Emulation of Multipath Solutions in Heterogeneous Wireless Networks Over Ns-3 Platform

Vadym S. Hapanchak[1(✉)] and António D. Costa[2]

[1] Algoritmi Center, University of Minho, Braga, Portugal
b7768@algoritmi.uminho.pt
[2] Algoritmi Center, Department of Informatics, University of Minho, Braga, Portugal
costa@di.uminho.pt

Abstract. The increased availability of devices equipped with multiple wireless interfaces leads to consider multihoming as one of the key features of the next generation 5G networks. This paper discusses the emulation technique that integrates Linux Container and ns-3 network simulator to evaluate emerging multipath applications and protocols. The presented solution was utilized as an experimental platform to analyze the performance of Multipath TCP (MPTCP) protocol in heterogeneous network environments, particularly wireless ones, such as WLAN and cellular networks. We tested the MPTCP in fixed and mobility scenarios, exploiting the ns-3 to provide multipath wireless connectivity. The obtained results show the protocol behaviour that might have been expected from the system under investigation. Thus, one could use the presented scheme to get emulation results with reasonable accuracy, as long as ns-3 follows up with external real-time events. We further discuss the main limitations of the described method, observed in large-scale scenarios.

Keywords: Simulation · Ns-3 · Multipath · MPTCP · Network emulation · LXC · Heterogeneous · Wireless networks

1 Introduction

Nowadays, mobile devices support different types of wireless interfaces, such as WiFi and LTE, to connect to the Internet. The commonly used TCP/IP standard supports only one interface per connection since it cannot handle multiple data streams. It has been shown that multipath transmission can improve network resilience and increase data throughput by using multiple links simultaneously [3,11]. The use of multipath approach has been recently adopted for the fifth generation (5G) cellular networks for use in services that aim at supporting the requirements of high data rate, high reliability and low latency [1].

Multipath TCP (MPTCP) [4] is the well known multipath transmission protocol that has attracted significant attention in both academia and industry.

D. Jiang and H. Song (Eds.): SIMUtools 2021, LNICST 424, pp. 3–18, 2022.
https://doi.org/10.1007/978-3-030-97124-3_1

MPTCP is an extension to the regular TCP that allows single end-to-end connection data traffic to be split across different interfaces (also paths). In the context of wireless communication with mobility, MPTCP aims to provide reliable connectivity by seamlessly switching between different interfaces during data transmission.

Many researchers tend to evaluate the performance of multipath solutions by using a discrete-event network simulators or via a controlled environment. Simulation tools offer a possibility of testing new applications and protocol stacks for arbitrary complex scenarios. Generally, network simulators do not allow running "real" code and require dedicated implementations of newly designed algorithms for communication systems. However, it takes time for new protocols to be implemented and integrated into the simulator. For example, MPTCP is still not included as a module in any stable version of ns-3 simulator, and the available 3rd-party simulation modules are not fully compliant with the actual MPTCP specifications.

Therefore, we believe testing new multipath solutions using their real implementation is more realistic than developing a new module within the simulation tool. Controlled environment tests require real testbed deployment, which can be expensive and hard to manage. For instance, studying the performance of multipath solutions in the controlled heterogeneous wireless environment requires real testbed deployment of WiFi, 4G/5G systems, which brings upon high costs. Furthermore, real hardware equipment is challenging to modify, upgrade and scale.

On the other hand, the emulation technique can combine real application with the controlled and reproducible network scenario to obtain more accurate and credible results. The unmodified application code can be used in controlled networking environments. Thus, the emulation model can provide more realistic results than simulation but have a higher maintenance complexity.

Motivated by all these factors, in this work, we conduct container-based emulation to analyze how multipath protocols can be further tested by the advantages of using their real implementation. To demonstrate the abilities of the emulation method, we carried out a simple study of MPTCP using the ns-3 in emulation mode. In particular, we focus on wireless networks with differing characteristics, such as Wi-Fi and 4G.

However, the main contribution of this paper is the integration of Linux Container (LXC) with the ns-3 simulator that allows conducting complex real-time experiments, with realistic link-layer emulation (i.e., signal propagation, modulation techniques, antenna design, etc.) and mobility models. Detailed description of the implementation and configuration of the emulation framework is provided in this paper. In addition, the main limitations of the presented approach are discussed as well.

This paper is organized as follows. Section 2 presents the related works that have used the same emulation strategy. Section 3 presents the concepts and tools used to create an emulation environment and provide a detailed description of the emulator setup. Section 4 shows some scenarios used to study the behaviour of

MPTCP over heterogeneous wireless networks and analyzes the achieved results. Section 5 discusses the limitations of the proposed emulation method. Finally, Sect. 6 presents our conclusions and future work.

