SMARTHO: A Network Initiated Handover in NG-RAN using P4-based Switches

Phanindra Palagummi and Krishna M. Sivalingam
Dept. of CSE, Indian Institute of Technology Madras, Chennai, INDIA
Emails: {cs15s042@cse.iitm.ac.in, skrishnam@iitm.ac.in, krishna.sivalingam@gmail.com}

***Abstract*—This paper deals with the design of protocols for 5G-and-beyond wireless networks. In particular, it considers a Next Generation RAN (NG-RAN), where the Base Band Unit (BBU) functions are split across a Central Unit (CU) and a Distributed Unit (DU). This paper proposes the use of Programming Protocol independent Packet Parsers (P4)-based switches between the CU and DU for processing packets exchanged between the two. We demonstrate the smart handover (SMARTHO) scheme for a mobile User Equipment (UE) that traverses a known fixed path. The idea is to perform the resource allocation in subsequent macro-cells in advance of the user's movement, by having the P4 switch spoof the behaviour of the UE. Based on an implementation using Mininet and P4BM software switches, it is seen that the proposed method leads to around 18% and 25% reduction in handover time, for two- and three-handover sequences, respectively.**

I. Introduction

This paper deals with improving handover performance in 5G Wireless networks, using the programmable data plane switch paradigm. The Next-Generation RAN (NG-RAN) architecture is considered. Here, real-time (RT) functions are deployed near the antenna site to manage air interface resources using the Distributed Units (DU). At the same time, non-real-time (NRT) control functions are hosted centrally in the Central Unit (CU). This split functionality is now part of the 3GPP specification [1]. The services offered by the CU and DU can be virtualized in software and placed in Commercial off-the-shelf (COTS) servers, using Network Function Virtualizaton (NFV) [2]–[5].

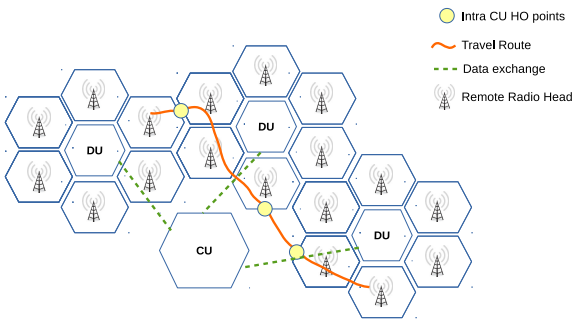


Figure 1: Intra-CU Handover.

In this paper, we design a solution for handling mobile device handover, using programmable data-plane switches based on P4 programming language [6]. P4-based switches are used to parse the packets and to invoke additional actions defined by the protocol designer. These actions can be made to perform simple forwarding or can aid functional behaviour of the system. P4 switches are expected to perform better than traditional L2-L3 or OpenFlow switches due to the additional functionality enabled.

In particular, we propose a Smart Handover (SMARTHO) process for fixed-path mobile devices, such as LTE users in a train, drones, predictable mobility devices, etc. is considered. In particular, the handover is considered for Intra-CU HO from one Remote Radio Head (RRH) to another RRH in a different DU, but connected to the same CU. This scenario is shown in Figure 1. A resource allocation scheme that reserves resources ahead of the UE in its path is proposed. The solution is implemented using a P4-based switch introduced between the CU and the DU. We use the P4 switch to spoof the behaviour of User Equipment (UE) and perform the resource allocation in advance. Using an implementation based on Mininet and P4BM software switch, it is seen that the proposed method results in an 18% and 25% improvement in the sequence of two and three handovers, respectively. We have considered the Intra-CU handover in this paper; however, this idea can be applied to other HO processes specified in 3GPP [1].

II. Proposed SMARTHO Framework

This section presents the proposed resource allocation framework.

A. Resource allocation during mobility

In a wireless network, user equipment (UE) handover from one cell to another cell is an important aspect of mobility management. In this paper, we consider intra-DU handover within a single CU. Typically, there are 3 phases in a handover (HO) process: Preparation, Execution and Completion.

