SODALITE: SDN Wireless Backhauling for Dense 4G/5G Small Cell Networks

August Betzler, Daniel Camps-Mur, Eduard Garcia-Villegas, Ilker Demirkol, and Joan Josep Aleixendri

***Abstract*—Dense deployments of Small Cells are key to fulfill the capacity requirements of future 5G networks. However, two roadblocks to the adoption of Small Cells are i) the limited availability and the cost of sites with wired backhaul resources, and ii) the complexity to manage a dense deployment of wireless backhaul nodes. Towards these challenges we propose SODALITE, a novel system that applies Software Defined Networking (SDN) to a wireless backhaul network. We present how SODALITE can be integrated to 3GPP's 4G and 5G architectures, and show the feasibility of SODALITE through LTE network testbed experiments. We substantiate the scalability of SODALITE through stochastic studies using real-life traffic traces from an LTE network and discuss the effects of cell densification and 5G system architecture on these studies. Further, a reliable backhauling solution for wireless links is introduced in SODALITE through SDN-enabled mechanisms that are capable of reconfiguring the data plane upon a link failure detection. Its reliability is shown through experiments on a LTE network testbed, and studied thoroughly via rigorous simulations and network emulator evaluations. As a result, we claim that SODALITE is a promising carrier-grade system to manage a wireless Small Cell backhaul.**

***Index Terms*—4G/5G Small Cells, wireless backhaul, SDN, fast re-route.**

I. Introduction

ADDRESSING the expected increase in mobile data demand is one of the main challenges of the mobile industry. The innovations needed to provide the required capacity are being developed as part of the future 5G, and complementing macro-cell sites with massive deployment of indoor and outdoor (street-level) Small Cells (SCs) is considered as the most promising approach to increase area capacity.

The main challenge of this architecture is the scarcity of backhaul resources. In particular, fiber deployment is unavailable in many SC sites and it is costly, making wireless SC backhauling solutions essential. Fig. 1 depicts an example dense SC deployment in an urban scenario with SCs installed in lamp-posts, macro-cell sites at rooftop level, and a wireless backhaul connecting SCs. However, due to the lossy and time-variant nature of wireless links, a resilient backhauling solution is necessary. Current deployments assume an over-provisioned backhaul; however, in future dense SC deployments, the wireless backhaul is likely to become the bottleneck, if it relays traffic from multiple SCs, or if spectrum is shared between access and backhaul. As illustrated in Fig. 1, the wireless backhaul segment will connect SCs to fiber attachment points, typically located at the macro-cell site, or in a street cabinet. Hence, in practice we expect a relatively small wireless backhaul segment (< 5 hops as in [1]), whereby a given SC could be connected to more than one fiber attachment point.

Several technologies can constitute the wireless backhaul [2], such as: i) unlicensed mmWave based on IEEE 802.11ad/ay, which provides multi Gbps data rates at 50 meters, but requires Line of Sight (LoS), ii) lightly licensed E-Band links providing similar capacities but with longer ranges, iii) Sub-6 GHz technologies, which, if based on IEEE 802.11ac, can provide hundreds of Mbps per-link without LoS [3], and even higher rates with IEEE 802.11ax [4], iv) proprietary point to multi-point solutions operating at micro-wave frequencies, and v) a hybrid combination of those.

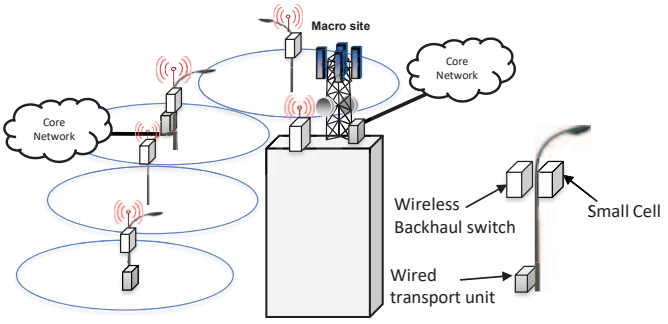


Fig. 1. Wireless backhaul physical layout.

Further key requirements in the wireless SC backhaul are flexibility and programmability, where an operator should be able to re-configure the data plane, e.g., if new SCs are installed, or switched off to reduce energy consumption. Operators should also be able to enforce per-user policies in the wireless backhaul in case of congestion, and the control plane should quickly re-route traffic in case of link failures.

In this paper we present SODALITE, a system to implement traffic engineering in the wireless SC backhaul, that addresses the aforementioned requirements. SODALITE is built on Software Defined Networking (SDN) technology to achieve the flexibility requirement. Its architecture defines clear interfaces to integrate with the 4G and 5G network architectures defined by 3GPP. Thanks to an SDN-based control of backhaul nodes, SODALITE supports the definition of per-user/per-application/per-slice policies in the wireless transport segment. The presented SODALITE architecture is implemented and validated in an LTE network testbed using standard open-source solutions such as Openflow (OF), showing its feasibility. Further, we validate the scalability of SODALITE through stochastic evaluations using traffic traces obtained from an operational LTE network. The projection of this scalability study to future 5G networks is also presented.

The unreliable nature of the wireless links is another challenge of a wireless SC backhaul. To achieve a reliable solution in such environments, SODALITE defines a hybrid SDN control plane, which combines a centralized computation of main and backup paths at the SDN controller, and a distributed agent embedded in the SODALITE nodes to provide fast path recovery. Two novel path allocation policies are evaluated

A. Betzler (august.betzler@i2cat.net), D. Camps-Mur, and JJ. Aleixendri are with the i2CAT Foundation. E. Garcia-Villegas, and I. Demirkol are with Universitat Politecnica de Catalunya, all in Barcelona, Spain.

in terms of capacity and reliability using an event based simulator. In addition, accurate recovery delays are evaluated using a network emulator and a real LTE testbed. We rely on 4G devices to perform our evaluation for practical reasons, but claim that our results are applicable to 5G.

To the best of our knowledge, SODALITE is the first SDN-based wireless backhauling solution in the literature that: i) is tightly integrated to 4G and 5G networks, ii) is validated through real cellular network testbed experiments, and iii) has its scalability proven using real network traces. Hence, we posit that SODALITE is the first carrier-grade SDN-based wireless backhauling solution, which promises to tackle the small cell densification challenges.

The rest of this paper is organized as follows. Section II positions the contributions of SODALITE with respect to the related work. Section III describes the SODALITE architecture. Section IV describes the centralized path allocation algorithms, and the distributed agent for fast path recovery. Section V provides the scalability analysis. Section VI provides an experimental evaluation of SODALITE's forwarding and path recovery policies. Finally, VII concludes the paper.

II. Related Work

Our work intersects with previous works in the areas of: i) SC transport architecture, ii) SDN forwarding in multi-channel mesh networks, and iii) fast recovery in wireless networks.

SC transport network architectures. The authors in [5] present a high level architecture for 5G ultra dense SC networks, and elaborate on the use of various wireless technologies in the fronthaul. The authors in [6] and [7] propose, respectively, high level interfaces between a transport controller and the 4G network to optimize resource allocation, and between RAN and transport for access-backhaul coordination. In [8], authors suggest to add GPRS Tunneling Protocol (GTP) match/action capabilities to OF, although the focus is to virtualize the data plane of a mobile gateway, and not the implications of the GTP tunnels on the transport network. In addition, the authors in [9] propose a low delay in-band scheme to backhaul the X2 interface between LTE SCs, which given its limited bandwidth is not applicable to the scenarios addressed in this paper. SODALITE advances the state of the art by defining in detail an interface between a transport SDN controller and the control plane of a 4G or 5G network, and by studying the scalability of such interface, in terms of rules maintained in each node and signaling overhead, using real traces obtained from an operational network.