2 Related Works

There are various tools for simulation or emulation available for the scientific and research community. Network simulation tools, such as NS-3[1] and OMNET++[2], among others, offer a greater degree of flexibility and sufficient accuracy to analyze large network topologies. In contrast to simulation, where the entire experiment is modelled in software, emulation interacts with the real systems, such as end hosts and physical networks, to run the experiments. This provides the means to generate more accurate results from the experiment, close to the real environment.

CORE[3] and Mininet[4] are two well known lightweight emulators, which use Linux namespaces to provide isolation between emulated hosts. However, at the moment, these emulators have limited realism in modelling the physical layer effects of wired and wireless connections, such as signal interference and propagation.

Some works have successfully exploited the combination between Docker and ns-3 simulator [7–9,12]. The Docker framework is used to create LXCs, while ns-3 is used for network scenario generation. Nevertheless, the authors of these works do not discuss whether the presented emulators have any limitations.

Authors in [10] successfully simulated large LTE topologies by integrating the ns-3 simulator with the CORE emulator. Different scenarios and topologies were tested, and the real-time traffic was successfully exchanged between LTE devices in simulation and external equipment. Zhang et al. [13] proposed a high-fidelity experiment platform for mobile networks (FEP). A presented testbed interconnect ns-3, VirtualBox, Android and Docker, and the platform was tested with customized MPTCP-enabled Android.

The integration of ns-3 and Mininet was evaluated by Pakzad et al. [6] regarding experiment result accuracy, performance fidelity and scalability limits. The authors consider the great potential of the emulation platform, which shows a high degree of accuracy and fidelity of the achieved results as long as the scheduler of ns-3 can keep up with external real-time events. Arroyo et al. [2] study the limitations of the interaction of networks simulated in ns-3 with real nodes to analyze video streaming applications. The evaluation was presented considering several LXCs and a simulated WiFi network. The authors conclude that the number of devices deployed within the scenario substantially impacts the fidelity of achieved results than the traffic load.

[1] https://www.nsnam.org/.
[2] https://omnetpp.org/.
[3] http://coreemu.github.io/core/.
[4] http://mininet.org/.

However, none of the above works has focused on multipath emulation, including the detailed description of the employed testbed and the scalability limits. In this paper, we aim to address this critical gap.

3 Emulation of Multipath Connectivity

Our emulator integrates a Linux Container (LXC) into an ns-3 network simulation tool to provide realistic behaviour of emerging multipath solutions Fig. 1. Although this paper focuses on MPTCP protocol, one can also use this approach to integrate arbitrary software into the emulation. Firstly, this section gives a brief overview of the LXC technology, ns-3 platform and MPTCP protocol. Then, the emulation scheme is explained in detail.

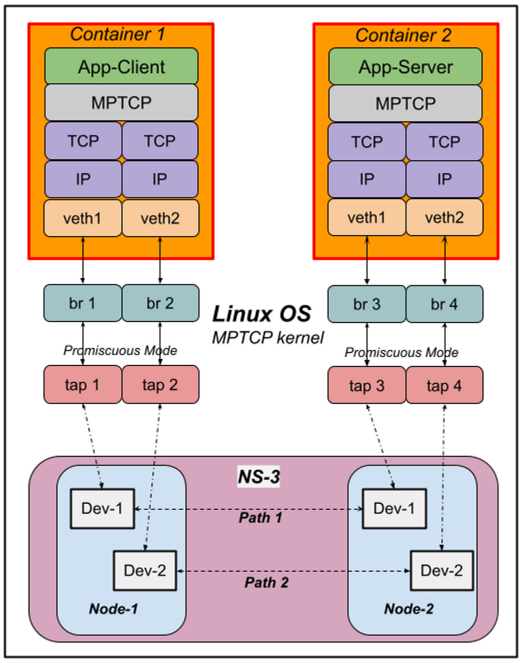

Fig. 1. Integration between LXC and ns-3 to emulate multipath connectivity

3.1 Background

Multipath connectivity is particularly useful in wireless networks and allows the concurrent use of different Radio Access Technologies (RAT). For example, if a mobile device has several interfaces (e.g., WiFi and 4G), it can simultaneously

use both networks to provide reliable connectivity. This work focuses on standardized Multipath TCP (MPTCP) [4], one of the most prominent candidate multipath protocols in the future 5G networks. MPTCP extends the standard TCP enabling transmission of a single connection (session) through several *subflows* Fig. 2.

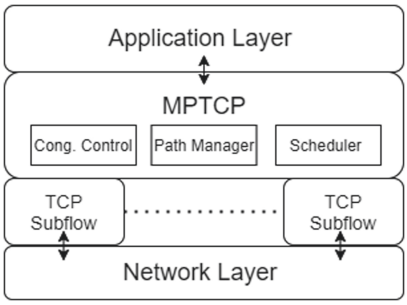

Fig. 2. Multipath TCP extension

MPTCP exploits multiple interfaces to aggregate the available bandwidth and enhance communication reliability. The advantage of the MTCP is that additional *subflows* may be created as soon as the mobile node has network coverage. Once a *subflow* is established, the application can send and receive data over multiple interfaces without disrupting the ongoing connection.

