe performance loss when executed on the testbed. This is definitely due to the larger overhead in the testbed as these strategies tend to replicate Interests. Considering that the average Interest size during simulation was about 40 bytes (according to the ndnSIM documentation, the minimum size of an Interest message is 14 bytes), an additional overhead of 8 + 20 + 18 = 46 bytes roughly doubles the size of each forwarded Interest. Also, BestRoute suffers from minor performance loss; however, as this strategy does not replicate Interests, the impact is substantially smaller. Interestingly, the forwarding strategies RFA and SAF do not suffer from significant performance loss. This is due to their basic principles of distributing traffic among the interfaces with the lowest load without creating replicas of Interests avoiding Interest drops at congested interfaces. Regarding the cache hits illustrated in Fig. 4g, there is no significant difference observable for the algorithms Broadcast, NNC, and BestRoute as their confidence intervals are overlapping. Surprisingly, SAF is able to maintain a slightly higher cache hit ratio when executed on the testbed, while for RFA exactly the opposite is true.

Conclusion and Outlook

This article presents a framework that enables researchers to readily deploy an NDN testbed on low-budget single-board computers, so-called Banana Pis. All required tools/applications are introduced, and an open source contribution provides disk images, scripts, and guidelines to enable researchers to set up their own testbed. The article shows the potential benefit of such a testbed by conducting a baseline experiment. A comparison of network emulations and simulations shows that the abstraction introduced by simulations hides important details (overheads, resource usage, etc.). For the future we expect that this testbed will help ICN/NDN researchers assess the performance of their proposals in more detail. Conducting experiments on physical networks and devices will provide researchers with deeper insights on their algorithms, reveal further challenges, and potentially support the progress in the research area of data-oriented communication.

Acknowledgment

This work was supported in part by the Austrian Science Fund (FWF) under the CHIST-ERA project CONCERT (A Context-Adaptive Content Ecosystem Under Uncertainty), project no. I1402 at Alpen-Adria-Universität Klagenfurt.

References

[1] B. Ahlgren *et al.*, "A Survey of Information-Centric Networking," *IEEE Commun. Mag.*, vol. 50, no. 7, 2012, pp. 26–36.
[2] G. Xylomenos *et al.*, "A Survey of Information-Centric Networking Research," *IEEE Commun. Surveys & Tutorials*, vol. 16, no. 2, 2014, pp. 1024–49.
[3] L. Zhang *et al.*, "Named Data Networking," *ACM SIGCOMM Comp. Commun. Rev.*, vol. 44, no. 3, 2014, pp. 66–73.
[4] V. Jacobson *et al.*, "Networking Named Content," *Proc. 5th ACM Int'l. Conf. Emerging Networking Experiments and Technologies*, 2009, pp. 1–12.
[5] S. Mastorakis *et al.*, "ndnSIM 2.0: A New Version of the NDN Simulator for NS-3," tech. rep. NDN-0028, UCLA, 2015.
[6] A. Afanasyev *et al.*, "NFD Developer's Guide," tech. rep. NDN-0021, 2014; http://named-data.net/publications/techreports/nfd-developer-guide/
[7] Data Networking Consortium, "The NDN Testbed," http://named-data.net/ndn-testbed/; last accessed: Jan. 2016.
[8] B. Hubert *et al.*, "Linux Advanced Routing and Traffic Control HOWTO v.1.0.1," 2012; http://www.lartc.org/; last accessed: Jan. 2016
[9] S. Heminger, "Network Emulation with NetEm," *Proc. Linux Conf.* Australia, Apr. 2005, p. 9.
[10] G. Zhang, Y. Li, and T. Lin, "Caching in Information Centric Networking: A Survey," *Computer Networks*, vol. 57, no. 16, 2013, pp. 3128–41.
[11] M. Zhang, H. Luo, and H. Zhang, "A Survey of Caching Mechanisms in Information-Centric Networking," *IEEE Commun. Surveys & Tutorials*, vol. 17, no. 3, 2015, pp. 1473–99.
[12] D. Posch, B. Rainer, and H. Hellwagner, "SAF: Stochastic Adaptive Forwarding in Named Data Networking," Computing Research Repository (CoRR), 2016, p. 14; http://arxiv.org/abs/1505.05259.
[13] G. Carofiglio *et al.*, "Optimal Multipath Congestion Control and Request Forwarding in Information-Centric Networks," *Proc. 21st IEEE Int'l. Conf. Network Protocols*, Oct. 2013, pp. 1–10.
[14] P. Erdős and A. Rényi, "On Random Graphs I," *Publicationes Mathematicae*, vol. 6, 1959, pp. 290–97.
[15] A. J. Menezes, P. C. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

Biographies

Benjamin Rainer received his B.Sc., M.Sc. (Dipl.-Ing.), and Ph.D. (Dr. techn.) in computer science with distinction from the Alpen-Adria-Universiät Klagenfurt, Austria. He is a postdoctoral researcher at the Department of Information Technology (ITEC) in the Multimedia Communication (MMC) research group working on the CONCERT project, Alpen-Adria-Universitäat Klagenfurt. His research interests are audio/video encoding, parallel computing, quality of experience, and ICN/NDN.

Daniel Posch received his B.Sc. and M.Sc. (Dipl.-Ing.) in computer science with distinction from the Alpen-Adria-Universität Klagenfurt. He is pursuing a Ph.D. (Dr. techn.) in computer science at the ITEC in the MMC research group, Alpen-Adria-Universität Klagenfurt. His research interests are multimedia technologies in ICN/NDN. Currently, he works on Interest forwarding strategies for NDN with the objective to enhance multimedia content delivery.

Andreas Leibetseder received his B.Sc., and M.Sc. (Dipl.-Ing.) in computer science with distinction from the Alpen-Adria-Universitäat Klagenfurt. He is pursuing a Ph.D. (Dr. techn.) in computer science at the ITEC in the Distributed Multimedia Systems (DMS) research group, Alpen-Adria-Universität Klagenfurt.

Sebestian Theuermann is a B.Sc. student in computer science at the Alpen-Adria-Universität Klagenfurt. He is a student assistant at the ITEC employed in the CONCERT project funded by the Austrian Science Fund (FWF). Currently, he supports NDN research and is actively involved in the development of the NDN testbed.

Hermann Hellwagner [M] is a full professor of computer science in the ITEC, Alpen-Adria-Universität Klagenfurt, leading the MMC research group. His current research areas are distributed multimedia systems, multimedia communications, ICN/NDN, and quality of service. He has received many research grants from national (Austria, Germany) and European funding agencies as well as from industry, is the Editor of several books, and has published more than 200 scientific papers. He is a member of the ACM, German Informatics Society, and Austrian Computer Society, and is Vice President of the Austrian Science Fund.

For the future we expect that this testbed will help ICN/NDN researchers assess the performance of their proposals in more detail. Conducting experiments on physical networks and devices will provide researchers with deeper insights on their algorithms, reveal further challenges and potentially support the progress in the research area of data-oriented communication.