SDN forwarding in multi-channel mesh networks. Application of SDN in multi-hop wireless networks has been studied in existing literature. For example, in [10], authors propose a hybrid architecture comprising an OLSR daemon configuring a control network and a centralized OF solution to configure the data plane, while demonstrating a simple gateway balancing policy. Work in [11] shows the separation of control and data planes for an in-band control plane. In [12], authors demonstrate the feasibility of using SDN switches in multi-hop networks of constrained devices, while demonstrating two different forwarding policies. The main contribution of SODALITE in this area is the definition of two novel policies that, unlike previous works, allocate primary and backup paths for each backhaul flow, and dynamically optimize forwarding according to measurements reported by backhaul nodes.

Fast link recovery. Solutions for failure recovery in wireless mesh networks have been approached using distributed protocols. In SDN, fast local re-reroute is a recognized problem that is often addressed with local fast failover groups, available since OF 1.3, coupled with Bidirectional Forwarding Detection (BFD). Fast failover group tables act as a local agent, associating main and backup actions to a given match. BFD is a keep-alive protocol that periodically checks link availability and triggers the reconfiguration of the failover group table when a link is broken. These schemes have been validated in wired networks [13] and in a mm-wave wireless backhaul [14]. Our main contribution in this domain is a custom distributed recovery agent that avoids the overhead and detection delays introduced by BFD-like solutions, by instrumenting the MAC layer to detect link failures.

III. SODALITE System Model

In this section we introduce SODALITE's design. We focus our explanation on 5G, since it is the most complex case, but notice that our design is backwards compatible to 4G.

A. A primer on 5G Architecture

5G introduces architectural changes both in the data and control planes with respect to 4G. In the data plane, a 5G base station is called gNB, and is composed of one Centralized Unit (CU) and one or more Distributed Units (DUs). The functionality split between CU and DU is set below the Packet Data Convergence Protocol (PDCP) layer. Thus, IP flows are transmitted from the core network to the CU, where the PDCP is hosted, and the CU decides the most appropriate DU to transmit each packet. The F1 interface is defined between CU and DU, and is implemented using per-UE GTP tunnels. Like in 4G, there is also an interface between CUs to facilitate handover, interference coordination or dual-connectivity. This interface is called Xn and is also based on GTP. Connecting the gNB and the core network, there is a single type of data plane element, namely the User Plane Function (UPF), which can be instantiated at multiple locations. The UPF controls the flow of packets, i.e. *PDU session*, between the gNB and the core network. The interface between the gNB and the UPF is called N3 and is again based on per-UE GTP tunnels. The left part of Fig. 2 depicts an example 5G data plane. It is worth noting that all interfaces between data-plane elements are based on GTP, and one GTP tunnel is maintained for each unidirectional PDU session instantiated by a UE. Finally, we envision two possible scenarios to deploy SCs in 5G: i) the physical SCs embedding both CU and DU, thus interfacing with the core network via N3, or ii) having the CU instantiated in some edge computing facility, e.g. on a macro-cell site, coordinating through the F1 interface a cluster of SCs with only DU functionalities.

The 5G control plane adopts a micro-service architecture, where control plane functions are software-based instances that offer services to each other. In this paper we refer only to the relevant functions of Access and Mobility Management Function (AMF) and the Session Management Function

(SMF). The AMF tracks the position of the UE, relocates data-plane tunnels upon handover, and supports authentication functions. The AMF communicates with the gNB through the N2 interface. The SMF is in charge of managing the flow of packets between the gNB and the core network for each PDU session, and interacts with the UPF through the N4 interface. These control plane functions are illustrated in the right part of Fig. 2. The interested reader is referred to [15] for a detailed description of the 5G control plane.

Another aspect of 5G, relevant to the analysis of signaling conducted in Section V-B, is the new Radio Resource Control (RRC) Inactive State *Inactive* state, added to 4G's *RRC Idle* and *Connected* states. In 4G, a UE can be in RRC Connected, where end to end bearers are established between the UE and the core network, or in RRC Idle, with no established bearers and with the core network being able to locate the UE only in terms of tracking areas. Whenever a new session starts, the UE transitions from RRC Idle to RRC Connected, which entails a fair amount of signaling. In order to reduce this signaling, 5G has introduced the RRC Inactive state, which behaves like RRC Idle to the eyes of the UE, and like RRC Connected for the core network, i.e. N3 GTP tunnels are established. Thus, in RRC Inactive, the RAN -more specifically, the CU of an anchor gNB- tracks the location of the UE in terms of RAN-based Notification Areas (RNAs). A UE in RRC Connected can transition either to RRC Idle or to RRC Inactive, but once in RRC Idle it can only transition to RRC Connected [16]. When the UE transitions to RRC Connected from RRC Inactive, the CU in the serving gNB may retrieve the UE context from the CU in the last serving gNB and, if needed, request a N3 path switch to the AMF. If an RRC Inactive UE moves only between DUs controlled by the same CU, then no N3 path update is necessary, but a new F1 session is established between the new DU and the CU.

B. SODALITE system model

Fig. 2 illustrates the SODALITE model, where 5G SCs are connected to SODALITE nodes that provide wireless transport connectivity. SODALITE can be used to wirelessly backhaul the F1 interface, connecting DUs to a centralized CU, or the N3 interface, connecting SCs implementing DU and CU to UPFs. SODALITE nodes are grouped in control plane areas, and there is a logically centralized SODALITE controller in charge of performing traffic engineering for each area. In particular, the controller programs SODALITE nodes to allocate backhaul paths between SCs and *gateways*, where gateways provide connectivity between a control plane area and the rest of the transport network. Control plane areas in SODALITE are designed based on the geography of a given SC deployment. In practice, a SODALITE area is expected to have a reduced number of hops (e.g. < 5), and to incorporate more than one gateway . Physically, the SODALITE controller is a software function that can run in the core but, ideally, it would be placed in an edge location, closer to the wireless nodes. Note that the SODALITE control areas are transparent to the RAN network, i.e. gNBs connected to SODALITE nodes in different control plane areas can still establish Xn interfaces for handover and interference coordination. The

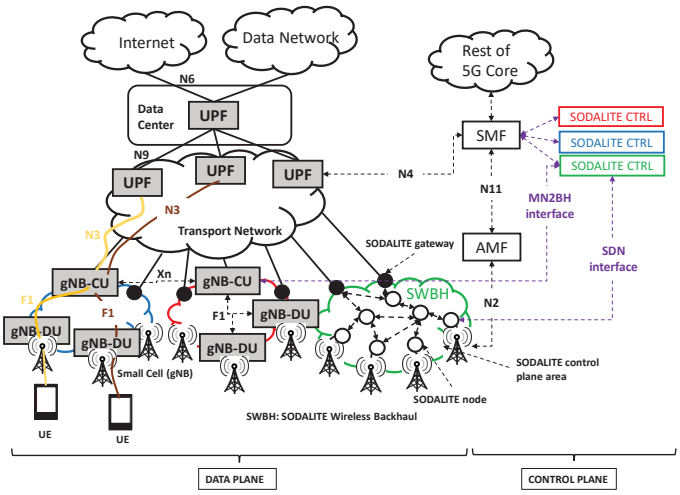


Fig. 2. SODALITE system model.

interested reader is referred to [17] for a detailed discussion of an SDN hierarchical and multi-area control plane that can be applied to a large-scale SODALITE deployment.