Linux Container (LXC) is a set of processes isolated from the rest of the operating system leveraging the *namespaces* and *cgroups* Linux kernel features. LXC is executed as simple processes with an independent view of system resources, a separate process tree and networking interfaces. Therefore, we select LXC as the primary node virtualization method since it has the flexibility of having several small virtual hosts implemented on a single physical system. Unlike full virtualization provided by Virtual Machines (VM), LXC is a lightweight virtualization technique that has reduced resource demands and shares the kernel with the host operating system (OS).

Network Simulator 3 (ns-3) is one of the most widely used open-source network simulators. It implements realistic modeling of different communication technologies, supports a wide variety of routing protocols and configuration of network parameters. It is a modular simulator based on the C++ language. Ns-3 enables the design, evaluation, and validation of various communication technologies with near real-life performance measurements.

NS-3 can work in so-called *real-time mode*, which allows network emulation, i.e., the integration of a simulation environment with real devices. When ns-3

runs in emulation mode, external devices can exchange packets with the simulated network in real-time using the *FdNetDevice* or *TapBridge* ns-3 modules. For instance, the link between Linux OS and ns-3 can be implemented using *tap* virtual network devices, allowing Layer-2 packet reception and transmission. Here, the *tap* device interacts with the *TapBridge* and is represented as a common *NetDevice* in the ns-3 environment.

3.2 Emulation Setup

MPTCP is implemented at the kernel-space of the Linux OS, working transparently for the user applications running on end-systems (e.g., client and server). To perform our evaluation, we added full support of Multipath TCP v0.94 to the Linux 5.4 kernel by using the publicly available patch for it.

LXC shares the same kernel space with the host OS providing a lightweight and isolated environment with its own filesystem, process hierarchy, network interfaces, and IP addresses. We create two MPTCP capable containers that represent client and server virtual nodes and can use multiple interfaces simultaneously. As shown in Fig. 1, our emulation framework incorporates the Linux bridges (*br*), tap devices (*tap*) and virtual interfaces (*veth*) in order to interconnect the corresponding container with the associated ns-3 *Node* in the simulator. It can be seen as a simple wired connection between two virtual interfaces that allow interaction with the deployed LXCs.

After LXC is binding with the *Node* inside ns-3, the ns-3 *Node* sends all the incoming network traffic to the LXC through a virtual *tap* interface connected to the *veth* of the container through a Linux *bridge*. Similarly, the LXC sends the outgoing traffic over the ns-3 network. All the required components (*bridge*, *tap*, *veth*) should be created and configured before the emulation session starts for the experiment to work correctly.

Ns-3 support gives the ability to generate network scenarios and simulate several types of wired/wireless communication technologies such as Ethernet, Wi-Fi, LTE, 5G, among others, with realistic physical layer signal propagation models in addition to different types of mobility models. In order to synchronise ns-3 simulation clock with the real-time (*wall*) clock an *real-time scheduler* should be used during experiments.

3.3 Implementation Details

This study focuses on multipath connectivity in heterogeneous wireless networks, such as WLAN and LTE. As for the LTE technology simulations, we have utilized the LENA open-source module provided by the ns-3 platform. It includes the radio protocol stack and LTE access mechanism defined by the *3GPP* standard that controls the connection between User Equipment (*UE*) and the eNodeB(*eNB*). In addition, *eNB* is connected via a point-to-point link to the Evolved Packet Core(*EPC*) network with a single Serving Gateway (*SGW*) and Packet Data Network Gateway(*PGW*).

In order to conduct our multipath experiments, we create two MPTCP-capable containers with two virtual interfaces (*veth*) each, linked to the ns-3 over Linux *bridge* interfaces Fig. 3. The combination of the *left* container and UE Node on ns-3, can be seen as one whole representing a client node. Accordingly, the *right* container and *Remote* node can be seen as one single host running application server logic.

The UE node has two wireless interfaces to perform multipath connectivity. One interface is connected to the LTE network via the *eNb*, and the other is connected to a WiFi network via Access Point (*AP*). The *Remote* node is connected to the EPC as well as to the AP via two dedicated wired links with the same characteristics.