The preparation phase deals primarily with resource allocation for the UE in the next DU. In this phase, the Measurement Report (MR) message from the Source_DU will be transmitted to the CU, which would select the Target_DU for the HO. The CU will send the HO request (UE Context Request), containing Target-DU-ID, UE context info & UE History Information. When the Target_DU receives the HO request, it begins handover preparation to ensure seamless service provision for the UE. The Target_DU will respond with setting up the Access Stratum (AS) security keys, uplink bearers connecting to the backhaul, reserve Radio Resource Control (RRC) resources to be used by

the mobile device over the radio link and allocates Cell-Radio Network Temporary Identifier. Once the resources are allocated by the Target_DU, a response message called the “UE context setup response” is sent to the CU. Once handover preparation between the two DUs (Source_DU and Target_DU) is completed, the execution phase and completion phase will happen to successfully perform the UE HO, the 3GPP standard presents the procedure in [1].

In this paper, we deal with the preparation phase, by proposing a advanced resource allocation scheme along a set of pre-defined DU nodes.

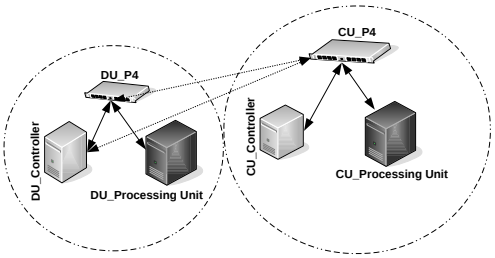


Figure 2: Proposed SMARTHO Framework.

B. SMARTHO Architecture and Components

In the proposed smart handover framework (called SMARTHO), we introduce programmability into the data plane without changing the existing architectural framework.

The main components of the proposed CU and DU architecture are COTS compute servers, P4 switches, and a Network Controller. The compute servers will implement the functions of CU, DU and Network Controller. The interconnections and components of SMARTHO framework are shown in Figure 2.

The network controller at the CU (CU_Controller) will store the “UE Mobility Information” and the “UE Context Information”. The network controller at the DU (DU_Controller) will store the RRC Connection Reconfiguration (RRCCR) message. The P4 switches will process the messages from processing units and perform the SMARTHO process, by sending appropriate instruction messages to CU and DU Controllers.

The first handover of a given UE will set the UE context information in the CU_Controller. After the first HO is completed, the SMARTHO initiation will happen which automates the subsequent handovers. The P4 switches in CU (CU_P4) and DU (DU_P4) will send the instruction messages to CU_Controller to access the mobility information and DU_P4 switches to store the RRCCR message respectively.

The entire 3GPP process with P4 switches in CU with sequence of messages is shown in Figure 3. In the first handover, P4 switches will parse the incoming packets and negotiate with local storage at CU to determine if the UE is having a fixed path. If so, after the completion of first HO, the P4-switch will generate "UE Context Setup Request" message and forward it to the Target HO entities,

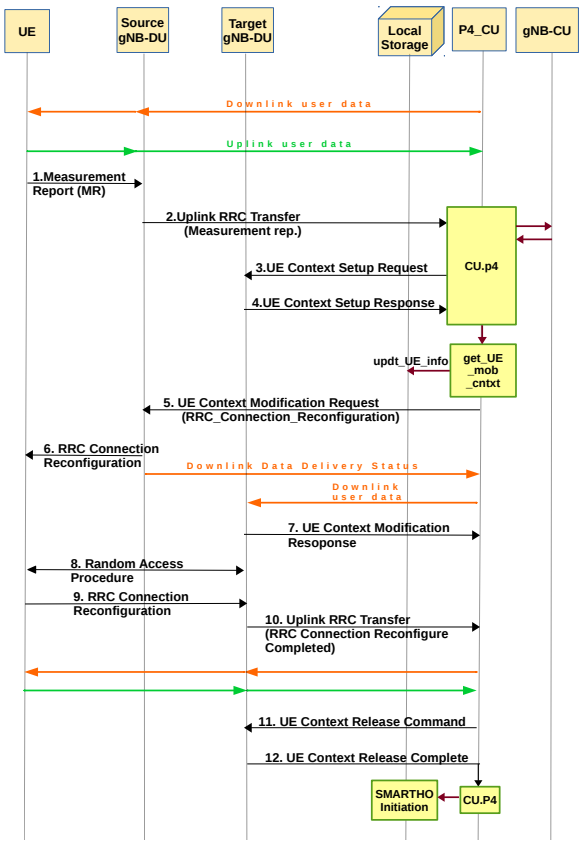


Figure 3: Sequence diagram of intra-CU Handover.

on behalf of the UE. This is referred to a Smart Handover (SMARTHO) in this paper.