Central to the SODALITE architecture is the definition of a backhaul *flow*. In SODALITE, the controller defines flows using the Tunnel End Point ID (TEID) included in the GTP header. As seen in the previous section, GTP is the common tunneling protocol used between 5G data plane entities, and one GTP tunnel is maintained for each PDU session; thus, SODALITE also supports 4G networks, where GTP is used to establish S1 bearers between SCs and the core. Note that this differs completely from the behavior of current backhaul networks, where all the traffic from a given base station is aggregated in a single backhaul path. The main benefit provided by SODALITE is thus that an operator is now able to implement in the wireless backhaul, per-subscriber, looking at the GTP TEID, or per-application policies, again by correlating the output of a DPI[1] engine in the core network with the GTP TEID. Examples of such policies include forwarding traffic from prioritized subscribers (or applications) through low delay paths, or allocating flows to the proper wireless transport technologies (e.g., mmWave or Sub-6) according to different delay/bandwidth requirements. SDN is key to dynamically enforce policies in the wireless backhaul and, together with the fine-grained policing enabled by SODALITE, it enables slicing in 5G transport networks.

Fig. 2 depicts two novel interfaces defined in SODALITE: an *SDN interface* between the SODALITE controller and SODALITE nodes, and an interface between the Mobile Network and the SODALITE controller (*MN2BH interface*).

C. SDN Interface

The interface between the SODALITE controller and the SDN agent in the SODALITE nodes is based on Openflow (OF) [18], with the custom extensions described next. First, the port statistics reported by OF are extended with parameters specific to the wireless technology being used by the SODALITE nodes. Without loss of generality, we assume

[1]Deep Packet Inspection (DPI) can be used to identify the applications carried within a PDU Session/LTE bearer.

TABLE I
OVERVIEW OF THE OF STATISTICS REPORTED BY SODALITE NODES

Statistic	Explanation
Wireless Port Statistics:	
Channel Number	Operational channel of a wireless interface
Channel Usage	Occupancy of a wireless channel ($0 \leq u \leq 1$)
Signal Strength	Peer RSSI value measured by the wireless port
Tx/Rx Bitrate	Physical data rate used over a certain link
Tx/Rx Bytes	Bytes transmitted/received through this port
Tx/Rx Packets	Packets transmitted/received through this port
Flow Statistics:	
Tx/Rx Bytes	Bytes transmitted/received via a flow
Tx/Rx Packets	Packets transmitted/received by a flow

hereafter that SODALITE nodes and gateways embed multiple IEEE 802.11 interfaces, and implement the SDN architecture presented in [19]. In this way, each potential destination reachable through a wireless interface is exposed through a different virtual interface connected to the OF agent. In this architecture, wireless-specific parameters reported by the OF port statistics are defined in Table I.

In order to provide per-subscriber/application awareness, OF is extended to support an additional match type based on the GTP TEID. Thus, a SODALITE controller can define unidirectional flows in the wireless backhaul identifying the corresponding sessions in the Xn, F1 and N3 interfaces for 5G, or in the S1 interface in a 4G network.

D. Mobile Network to Backhaul Interface (MN2BH)

In order to optimize resources in the various network elements (DU, CU, UPF), the *UE context* and the corresponding data connections in the Xn, F1 and N3 interfaces are maintained when the UE is in RRC Connected state, and partially released when the UE is in RRC Idle or RRC Inactive. The fundamental idea behind SODALITE is to instantiate/release per-UE backhaul flows in the wireless transport following instantiation/release of (per-UE) F1 or N3 connections. Consequently, an interface is required between the SODALITE controller and the 5G/4G Mobile Network, which is aware of the RRC state of each UE. We refer to this interface as the *Mobile Network to Backhaul (MN2BH)* interface.

In 5G, when the UE transitions from RRC Idle to RRC Connected, the SMF is the entity in charge of establishing the corresponding N3 connection. Thus, the SODALITE controller interfaces with the SMF. However, if the UE transitions from RRC Inactive to RRC Connected, the RAN (CU in serving gNB) is in charge of reestablishing the F1 interface. Hence, in an implementation where CU and DU are separated and SODALITE is used as wireless transport for the F1 interface, the SODALITE controller needs to interface with the CUs of a given area. In 4G, the Mobility Management Entity (MME) is in charge of establishing and releasing the S1 bearer when the UE transitions between RRC Connected and RRC Idle. Therefore, in 4G, the SODALITE controller would interface with the MME. A complete definition of all these interfaces is out of the scope of this paper. However, as a matter of example, we describe next the MN2BH interface between a SODALITE controller and the SMF for the case where SCs implement collocated CU and DU functions and SODALITE is used as wireless transport for the N3 interface. Similar interfaces can be easily derived from the provided example.

Fig. 3 provides an example description of some of the procedures executed over the *MN2BH* interface, which is divided in three phases. First, the UE transitions from RRC Idle to RRC Connected upon the arrival of an uplink packet. The gNB notifies the AMF that the UE is in RRC Connected, and then the AMF initiates an authentication procedure with the UE. Once the UE is authenticated, the AMF starts a PDU session modification with the appropriate SMF. The SMF communicates the TEID used for the uplink part of the N3 interface (*N3-UPF-TEID*), and the involved gNB to the SODALITE controller. With this information, the SODALITE controller runs its path allocation engine and programs the newly computed path in the affected SODALITE nodes (c.f. Fig. 2). Then, the controller notifies the SMF that the transport path for the uplink N3 interface is up, after which the SMF responds to the PDU session request from the AMF, and the AMF notifies the gNB about the uplink GTP TEID.

After the uplink path is established (phase 2 in Fig. 3), the gNB assigns a TEID for the downlink path (*N3-gNB-TEID*) and notifies the AMF. Consequently, the AMF initiates a PDU session modification with the SMF. The SMF notifies the SODALITE controller, which proceeds to instantiate a downlink path. The UPF is assumed to have connectivity to all the gateways through the transport network. Once the downlink transport path is available, the SMF notifies the UPF and traffic begins to flow in the downlink direction. To minimize the call setup delay, a SODALITE controller could proactively pre-compute paths between gNBs and gateways in its control area, as explained later in Section IV-A.

In the third phase of Fig. 3, the inactivity timeout in the gNB triggers a transition to RRC Idle, for which the gNB requests the AMF to release the UE context, thus saving resources. The gNB then releases the radio resources and the UE transitions to RRC Idle. Following, the AMF requests the SMF to end the PDU session. The SMF notifies the SODALITE controller, and the downlink (*N3-gNB-TEID*) and uplink (*N3-UPF-TEID*) backhaul flows associated to that UE are released.

In scenarios using SODALITE as wireless transport for the F1 interface, a similar procedure would be established between the gNB CU and the SODALITE controller when the UE transitions between RRC Connected and RRC Inactive/Idle.