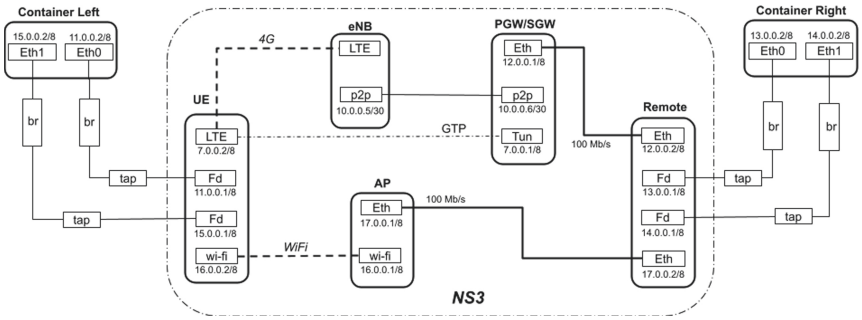

Fig. 3. Detailed scheme of emulated multipath environment

Many emulation based studies use the ns-3 *TapBridge* module since it allows the interface of the ns-3 node to be mapped directly into a tap device in the host OS. In that case, external devices can exchange Layer-2 network traffic with the simulator. However, not all ns-3 *NetDevice*s support this type of connection, e.g., LTE devices do not support bridging of *tap* devices. Instead, *FdNetDevice (FD)* was used to connect Linux containers with the simulated environment. *FD* can read and write traffic using a file descriptor associated with a tap device, a socket, or a user-space process. Also, we found *FD* to be more stable during the simulation experiments, providing accurate and reproducible results.

UE and Remote nodes relay traffic from LXC to the ns-3 network and vice versa through the *tap* device bound to the *FD*. This approach results in the additional hop and requires more careful configuration, but on the other hand, it allows the simulation of all network models currently available in ns-3. It is essential to mention that the upper bound limit for TCP throughput that can be achieved for this configuration with *FD* is around 70 Mbps and the delay induced by the additional hop is less than 0.5 ms. As mentioned by the ns-3 documentation[5], this limit is most likely due to the hardware's processing power involved in the tests.

[5] https://www.nsnam.org/docs/models/html/fd-net-device.html.

As shown in Fig. 3, there are two available ways by which UE can reach the Remote host: via LTE network (*path1*) and WiFi network (*path2*). To create a multipath communication environment, routing tables of LXC containers were configured such that all traffic from the first interface (*veth0*) are sent over *path1*, and all traffic from the second interface (*veth1*) are routed through *path2*. Furthermore, routing paths have to be defined for each ns-3 node to access the external networks Fig. 4.

```
routing table of PGW
Node: 4, Time: +0s, Local time: +0s, Ipv4StaticRouting table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
127.0.0.0       0.0.0.0         255.0.0.0       U     0      -      -   0
7.0.0.0         0.0.0.0         255.0.0.0       U     0      -      -   1
14.0.0.4        0.0.0.0         255.255.255.252 U     0      -      -   2
12.0.0.0        0.0.0.0         255.0.0.0       U     0      -      -   3
11.0.0.0        7.0.0.2         255.0.0.0       UGS   0      -      -   1
13.0.0.0        12.0.0.2        255.0.0.0       UGS   0      -      -   3

routing table of AP (wifi)
Node: 2, Time: +0s, Local time: +0s, Ipv4StaticRouting table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
127.0.0.0       0.0.0.0         255.0.0.0       U     0      -      -   0
16.0.0.0        0.0.0.0         255.0.0.0       U     0      -      -   1
17.0.0.0        0.0.0.0         255.0.0.0       U     0      -      -   2
15.0.0.0        16.0.0.2        255.0.0.0       UGS   0      -      -   1
14.0.0.0        17.0.0.2        255.0.0.0       UGS   0      -      -   2

routing table of UE
Node: 0, Time: +0s, Local time: +0s, Ipv4StaticRouting table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
127.0.0.0       0.0.0.0         255.0.0.0       U     0      -      -   0
16.0.0.0        0.0.0.0         255.0.0.0       U     0      -      -   1
7.0.0.0         0.0.0.0         255.0.0.0       U     0      -      -   2
15.0.0.0        0.0.0.0         255.0.0.0       U     0      -      -   3
11.0.0.0        0.0.0.0         255.0.0.0       U     0      -      -   4
0.0.0.0         7.0.0.1         0.0.0.0         UGS   0      -      -   2
14.0.0.0        16.0.0.1        255.0.0.0       UGS   0      -      -   1

routing table of Remote
Node: 1, Time: +0s, Local time: +0s, Ipv4StaticRouting table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
127.0.0.0       0.0.0.0         255.0.0.0       U     0      -      -   0
17.0.0.0        0.0.0.0         255.0.0.0       U     0      -      -   1
12.0.0.0        0.0.0.0         255.0.0.0       U     0      -      -   2
14.0.0.0        0.0.0.0         255.0.0.0       U     0      -      -   3
13.0.0.0        0.0.0.0         255.0.0.0       U     0      -      -   4
0.0.0.0         12.0.0.1        0.0.0.0         UGS   0      -      -   2
15.0.0.0        17.0.0.1        255.0.0.0       UGS   0      -      -   1
16.0.0.0        17.0.0.1        255.0.0.0       UGS   0      -      -   1
```

Fig. 4. Routing tables of principal nodes in simulation

In LTE networks, the end-to-end IP connectivity is provided within the EPC through the PGW. The PGW uses the destination IP address to identify a single UE and the eNB to which it is attached. Therefore, the 3GPP specification does not allow transmission of the packets with destination IP addresses different from registered UE. The PGW silently drops those packets because it would not know through which eNB it should be routed.