C. Architecture and Design of P4 switches

There are several switch architectures, that support protocol independent switches. In this paper, we use the Very Simple Switch (VSS) Architecture [7]. VSS has basic programming blocks needed for protocol independent switch, which are sufficient to implement the SMARTHO process.

Next generation mobile networks have a complex packet structure, designing a parser for entire packet structure would overload the functionality of the P4 switch, increasing the complexity of the parser. Also, the structure of the packets for mobile networks would depend on the state information. P4 switches are not scalable to parse such packets as of now. To simplify this process, we design a *tag*-based approach to identify necessary packets for SMARTHO. The tag will be added by the processing units or controller.

The P4-switches in the SMARTHO model handles three types of packets:

1) User packets of the 5G system: These packets are simulated as ICMP packets encapsulated over the tag, the forwarding is done using tag information.
2) Control packets for HO: In case of Intra CU HO, the entire HO process has twelve control messages exchanging, shown in Figure 3. These packets have

to be identified and will be sent to P4 switches or controller for processing.

3) Instruction packets: These packets will either instruct the P4 switch to initiate specific methods in Match-Action control block or the controller to store/retrieve the data.

Three special data structures have been defined to store the necessary state information: Mobility Table (MT), Controller Cache (CC) and RRC Table (RRCT). MT and CC will reside in CU_Controller and RRCT will reside in DU_Controller. The details are not specified due to lack of space.

All the three types of packets are encoded with the respective tags. The differentiation is done based on the extracted tag and examining the valid/invalid bit [7].

III. Implementation Details

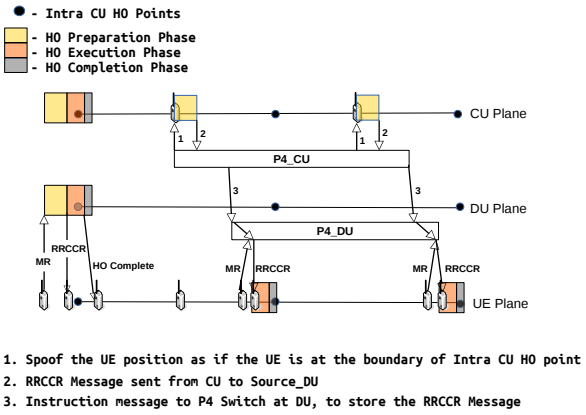


Figure 4: Operation of handover process, using P4 switches

As described earlier, we perform the HO preparation phase in advance of the UE movement, in order to decrease the overall HO time. Figure 4 presents the working details of SMARTHO, with a sequence of three Intra-CU handover (HO) points. The operation of SMARTHO has three phases: SMARTHO-Data Setup, SMARTHO-Initiation and SMARTHO-Completion, as described below.

A. Data Setup

The current context of the UE has to be retrieved, before the start of the SMARTHO process. The context information of UE can be retrieved from the "UE Context Setup Request" message, which is exchanged between CU and Target_DU as shown in Figure 3. The UE context information is updated in the data table CC.

This message is sent to the CU_P4 switch. The CU_P4 switch can identify the control packets for HO, this can be done by changing the code at CU part, to send the HO message "UE Context Setup Request" with tag value 0x03, as discussed in Section II-C. The CU_P4 will identify the tag and execute a routine to send the message set_ue_context to the CU_Controller, which will store the UE context information in CC. The set_ue_context contains the UE identifier, Aggregate Maximum Bit Rate (AMBR) for the UE, and other relevant information.

The P4 switch at CU identifies the set_ue_context message and forwards it to the CU_Controller. Once the CU_Controller receives the set_ue_context message, it updates its CC using a packet sniffer at the controller.