E. Impact of Transport Security on SODALITE

In 4G, and presumably also in 5G, IPsec can be used to encrypt communications between the gNB (SCs in our case), and an IPsec gateway in the core network. With IPsec, the GTP header is encrypted and, hence, SODALITE nodes would have no access to the GTP TEID, as required in the proposed architecture. However, SODALITE nodes support encryption at layer 2, rendering IPsec unnecessary within the SODALITE control area. IPsec may still be used between the area gateways and the core network if the remaining portion of the network is deemed insecure. We posit that operating SODALITE very close to the edge of the network, and given the performance

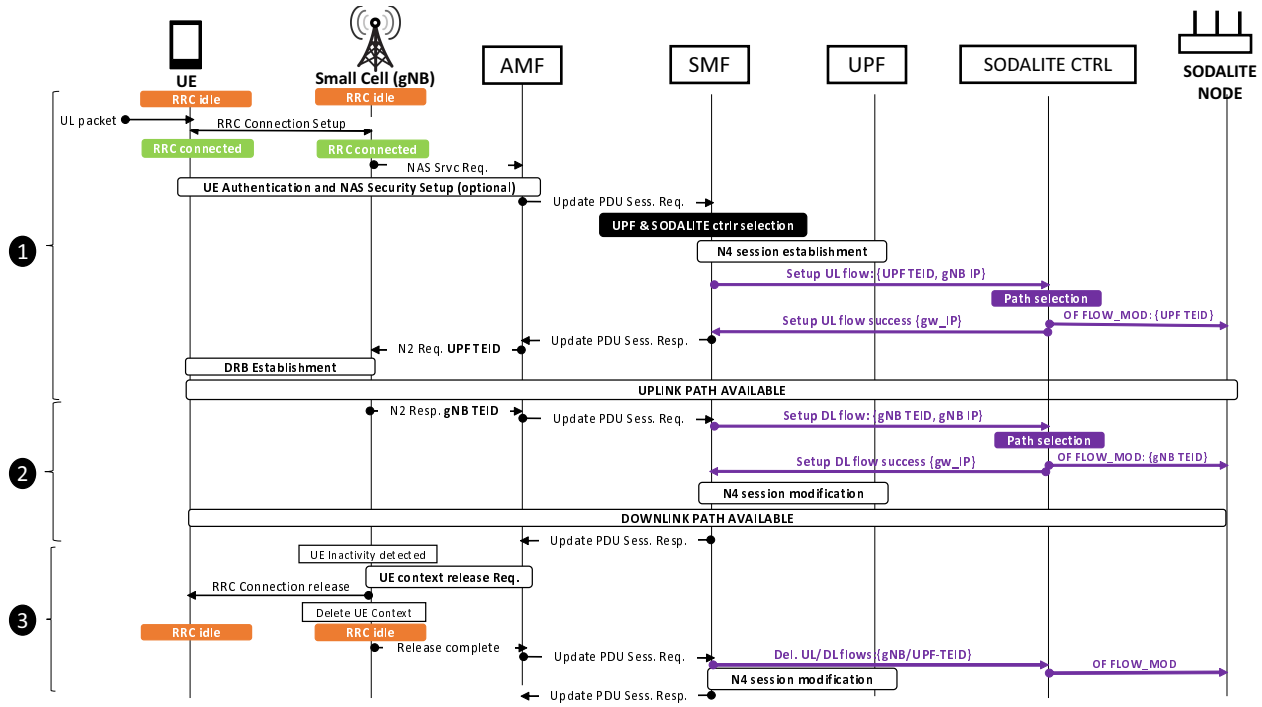


Fig. 3. Exemplary operation of the MN2BH interface when a UE moves from RRC Idle to RRC Connected and viceversa. Highlighted in purple are the message flows between the SODALITE controller and the SMF.

provided by current Network Processing Units (NPUs), per-node encryption is feasible.

IV. SODALITE Hybrid Forwarding Policies

To deliver a carrier-grade service, SODALITE's SDN architecture is composed of two sub-systems. First, a centralized proactive computation of per-flow main and backup paths. Second, a fast recovery agent in the SODALITE nodes that quickly reconfigures forwarding upon detecting a link failure. Next, we describe these sub-systems and the mechanisms used to identify link failures and trigger the fast recovery agent.

A. Centralized Computation of Main and Backup Paths

The left part of Fig. 4 illustrates the physical architecture of a SODALITE control area, where we can see SODALITE nodes having one or more physical radio interfaces (*phy. ifc*). Each physical radio interface of a SODALITE node is point to multi-point (P2MP) in nature, and operates on a predefined channel. Over each physical radio interface, a set of *virtual* point to point (P2P) interfaces are instantiated representing each peer node physically reachable through the interface (denoted as *virt. ifc* in Fig. 4). The wireless interface virtualization approach used in SODALITE is described in detail in [19], and ensures that as long as there is physical connectivity between two nodes, a corresponding virtual interface will be attached to the local SDN agent. The local SDN agent is in charge of packet forwarding in the data plane, and is programmed by the SODALITE area controller. The middle part of Fig. 4 describes the logical network model maintained by the controller, which corresponds to the physical network on the left.

Automatic network topology discovery is enabled using traditional OF mechanisms. Consequently, the controller maintains a set $\boldsymbol{N} = \{n_1, ..., n_N\}$ of nodes in the network, and

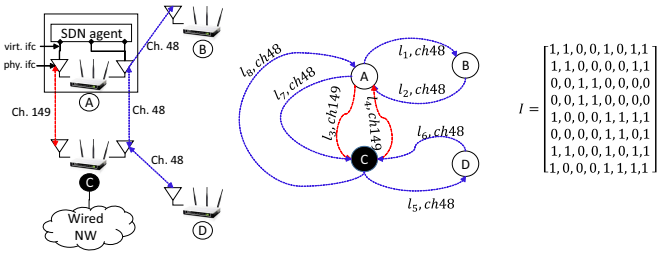


Fig. 4. Physical network with SDN architecture showing detail for one SODALITE node (left). Network model in the SODALITE controller (middle). Interference matrix (right)

a set $\boldsymbol{L} = \{l_1, ..., l_M\}$ of unidirectional links between nodes. Table II depicts the state that the controller maintains for each link, which is periodically updated through OF port statistics (c.f. Section III-C). Based on the discovered topology, the SODALITE controller derives an interference matrix, $\boldsymbol{I} = \{i_{q,m} \in \{0,1\} \mid 1 \leq q,m \leq M\}$ depicted on the right hand side of Fig. 4 , capturing potential interference between backhaul links. For the case of sub-6 GHz radios, which is the main case studied in this paper, $i_{q,m} = 1$ if the links share the same channel, i.e. $ch_q = ch_m$, and the origin nodes of the two links are the same, or are one hop away in the network.

The main goal of the SODALITE controller is to compute, for each transport flow, a *main* and a *backup* path connecting the gNB (or eNB) where the UE is associated, to a gateway in the SODALITE control area (recall that flows are defined as unidirectional). Transport flows are computed whenever a UE transitions to RRC Connected (c.f. Section III-D), and they are updated when congestion is detected in the network (more details provided later). For each flow, the SODALITE controller uses the metric $\boldsymbol{\mu_{P_x}}$ to evaluate the potential effect

TABLE II
STATE KEPT BY THE SODALITE CONTROLLER FOR EACH LINK IN L

Parameter	Description
r	EWMA of physical data rate (bps)
ρ	EWMA of carried traffic (bps)
MTU	MTU size of the radio interface
ch	Configured channel in the radio interface
n_o	Origin (transmitting) node for this link
n_r	Remote (receiving) node for this link
h_o	Origin (transmitting) physical interface for this link
h_r	Remote (receiving) physical interface for this link
u	EWMA of utilization level sensed by h_o $(0 \leq u \leq 1)$
$\boldsymbol{F}$	Set of flows currently being forwarded through this link

of selecting a main path P_x on the overall state of the wireless backhaul, and the metric $\delta(P_x, P_y)$ that measures the similarity between paths P_x, P_y, in order to select the best backup paths.