We first define a static routing at the PGW to access the external network outside the LTE to overcome this problem. Next, we replace the IP address bonded to UE on the PGW application with the IP address of the left container. Thus, all traffic in the LTE path destined to the container is routed to UE by the PGW, then UE reroutes it to the container via the FD interface.

4 Experimental Validation

This section presents the evaluation scenario, parameters setting, metrics used, validity tests, and corresponding results. The emulation environment was configured as realistic as possible to get reliable results. We have created different network topologies under various conditions to evaluate the proposed multipath emulation method. However, in this paper, we present only a short part of the obtained results since a detailed evaluation of MPTCP is out of the scope of this paper. All other scenarios we make publicly available[6] for research and use.

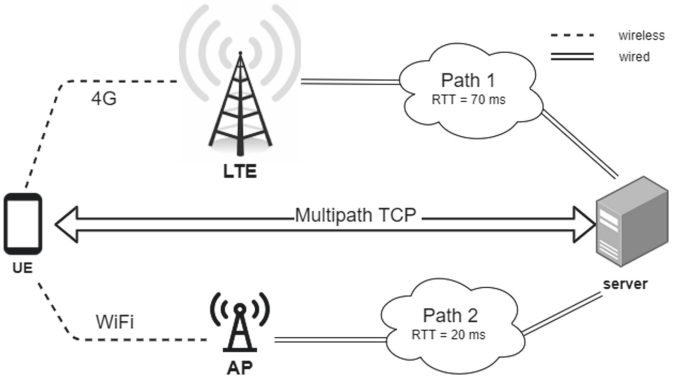

Fig. 5. The topology used for the emulation

4.1 Framework Configuration

Most protocols and channel configurations in our experiments use their default settings unless otherwise specified. The emulation parameters used for analysis are configured as listed in Table 1.

The *default* EPS bearer was used in the LTE network with the channel error models disabled. In addition, the number of resource blocks for uplink and downlink is set to 100 to achieve a bandwidth of 20 MHz. The eNb uses the FDD mode with a downlink frequency of 2120 MHz and an uplink frequency of 1930 MHz. In the simulation, the transmission power of 10 and 30 dBm was used by UE and eNB, respectively. Also, The *FriisSpectrumPropagationLossModel* is used to model signal fading.

[6] https://github.com/vandit86/ns3-scratch.

Table 1. Network configuration parameters

WiFi parameter	Value
Standard	802.11a
Phys. channel model	YansWifiChannel
Channel number	36
Channel width	20 MHz
Frequency	5180 MHz
Data mode	OfdmRate6Mbps
TX-power level	23 dBm
Rx-sensitivity	-101 dBm
LTE parameter	**Value**
eNb Tx Power/Noise figure	30/5 dBm
UE Tx Power/Noise figure	10/9 dBm
Uplink/Downlink freq	1930/2120 MHz
Acm model	PiroEW2010
Transmission mode	SISO
Number of resource blocks	100 (20 MHz)
Loss model	FriisSpectrumPropagationLossModel
MPTCP parameter	**Value**
Packet scheduler	default (RTT-based)
Path manager	full mesh
Congestion control	OLIA

Linux kernel implementation of MPTCP provides flexibility to select the path manager, congestion control algorithm, and packet scheduler to be used. In our experiments, we use the *"default"* RTT-based packet scheduler with the *"full mesh"* path manager, which ensures the creation of multiple sub-flows per connection. The original TCP configuration parameters were left unchanged. We employ the *OLIA* congestion control as it is less aggressive toward other TCP sessions over congested paths, which satisfies the design goals of MPTCP. The experiments were carried out on an AMD EPYC 7302 machine running a VM Linux Mint 20 (Kernel v.5.4.0) with MPTCP v0.95 configured with eight cores and 20 GB of RAM.

Once the experiment is deployed, we can "attach" to any container and perform metric tests with the common tools. During the emulation, data traffic is generated using the *Iperf3*, which allows the adjustment of TCP-related parameters like congestion control algorithm and receiver buffer size (*TCP RWND*). Furthermore, *Iperf3* works in client-server mode, measuring goodput at each end of the network path. *Tcpdump* and *Wireshark* tools are used to capture packets and analyze communication details, such as packet loss or round-trip time (RTT). In addition, the Linux Traffic Control may be used to change delay, jitter, bandwidth shaping, among others, of the LXC interface.