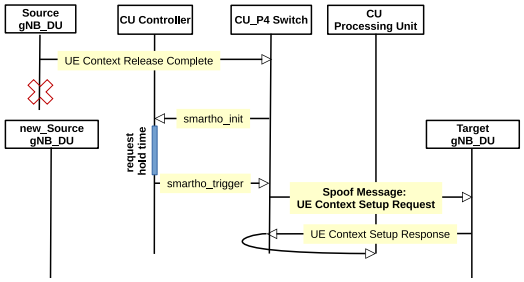


Figure 5: Initiation sequence of SMARTHO.

B. SMARTHO - Initiation

The initiation of the SMARTHO process is shown in Figure 5. The Source_gNB_DU sends the "UE Context Release Complete" message with a tag value of 0x0*c* to the CU_P4. This switch parses the packet and identifies the message with the tag value and initiates the process of SMARTHO. This is done by sending the smartho_init message to the CU_Controller with a tag value of 0x02. The purpose of the smartho_init message is to retrieve the address of Target_gNB_DU from MT for the next HO and *delay* information of the UE. This delay value is used to hold the process before starting the preparation phase.

The CU_Controller runs a packet sniffer at the ingress port. When a smartho_init message is received, the sniffer runs a background process. This will send the smartho_trigger message to the CU_P4 switch with a tag value of 0x02.

The smartho_trigger message is the basis to send the spoofed "UE Context Setup Request" message for the next HO to the Target_gNB_DU. This will initiate the HO preparation phase for the subsequent HO.

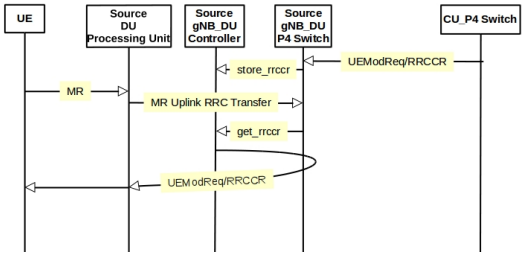


Figure 6: Completion sequence of SMARTHO.

C. SMARTHO Completion

The final phase of SMARTHO is to handover the UEModReq/RRCCR message as a response to UE MR, as shown in Figure 6. The UEModReq/RRCCR message that is sent

from CU to Source_DU is intercepted by the Source_DU_P4 switch. This would instruct the Source_DU_Controller to store UEModReq/RRCCR message. This message contains the UEModReq information that is updated in the RRCT of DU_Controller.

When a UE sends the MR to Source_DU (S_DU), the S_DU would respond with "Uplink RRC Transfer message" to CU. The DU_P4 switch intercepts this message and instructs the controller to get the UEModReq/RRCCR message which is forwarded to UE as shown in Figure 6.

IV. Performance Evaluation

The proposed SMARTHO framework was implemented in the mininet emulation environment [8], where mininet-based hosts emulate the CU, Source_DU, Target_DU & RRH. Mininet hosts are connected using P4 switches, developed using the P4 behaviour model (P4BM) with VSS model architecture, [9]. Raw data packets are created using the *scapy* tool [10], that sends a continuous sequence of raw data packets from one host to another. User and control traffic are generated to simulate the mobile traffic and measure the HO performance. User traffic is represented using ICMP ping packets over a tag. Control traffic is generated to simulate the HO procedure, packets are created with customized headers containing UE identification, over the tag. The tag of the control packet is also used as the identification for the HO message. We considered a tandem of Intra-CU HOs, sending user packets between the RRH and CU. User packets are generated as parallel ping process in RRH to simulate varying arrival rates. The inter-arrival time between Intra-CU HO was exponential. The HO time is measured from the moment RRH has sent the MR message to the Source_DU, to the RRCCR message received at RRH indicating the HO is completed.

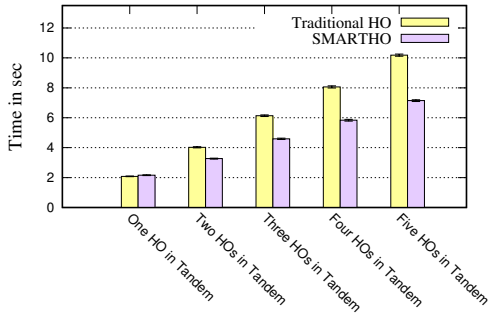
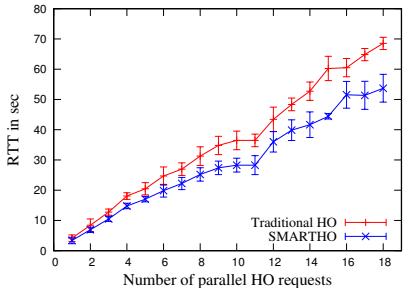
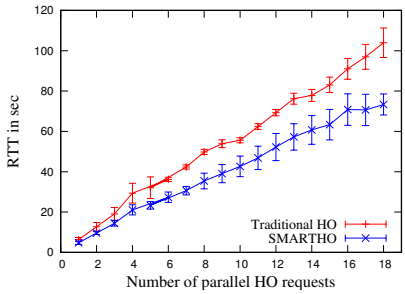