The vector $\boldsymbol{\mu} = \{u_1, ..., u_M \mid 0 \leq u_i \leq 1\}$ captures the network state in terms of link utilization, where u_i is the utilization of $l_i \in \boldsymbol{L}$ (c.f. Table II). Note that $\boldsymbol{\mu}$ is periodically updated by the OF statistics received from SODALITE nodes. The vector μ is an internal representation of the network state in the SODALITE controller that allows the controller to analyze how different path allocations would impact the network state. Adopting this approach allows SODALITE to design path allocation algorithms that are reactive in nature, i.e. they continually measure the network state and update the computed paths accordingly. To understand how a potential path allocation for a new flow affects $\boldsymbol{\mu}$, we first define a path as a vector $\boldsymbol{P} = \{p_1, ..., p_M \mid p_i \in \{0, 1\}\}$, where $p_i = 1$ if l_i belongs to the path. Considering that a new flow f with traffic load of ρ_f bps needs to be allocated, it is possible to compute the level of utilization introduced by this flow when traversing link l_i as $\Delta f(l_i) = \frac{\rho_f}{\mathrm{MTU}_i}(\frac{\mathrm{MTU}_i + P_{\mathrm{OH}}}{r_i} + A_{\mathrm{OH}})$. P_{OH} and A_{OH} are respectively the per-packet header overhead, measured in bits, and the channel access related overhead, measured in seconds, and r_i is the average physical data rate used in l_i; information already available at the controller (c.f. Table II). Consequently, the SODALITE controller estimates the link utilization in the wireless backhaul if f is allocated through path $\boldsymbol{P_x}$ as

$$\boldsymbol{\mu_{P_x}} = \boldsymbol{\mu} + \boldsymbol{I} \times \left(\boldsymbol{P_x} \circ \Delta_f(\boldsymbol{L})\right), \tag{1}$$

where $\Delta_f(\boldsymbol{L}) = \{\Delta_f(l_1), \ldots, \Delta_f(l_M)\}$ and $\circ$ is the Hadamard product (i.e., element-wise product) for two vectors. Hence, the SODALITE controller can use the metric $\boldsymbol{\mu_{P_x}}$ to compare the impact of allocating different main paths to a given flow.

Having defined a metric to compare the impact of main paths, we now define the metric used by the controller to evaluate different potential backup paths $\boldsymbol{P_y}$ given a main path $\boldsymbol{P_x}$. Intuitively, in order to increase reliability we should choose backup paths to be as disjoint as possible from the main path. Note that backup paths in SODALITE are proactively set up to allow for a quick reaction to failure, but the use of a backup path is only transient, because after the controller discovers through regular topology updates that a link has failed, an overall path re-allocation can be performed as we will discuss later. We denote $\delta(\boldsymbol{P_x}, \boldsymbol{P_y})$ to be the measure of similarity between paths $\boldsymbol{P_x}$ and $\boldsymbol{P_y}$ and define it as

$$\delta(\boldsymbol{P_x}, \boldsymbol{P_y}) = \lambda \frac{|h_o(\boldsymbol{P_x}) \cap h_o(\boldsymbol{P_y})|}{|h_o(\boldsymbol{P_x})|} + (1 - \lambda) \frac{|n_o(\boldsymbol{P_x}) \cap n_o(\boldsymbol{P_y})|}{|n_o(\boldsymbol{P_x})|}, \tag{2}$$

where $h_o(\boldsymbol{P_j})$ and $n_o(\boldsymbol{P_j})$ are, respectively, the set of transmitting physical radio interfaces and the set of transmitting nodes of the links traversed by $\boldsymbol{P_j}$ (c.f. Table II). Note that, in general, $0 \leq \delta(\boldsymbol{P_x}, \boldsymbol{P_y}) \leq 1$, and $\delta(\boldsymbol{P_x}, \boldsymbol{P_y}) = 1$ when P_y contains all nodes and interfaces found in P_x. Conversely, if the two paths use completely disjoint sets of radio interfaces and nodes, $\delta(\boldsymbol{P_x}, \boldsymbol{P_y}) = 0$. The constant λ allows to trade off the importance of being disjoint in terms of radio interfaces or nodes, which is related to the failure probability of each component. Based on those two metrics, we define two different policies for path allocation.

Sequential Policy: in this policy, the main and backup paths are allocated sequentially. Given a set of K possible paths between two nodes, the controller first allocates the main path as the path that minimizes the maximum link utilization in the network. Then, the controller searches the remaining set of $K - 1$ paths and allocates as backup path the path that minimizes $\delta(\boldsymbol{P_x}, \boldsymbol{P_y})$. Formally, the two paths are found as

$$\boldsymbol{P_{main}} = \underset{\boldsymbol{P_x} \in K}{\arg\min} \quad \max(\boldsymbol{\mu_{P_x}}), \tag{3a}$$

$$\boldsymbol{P_{backup}} = \underset{\boldsymbol{P_y} \in K \setminus \{\boldsymbol{P_{main}}\}}{\arg\min} \quad \delta \boldsymbol{P_{main}}, \boldsymbol{P_y}). \tag{3b}$$

The sequential policy minimizes the network utilization incurred by the main path, at the cost of selecting sub-optimal backup paths. To have more flexibility in addressing the trade-off between the performance of the main path, and the reliability of the backup path we introduce the joint policy.

Joint Policy: A joint allocation of the main and backup paths is considered through a multi-objective optimization problem controlled by a weight parameter $0 \leq \gamma \leq 1$ as follows

$$(\boldsymbol{P_{main}}, \boldsymbol{P_{backup}}) = \underset{(x,y)}{\arg\min} \; (\gamma \max(\boldsymbol{\mu_{P_x}}) + (1 - \gamma)\delta(\boldsymbol{P_x}, \boldsymbol{P_y})), \tag{4}$$

where $\boldsymbol{P_x} \in K$ and $\boldsymbol{P_y} \in K \setminus \{\boldsymbol{P_x}\}$.

In order to find the optimal allocation for the sequential and the joint policies, the set of acyclic paths between the source and destination nodes should be considered. However, the number of simple paths between two nodes grows exponentially with the size of the network, which renders this approach impractical. For example, the number of simple paths between two vertices for a complete graph of n nodes is $K = (n-2)! + \sum_{j=1}^{n-2} \binom{n-2}{j} \frac{j!}{n-j-1}$. In addition, once the set of K paths is computed, the sequential policy has a complexity of $O(2K)$ and the joint policy of $O(K^2)$.

To speed up the computation of the candidate K paths we propose the following heuristic. Given a source-destination tuple, the SODALITE controller computes a set of K-shortest paths using the WCETT metric [20]. This metric weighs two factors, the sum of the Expected Transmission Time (ETT) of the links across the path, and a factor that penalizes using multiple links in the same channel. Hence, the WCETT metric results in a reduced set of paths that are good in terms of

minimizing congestion, i.e. $\max(\boldsymbol{\mu}_{\boldsymbol{P_x}})$. Alternative heuristics to pre-select the candidate K paths could be considered if per-application policies are in place, for example allowing only shorter paths in K if delay constrained applications are considered. In Section VI, we evaluate the effect of this heuristic on the two defined policies.

Finally, we provide a few remarks on the situations that trigger a new path allocation in the SODALITE controller. The first situation is when the controller is notified by the mobile network that a new PDU session with a given GTP TEID is being provisioned (c.f. Section III-D), which triggers the configuration of transport flows for the uplink (UL) and downlink (DL) directions. Given an initial flow setup, the controller needs to make an assumption on the traffic load of the flow to be allocated ρ_f, for which SODALITE uses the average of the other flows currently being transmitted through the network. Note, though, that the load injected by each flow into the network varies over time, which can result in some link being congested, i.e. the utilization level of a link exceeds a pre-configured threshold $u_i > u_{THR}$. The controller detects these congestion events through periodic OF port statistics. Upon detecting a congestion event , the controller applies a flow reallocation policy to mitigate the congestion event. The heuristic implemented in SODALITE is described as follows.