4.2 Simple Scenario

The network topology of the evaluation scenario is the one shown in Fig. 5. We simulate a scenario where a multihomed UE is connected to the server with two network interfaces, i.e. WiFi and 4G, each with unique IP addresses. Two transmission paths are used to evaluate the performance with heterogeneous links. The average RTT configured for *path 1* is 70 ms and for *path 2* is 20 ms. The WiFi AP and LTE-PGW are connected to the server with 100 Mbps wired links. All nodes remain stationary during the simulation.

To perform our experiments, we use the *Iperf3* tool to run MPTCP sessions between the UE and server with an unlimited load that guarantees link saturation. As can be seen on Fig. 6, MPTCP splits the generated traffic amongst two *subflows*. It aggregates the bandwidth of available links, and we get increased overall throughput as expected. The average RTT on *path 2* grows rapidly due to the saturation of the wireless AP capacity Fig. 7.

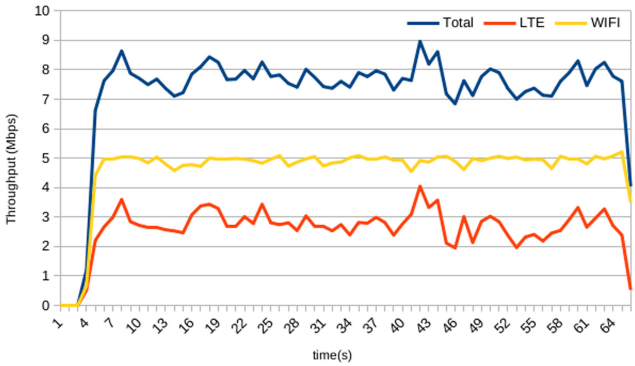

Fig. 6. Registered throughput with enabled MPTCP

4.3 Mobility Scenario

Multipath TCP supports seamless handovers by design, i.e., traffic flow switches from one network without disrupting application-layer data flow. Therefore, MPTCP is considered to be a promising approach in mobility scenarios, as it can migrate connections between different wireless interfaces.

One of the main features of the ns-3 based emulation is the ability to create mobility scenarios depending on the need of the experiment. We create a simple mobility scenario where the UE moves away from WiFi AP, with a constant velocity of 2 m/s, to trigger the handover procedure. Figure 8 shows the average downlink throughput for UE with rate-limited traffic, i.e., the *Iperf3* data rate was limited to 1 Mbps.

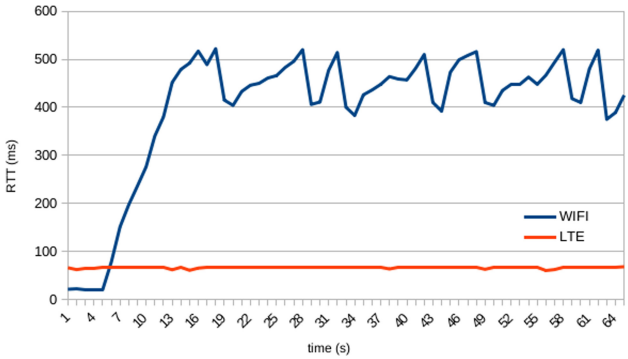

Fig. 7. Measured delay on each path

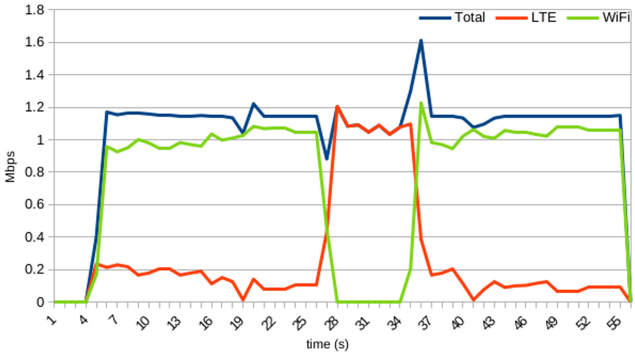

Fig. 8. Throughput on each interface during the handover

A UE is initially connected to both cellular and WiFi networks while receiving data simultaneously. The default MPTCP scheduler selects a path that guarantees the lowest delay for each segment. Although MPTCP transmits data via all available paths, almost all the traffic is sent via WiFi network (*path 2*). It can be seen that when the UE goes out of the coverage of the WiFi network, MPTCP can detect the connection loss and switch traffic flow to the *path 1* (LTE network), thus guarantee throughput required by the application. Notice that the application layer is unaware that the WiFi link goes down since data transmission proceeds on an alternate path.

Next, UE moves back to AP, and after some time, as the WiFi connectivity is restored, the MPTCP scheduler selects the path with the lowest RTT as the primary network path to send data. With this example, we evaluate the performance of MPTCP during the handover and show that our emulation technique responds as expected to the mobility aspect.