Figure 7: Performance of Intra-CU HOs in tandem.

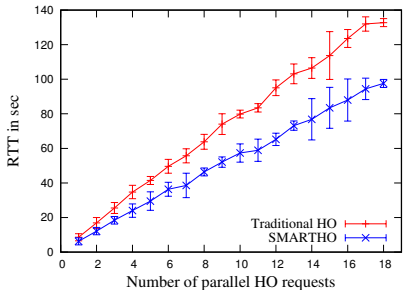
Figure 7 presents the performance for handover time on a single UE. The graph shows the total time spent for handover. As seen, the SMARTHO process performs better than the traditional HO process. There is no improvement of HO response time with single HO, this is because the SMARTHO process will perform the data setup in first HO and automates the subsequent HOs. Improvement of 18% for two tandem HOs and 25% for three tandem HOs is achieved and this improvement will increase as the tandem of HOs



(a) Two Intra-CU HO sequence



(b) Three Intra-CU HO sequence



(c) Four Intra-CU HO sequence

Figure 8: Performance comparison of traditional handover and SMARTHO in terms of Intra CU-HO time.

increases. This is because the overall time spent on HOs will proportionally decrease as the HO preparation phase is done in advance for all the subsequent HOs.

In the next study, we increased the intensity of HO requests, with multiple UEs requesting handovers. Figure 8 presents the response time. The results show that the SMARTHO process performs better than the traditional HO process, with higher improvements with increase in the number of transmit nodes.

Acknowledgments: We thank Mr. Karthik Karra, Dr. Manikantan Srinivasan, and Dr. C.S Ganesh (IIT Madras) for sharing their insights and feedback. Part of the work was supported by an IIT Madras IRDA-2017 award.

V. Conclusions

In this paper, we have presented the use of P4-based dataplane switches to improve handover efficiency, in a wireless network. The proposed approach has been studied using a Mininet implementation. The experimental results show that the proposed SMARTHO approach does have benefits over the traditional handover process.

REFERENCES

[1] 3GPP, "NG-RAN;Architecture description," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.401, 06 2018, version 15.1.0.

[2] I. Giannoulakis, E. Kafetzakis, G. Xylouris, G. Gardikis, and A. Kourtis, "On the applications of efficient NFV management towards 5G networking," in *Proc. of Intl. Conf. on 5G for Ubiquitous Connectivity (5GU)*, 2014, pp. 1–5.

[3] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC)," *IEEE Network*, vol. 28, no. 6, pp. 18–26, 2014.

[4] S. Abdelwahab, B. Hamdaoui, M. Guizani, and T. Znati, "Network function virtualization in 5G," *IEEE Communications Magazine*, vol. 54, no. 4, pp. 84–91, 2016.

[5] J. Costa-Requena, J. L. Santos, V. F. Guasch, K. Ahokas, G. Premsankar, S. Luukkainen, O. L. Pérez, M. U. Itzazelaia, I. Ahmad, M. Liyanage *et al.*, "SDN and NFV integration in generalized mobile network architecture," in *Proc. European Conference on Networks and Communications (EuCNC)*, 2015, pp. 154–158.

[6] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[7] *P4 16 Language Specification*, The P4 Language Consortium, 5 2017, ver. 1.0.0.

[8] "Mininet," mininet.org, 2018.

[9] P4 language Consortium, "P4 behaviour model," https://github.com/p4lang/behavioral-model, 2013.

[10] SCAPY, "SCAPY - A Python packet crafting tool," https://github.com/secdev/scapy, 2016.