On a congested link l_i, the controller finds the set of flows $\boldsymbol{F} = \{f_1, ..., f_F\}$ traversing link l_i (c.f. Table II), sorts them in decreasing order of load (i.e. according to ρ_{f_j}), and loops through $\boldsymbol{F}$ trying to sequentially reallocate each $f_j \in \boldsymbol{F}$ until an allocation is found where the congestion situation is removed, i.e. $\max(\boldsymbol{\mu}) < u_{THR}$. To reallocate a flow, including main and backup paths, the controller first removes from $\boldsymbol{\mu}$ the effect of the current path $\boldsymbol{P_x}$ traversed by f_j, i.e. $\boldsymbol{\mu_0} = \boldsymbol{\mu} - \boldsymbol{I} \times \left(\boldsymbol{P_x} \circ \Delta_{f_j}(\boldsymbol{L})\right)$, and then re-runs the procedure described above according to the policy being used. The interested reader is referred to [12] that describes the conditions that may trigger a new backhaul path computation, albeit for a simplified policy that does not support allocation of backup paths.

B. Distributed Fast Path Recovery via Fast Local Link Reroute (FLRR) Agent

In addition to the SDN agent in charge of packet forwarding, SODALITE nodes embed a second agent in charge of fast path recovery in case of link failure. This agent is henceforth referred to as the Fast Local Link Reroute (FLRR) agent.

After a main and backup path have been determined, the controller assigns one of these five logical roles to the nodes along the paths: *common* nodes, *switch* nodes, *intermediate* nodes, *next-to-merge* (NTM) nodes, and *destination* nodes. The roles are assigned as follows:

1) A node where the main and backup paths branch-off is defined to be a switch node.
2) A node which is the origin of a link common to the main and backup paths is identified as common node.
3) Nodes originating only a main path link that connect to a common or switch node, i.e. where the main and backup path merge, are defined to be NTM nodes.

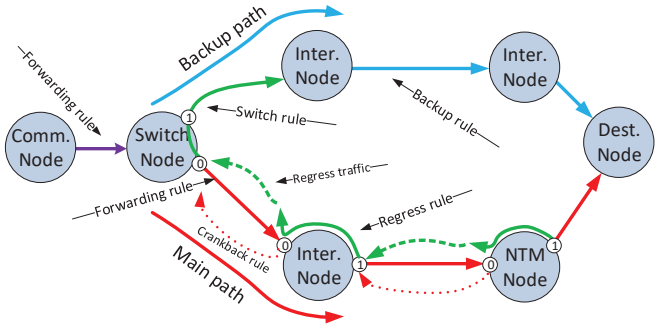


Fig. 5. Example topology with roles assigned by the controller depending on their position on main and backup paths, along with the different FLRR rules.

4) Nodes that originate links exclusively for the main or the backup path, but are not NTM nodes, are identified as intermediate nodes.
5) The node were the main/backup paths terminate is identified as destination node.

Once the roles have been assigned to the nodes on the main and backup paths, the controller proceeds with the installation of FLRR-specific rules in the SDN agent of each node involved in both paths. There exist four different rules in FLRR: i) *forwarding* rules, present in all FLRR nodes, ii) *regress* rules, present only in intermediate nodes of the main path, iii) *crankback* rules, pushed locally by the FLRR agent on nodes that either detect a link break or an active regress rule, as explained later, and iv) *switch* rules, present only in switch nodes. Each of these rules performs matching on two fields: the GTP TEID and the input port wherein the data packet entered the SDN agent. While the GTP TEID is the key identifier for different end-to-end data flows, the input port is necessary for FLRR to take autonomous recovery decisions.

Forwarding rules are used for the main and backup paths, pointing towards a path's destination, i.e., the gateway or entry nodes for up- and down-streams, respectively. Forwarding rules match on the input port through which packets enter the switch from the previous hop and they output the packets via the port that connects with the next hop in the path. Note that at a destination node a *forwarding* rule hands the traffic over to a local output port, e.g. the physical interface that connects with the SC or the wired network. *Crankback* rules are of temporary nature: once the FLRR agent of a node on the main path detects a link break on its originating link, it replaces the forwarding rule with a *crankback* rule. It defines the *crankback* rule by updating the output port of a *forwarding* rule to be the same as the input port. This reverts the traffic direction and makes packets return to the previous hop.

On the way back, either a *regress* rule or a *switch* rule intercepts the regressing traffic, depending on whether the previous hop is an intermediate or a switch node, respectively. A *regress* rule forwards regressing traffic (i.e. traffic that traverses an intermediate node in opposite direction) back towards the preceding node on the path. Once a *regress* rule detects matching packets, the FLRR agent replaces the existing *forwarding* rule for the same GTP TEID with a *crankback* rule, which prevents forwarding new packets in the direction of the broken link. When regressing traffic hits a node with a

switch rule, the traffic is redirected towards the backup path to reestablish communication. As soon as packets start matching the *switch* rule on a node, the FLRR agent modifies the output port of the original *forwarding* rule with the same GTP TEID to point towards the backup path.

Fig. 5 shows an example topology that highlights the roles along with the corresponding FLRR rules. Continuous lines show proactively installed rules. Dashed lines indicate how traffic would flow if a link break is detected on the main path, whereas the dotted lines indicate *crankback* rules that replace the original forwarding rules. Note that an arbitrary topology could include further common, intermediate, switch or NTM nodes. In our implementation, the FLRR agent is a user space program that monitors and configures the SDN agent.

C. Link Break Detection

The dynamics of link failures in wireless networks are more complex than in wired networks. For example, a link could temporarily fail due to congestion, an obstacle (mmWave links), or an energy conserving module in the SDN controller could decide to switch off a SODALITE node.

Traditionally, detection of broken links between adjacent nodes use some sort of keep-alive protocol, whereby a link failure is assumed after a given number of consecutive control messages are lost. Thus, there is an inherent trade-off between detection delay and signaling overhead. Maximizing the bandwidth available to user plane transmissions is especially critical in wireless backhaul networks, hence, SODALITE proposes a cross-layer scheme to detect link breaks, where existing MAC signaling is re-used for this purpose. The detailed description that follows is based on IEEE 802.11 radios, but similar principles could be applied to other radio technologies.

IEEE 802.11 radios periodically generate beacon frames to facilitate tasks such as node discovery. SODALITE nodes interpret the transmission of beacon frames from peer nodes as keep-alive messages for the purpose of maintaining link reliability information. If a certain number of consecutive beacons are lost, or if they are received with RSSI below a configured threshold, the radio notifies the FLRR agent that the corresponding link is broken. Notice that the link break detection time can be decreased by reducing the Beacon Interval (*BI*), which, however, increases signaling overhead. To resolve this trade-off, SODALITE also exploits another signal from the MAC layer to trigger a link break detection. If a packet transmitted from a node to its peer exceeds a configured number of retransmissions, that link is considered broken and the FLRR agent is notified. The threshold on the retransmission counter could be provisioned by the SODALITE controller to the FLRR agent.