5 Limitations

As mentioned previously, during the tests with the FdNetDevice we have found a maximum achieved TCP throughput of approximately 70 Mbps. This result confirms the limits declared in ns-3 documentation. In this section, we evaluate the scalability of the described emulation method by increasing the complexity of the network topology. Afterwards, we discuss the limitations exhibited by the simulator and point out the possible solutions to overcome these constraints.

5.1 Scalability Evaluation

Low scalability is a common problem for all emulation-based experiments. Due to the limited hardware resources, emulators cannot reproduce the correct behaviours of a real network with a large topology and high traffic load. The fidelity of the emulator's results relies on physical resources of the host system, such as memory and CPU.

Ns-3 itself is an event-driven single-threaded[7] simulator that uses one CPU core to execute all the simulated events that are enqueued according to the simulation time they take place. Ns-3 applies a *real-time scheduler* in emulation mode that synchronizes the simulation clock with the hardware clock (*wall clock*). Thus, the events in the simulation are processed according to the system time.

However, in large scale scenarios, numerous events can occur within a short period. The simulation becomes time-consuming, so the execution of the enqueued event may be delayed. This delay, or *"jitter"*, is defined as a difference between the time an event should be handled and when it actually does. Ns-3 can control the fidelity of emulation if the real-time scheduler is configured to run in *"HardLimit"* mode. Thus, the simulator continually monitors *jitter* value. When the *HardLimit* mode is enabled, the experiment will abruptly stop if the *jitter* becomes too large (100 ms by default).

To investigate the scalability of our experiments, we studied the impact of the number of LTE devices in simulation by measuring the *jitter* of the simulator and RTT between two LXC. The same static scenario was employed, as described in the previous section Fig. 5. Several additional LTE devices created within the simulation are connected to the same eNb, downloading a 512 Kbps UDP data stream from a separate server connected to the same PGW.

Similar to our first experiment, we created the MPTCP session by performing an *iperf3* test with an unlimited load between two LXCs for 30 s. Figure 9 shows the evolution of the *jitter* during one simulation for a different number of connected LTE devices. The results indicate that the ns-3 can maintain a low *jitter* value when the number of LTE devices within the scenario is below twenty. However, with the 25 LTE devices, the processing of simulation events cannot keep up with the wall clock. Thus the emulation cannot guarantee a reasonable accuracy of the experimental results.

[7] *FdNetDevice* and *TapBridge* use auxiliary threads.

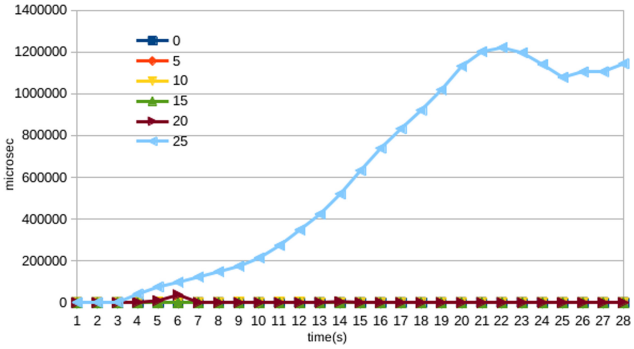

Fig. 9. The impact of LTE devices on simulation delay (*jitter*)

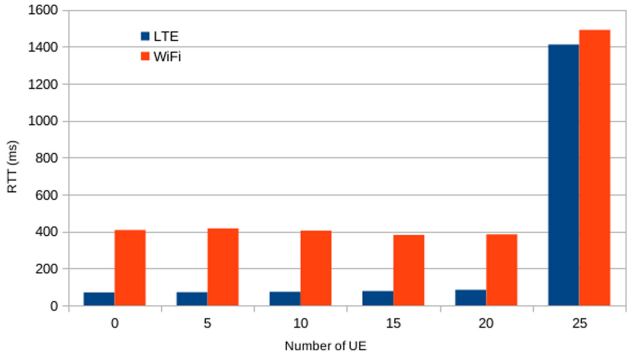

Fig. 10. RTT between LXCs with different number of LTE devices

Figure 10 confirms this conclusion by evaluating the RTT between two LXC's on both paths (LTE and WiFi). It shows that the number of network devices within the ns-3 simulator can limit the fidelity of the produced results by raising the *jitter*. This happens since the CPU cannot keep up with the increasing computational load; thus, the difference between the simulator and system clocks is growing. Therefore, it is important to find a limit that guarantees acceptable *jitter* for a particular emulation scenario.

One can use a faster host system for the network emulation (i.e., more CPU, larger RAM) in order to improve the scalability of the container-based experiments. The time dilation technique [5] is another promising approach to deal with the system's scalability issues. It's consists of a notion of *virtual time* that allows scaling system time for some selected processes. Thus, the real-time clock value is replaced by the virtual time clock tied to each process. For instance, when LXC is attached to a virtual clock scaling by a factor of N, it is considered to run N times more slowly than on the actual execution platform. Thus, by simply slowing the advancement of virtualized time, LXC will not exceed the

rate at which ns-3 can advance. However, some modification to the Linux kernel is needed to use this approach.

6 Conclusions and Future Work

This work presents an emulation technique to analyze multipath data delivery over heterogeneous wireless links. It exploits the realistic network behaviour offered by the ns-3 platform and the functioning of real nodes provided by the Linux containers.

In particular, we have carried out an illustrative study on the behaviour of MPTCP protocol when used over wireless networks with differing characteristics., such as Wi-Fi and LTE. First, we have described in detail the emulation scheme, which allows the interaction of ns-3 with real systems. The proposed emulation method was validated for various scenarios in terms of accuracy and fidelity of achieved results. The expected MPTCP behaviour and experiment outcomes were registered through the experiments, validating the presented method as a flexible way to test emerging multipath protocols and applications.

Finally, we have discussed the limitations exhibited by the presented emulation technique. It is shown that the processing load of the ns-3 based emulation has a significant impact on the capacity of the simulator to perform a real-time execution. The emulator can produce sufficiently accurate results if the number of physical devices is low. The simulation delay (*jitter*) was used as a primary fidelity indicator.

Our current work explores the proposed emulation technique for a wide range of wireless networks (e.g., 5G). As future work, we plan to extend the presented scheme with realistic mobility models by using a traffic simulator. Furthermore, future work should improve the capacity of the emulator to perform a real-time execution to provide support for large-scale experiments.

References

1. 3GPP: Ts 23.501, system architecture for the 5g system, v16.4 (2020)
2. Arroyo, J., Diez, L., Agüero, R.: On the use of emulation techniques over the ns-3 platform to analyze multimedia service QoS. In: Agüero, R., Zinner, T., García-Lozano, M., Wenning, B.-L., Timm-Giel, A. (eds.) MONAMI 2015. LNICST, vol. 158, pp. 196–208. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26925-2_15
3. Elattar, M., Wendt, V., Neumann, A., Jasperneite, J.: Potential of multipath communications to improve communications reliability for internet-based cyberphysical systems. In: IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2016-November (2016). https://doi.org/10.1109/ETFA.2016.7733536
4. Ford, A., Raiciu, C., Handley, M.J., Bonaventure, O.: TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824 (2013). https://doi.org/10.17487/RFC6824, https://rfc-editor.org/rfc/rfc6824.txt

5. Lamps, J., Nicol, D., Caesar, M.: Timekeeper: a lightweight virtual time system for Linux (2014). https://doi.org/10.1145/2601381.2601395
6. Pakzad, F., Layeghy, S., Portmann, M.: Evaluation of Mininet-WiFi integration via ns-3. In: 26th International Telecommunication Networks and Applications Conference, ITNAC 2016, pp. 243–248 (2017). https://doi.org/10.1109/ATNAC.2016.7878816
7. Petersen, E., Antonio To, M.: DockSDN: a hybrid container-based software-defined networking emulation tool. Int. J. Network Manage. (2021). https://doi.org/10.1002/nem.2166
8. Petersen, E., Cotto, G., To, M.A.: Dockemu 2.0: evolution of a network emulation tool. In: 2019 IEEE 39th Central America and Panama Convention, CONCAPAN 2019 2019-November (2019). https://doi.org/10.1109/CONCAPANXXXIX47272.2019.8977002
9. Portabales, A.R., Nores, M.L.: Dockemu: extension of a scalable network simulation framework based on docker and ns3 to cover IoT scenarios. In: SIMULTECH 2018 - Proceedings of 8th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (Simultech), pp. 175–182 (2018). https://doi.org/10.5220/0006913601750182
10. Sabbah, A., Jarwan, A., Issa, O., Ibnkahla, M.: Enabling LTE emulation by integrating CORE emulator and LTE-EPC Network (LENA) simulator. In: IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC, 1–6 October 2017 (2018). https://doi.org/10.1109/PIMRC.2017.8292642
11. Suer, M.T., Thein, C., Tchouankem, H., Wolf, L.: Multi-connectivity as an enabler for reliable low latency communications - an overview. IEEE Commun. Surv. Tutor. **22**(1), 156–169 (2020). https://doi.org/10.1109/COMST.2019.2949750
12. To, M.A., Cano, M., Biba, P.: DOCKEMU - a network emulation tool. In: Proceedings - IEEE 29th International Conference on Advanced Information Networking and Applications Workshops, WAINA 2015, pp. 593–598 (2015). https://doi.org/10.1109/WAINA.2015.107
13. Zhang, T., Zhao, S., Cheng, B., Ren, B., Chen, J.: FEP: High fidelity experiment platform for mobile networks. IEEE Access **6**, 3858–3871 (2018). https://doi.org/10.1109/ACCESS.2018.2793943