The proposed hybrid scheme addresses the trade-off between detection time and signaling overhead in the following way. When there is no traffic flowing between two nodes, and therefore quick detection is not critical, a link break is detected through a consecutive number of missed beacons, which may result in increased detection times if a large *BI* is used. On the other hand, when traffic is flowing between two nodes and link break detection time is critical to prevent packet loss, an excessive number of retransmissions is the signal used to quickly notify the FLRR agent about a link break.

V. Scalability Analysis

SODALITE introduces new signaling to set up and tear down backhaul flows, and requires its nodes to maintain per-session state. Therefore, in this section we study the scalability properties of our architecture, based on network measurements gathered from an operational LTE network consisting of 10 eNBs and 33 cells, during a period of 30 days. The possible effect of cell densification or system architecture in 5G on the scalability results are discussed in each subsection.

A. Dimensioning Flow Tables in the SODALITE Nodes

In the collected traces, an *Active UE* corresponds to a UE having one Signaling Radio Bearer (SRB) and at least one non-guaranteed bitrate Data Radio Bearer (DRB) successfully configured. In our measurement campaign UEs have at most one Best Effort DRB, which aggregates all Internet traffic. Hence, we can use the number of active UEs in a cell to estimate the number of SODALITE backhaul flows originating from or ending at that cell. In practical deployments though, UEs could also have a second DRB for VoLTE services, which could be modelled adding a 2x factor to the results presented in this section. To calculate the number of rules at a SODALITE node, we consider the worst case of 3 rules per flow, which corresponds to the *switch* nodes, as well as *intermediate* nodes with 2 rules on the main path and 1 rule on the backup path. Considering one UL and one DL flow for each active UE, for n UEs being backhauled, $6n$ rules need to be maintained at any SODALITE node traversed.

To dimension the number of rules maintained at peak traffic hours, we first find the busy hour from our LTE traces, to then generate an empirical CDF of the number of active UEs per cell during the busy hour. We then derive numerically the CDF of the number of rules that a SODALITE node would carry for a given number of cells as follows. First, we generate randomly the number of flows for each cell (by using the CDF of number of active UE), then sum these numbers and multiply it by 6 to find the number of rules for these random samples. After producing $100,000$ tuples of random number of flows for all cells, we generate the corresponding CDF for the number of rules at a SODALITE node, which is shown in Fig. 6.

Dense urban 5G deployments are expected to aggregate no more than 10 SCs through a wireless backhaul [1]; in Fig. 6, we show an evaluation with up to 20 aggregated cells. The CDF generated from the traces for the number of rules per cell aligns with the 1 cell case, which validates the numerical method used for the evaluations. For all the considered cases, the SODALITE nodes are hence expected to have less than $2K$ rules. This number is lower than the hardware capacity of commodity switches, which typically rely on Ternary Content Addressable Memory (TCAM) to implement the flow tables, allowing $2K$ to $24K$ rules [21], and can also be easily accommodated by software switches. These results show that SODALITE can be implemented on current LTE systems even dropping the assumption of one flow per UE, and should scale to 5G deployments, without any capacity issues.

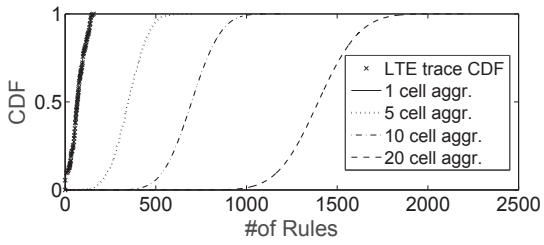


Fig. 6. CDF of number of rules at a transport node carrying flows of a given number of cell.

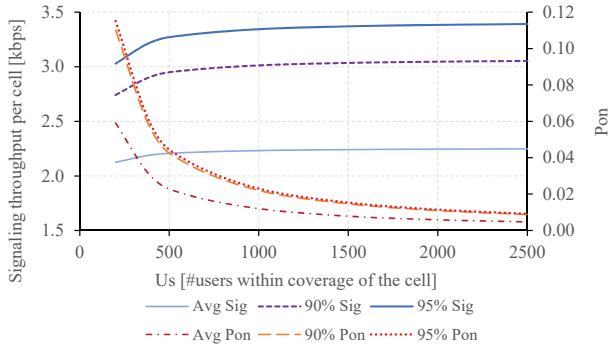


Fig. 7. Signaling per cell and P_{on} for different combinations of U_s, U_a and D_h

B. Control Plane Overhead in SODALITE

In order to assess the amount of signaling traffic necessary to keep an updated rule set in all the SODALITE nodes, we first model each user as an ON-OFF state machine. From the point of view of the backhaul control, a UE in ON state represents multiple entries in the forwarding table of a set of SODALITE nodes (cf. Section IV) while, for UEs in the OFF state, no record is kept. Each transition between the two states entails the generation of OF messages from the controller to those SODALITE nodes participating in the forwarding of the affected flows. Transitions from ON to OFF trigger the signaling that will remove any reference to the inactive UE from the flow tables, while OFF to ON transitions require the creation of new entries to handle the new UE's flows.

Therefore, knowing the transition rates between ON and OFF states for an average user, along with the total number of users present in a cell, we can provide a good estimation of the amount of signaling generated in the proposed architecture. According to this model, ON to OFF transitions for a UE occur at an average rate β (i.e. the average time a UE's flows are active in the backhaul is $1/\beta$). The average OFF to ON transition rate is α (i.e. the average time in OFF state is $1/\alpha$).

As in the previous section, we consider the worst case where an OFF→ON transition requires six new rules (size of OF packet measured at IP layer is $L_{on} \approx 620B$) to be forwarded to all involved nodes, while an ON→OFF transition requires only one message ($L_{off} \approx 150B$) to remove those rules. Therefore, the average amount of backhaul signaling received by each switch, for each cell aggregated on that switch's path is

$$\rho_{Sig} = U_s \, (P_{off} \, \alpha \, L_{on} + P_{on} \, \beta \, L_{off}), \tag{5}$$

where U_s represents the total number of ON-OFF machines on each cell (i.e. number of subscribers covered by a cell), P_{on} and P_{off} are the steady state probabilities of finding the user in ON and OFF states, respectively (see (6)).

Note that U_s is not the number of simultaneously active UEs per cell, here denoted as U_a. While U_a can be directly obtained by analyzing the network traces (on average, U_a =11.8 during the busy hour; 23.0 at CDF 95%), unfortunately, the real value of U_s is unknown. We will study values of U_s between 200 and 2500 to later prove that the real value of U_s has minimal impact on (5). Following the ON-OFF model, U_a (the average number of active UEs in a cell covering a population of U_s subscribers) is just $U_a = U_s P_{on}$, where

$$P_{on} = \alpha/(\alpha + \beta), \quad P_{off} = 1 - P_{on}. \tag{6}$$

Also note that, for LTE and assuming static users, ON and OFF states would directly map to RRC Connected and RRC Idle, respectively. In 5G, the correspondence to *connected*, *idle* and *inactive* states depends on the interface over which SODALITE works (F1 or N3). Although our analysis applies both to 4G and 5G, it is particularized to 4G given the nature of our traces and the fact that it represents an upper bound (note that 5G's *inactive* state reduces the impact of UE transitions).

With mobility, an active UE performing a handover (involving a change in the backhaul path) represents an ON to OFF transition for the transport nodes in the old path and an OFF to ON transition for the nodes in the new path, even though the UE remains in RRC Connected state during the whole process. Hence, in order to obtain a value for α that includes the effects of mobility, we measure the average rate at which new DRBs are established in a cell during a busy hour (D_h), since it includes both inactive users becoming active as well as active users moving to that cell. Then, $\alpha = D_h/U_s$. The only remaining unknown is β, which can easily be derived using (6). Then, $\beta = D_h(1/U_a - 1/U_s)$.

With the values of U_a and D_h obtained from the traces, and for a wide range of U_s (from 200 to 2500), the average active time at the link layer ($\frac{1}{\beta}$) varies only from 32 s to 45 s. Note that, at transport or network layers, data sessions can last for hundreds of seconds [22]. This divergence is due to two factors: i) whenever there is an inter-packet gap of 12 s or more [23], the RRC inactivity timer automatically moves the UE to RRC Idle (or RRC Inactive in 5G), and ii) handovers entail an ON to OFF transition in the old cell, even though the user's session is alive during the process.

Finally, following (5) we can compute the signaling traffic generated by the controller and received by each SODALITE node serving a particular cell. Fig. 7 shows the signaling throughput per cell during a busy hour for three different cases: i) Avg: U_a = 11.8, and D_h = 1320, ii) 90%: U_a = 22.8, and D_h = 1800, and iii) 95%: U_a = 23.0, and D_h = 2000.

Note that U_s has almost no impact on the resulting signaling traffic per cell. This is because, for a given U_a, having a larger population implies that those U_s subscribers are active less often (i.e. P_{on} decreases with U_s, as shown in Fig. 7).

With these numbers, even not considering multiplexing gains when multiple cells are aggregated, the signaling generated can be measured in tens of kbps. For example, a backhaul path consisting of four hops aggregating 10 to 15 cells would generate, about 85 kbps, on average, during the busy hour

(less than 180 kbps in 95% of the cases). Recall that this analysis assumes a worst case, where all SODALITE nodes require six rules per user and handovers require a complete reconfiguration of the path (i.e. no common nodes between old and new paths). These results clearly show that SODALITE's signaling is negligible compared to the 4G/5G user data rates.

VI. Performance Evaluation

In this section, we provide an experimental evaluation of SODALITE. First, we evaluate the trade-offs incurred by the path allocation policies introduced in Section IV-A. Then, we study the delay introduced by SODALITE when detecting a link failure and moving the traffic to the backup path. Finally, we demonstrate the behavior of SODALITE in a fully operational LTE network hosted in the NITOS testbed [24]. Given the similarities between 4G and 5G backhaul interfaces, we posit that the derived conclusions also apply to 5G.

A. Evaluating SODALITE Path Allocation Policies

In order to evaluate the sequential and joint policies introduced in Section IV-A, we consider network sizes between 6 and 14 nodes representing a SODALITE control area. For each size, we randomly generate 10 different *grid-like* topologies, based on a Small Cell urban deployment for a typical European city proposed in [25]. Random topologies are generated in the following way. First, a random number of nodes is deployed along a chain topology with an inter-site distance of 100 meters (representing a main street). Then, additional node chains, orthogonal to the main one (representing crossing streets), are randomly added to the topology until the intended size is reached. The SODALITE nodes in our evaluation are dual-radio, with each radio operating on a different channel, the coverage range is set to 90 meters. To derive the data-rate available in each link we use the TGn channel model E defined in [26], and the SNR to MCS mapping tables reported in [27] for a 40 MHz channel, which results in a range of per-link PHY data-rates between 13 Mbps and 260 Mbps. Two of the nodes are randomly chosen to act as gateways. Each node in the network is collocated with an eNB and exchanges UL and DL traffic with one of the two gateways.

A custom made simulator is used to evaluate and compare the performance of the sequential and joint policies in terms of the number of admitted flows, which measures the amount of resources taken by the main path, and the reliability of the backup path. In addition, we compare our policies to policies that choose their paths based on the WCETT and shortest path metrics.To evaluate the impact of each policy on network capacity, UE flows are randomly generated with a size between 1 Mbps and 5 Mbps, then a path allocation decision is taken according to the policy under study, until the maximum network utilization reaches a configured threshold ($u_{THR} = 0.9$). The resulting number of admitted flows is reported in the upper row of Fig. 8. To measure the reliability of the backup path, we sequentially remove each physical interface belonging to the main path, and increase a counter *cnt* if the backup path does not contain that interface. Reliability is then computed as $\frac{cnt}{N_{pif}}$, where N_{pif} is the number of physical interfaces in the main path. Reliability is depicted in the lower row of Fig. 8. Reliability for WCETT and shortest path is not reported since they do not support setting of backup paths.

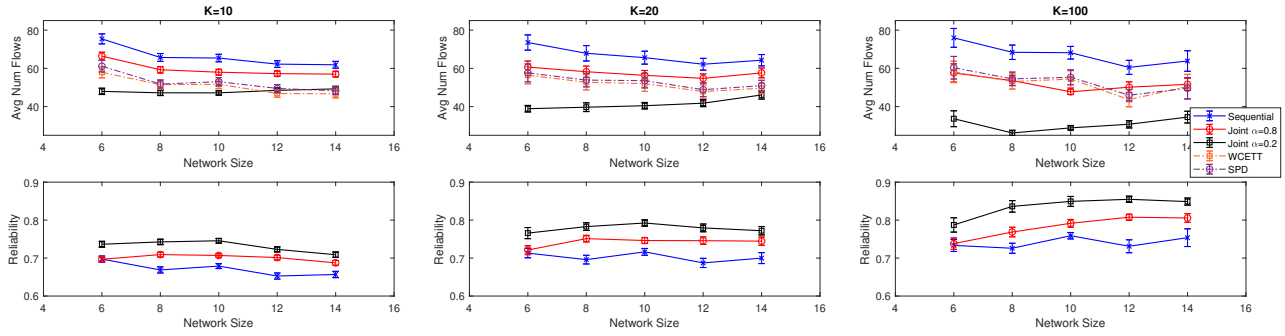
Fig. 8 presents the results of the sequential policy (blue line), the joint policy with $\gamma = 0.8$ (red), the joint policy with $\gamma = 0.2$ (black), WCETT (dashed pink line), and the shortest path policy (dashed cyan line), for the generated random topologies. In addition, $K = \{10, 20, 100\}$ are considered in the study, to evaluate the effect of the K-shortest path heuristic on the performance of the sequential and joint policies, which does not impact in the WCETT and shortest path policies that only consider one path for each flow. 95% confidence intervals are overlaid along the obtained mean values.

Looking at Fig. 8 we can see that, the sequential and the joint-0.8 policies significantly outperform the standard WCETT and shortest path policies in terms of admitted flows. This validates the network model used by the SODALITE controller, and the criteria of minimizing worst-case congestion when allocating flows. In addition, the effect of K on the number of admitted flows is relatively small (only visible for larger topologies), which validates the ability of the WCETT K-shortest path heuristic to generate a set of candidate paths with a reduced utilization. On the other hand, the joint-0.2 policy does not outperform WCETT and shortest path in terms of admitted flows, and even performs slightly worse when $K > 10$. The reason is that joint-0.2 prioritizes finding disjoint main and backup paths, even if these undergo a higher utilization; on the other hand, joint-0.2 improves reliability, as compared to the sequential and joint-0.8 policies.

The performance of the reliability index is more dependent on K, especially for the joint-0.2 policy. The reason is that, with larger K, longer and potentially more disjoint backup paths are included within the set of candidate paths. However, longer backup paths may lead to a reduced performance until the SDN controller realizes that the main path has been disrupted and re-configures the network. We consider that $K = 20$ is a good compromise between computational complexity in the SDN controller and the reliability performance trade-off obtained under the different policies.

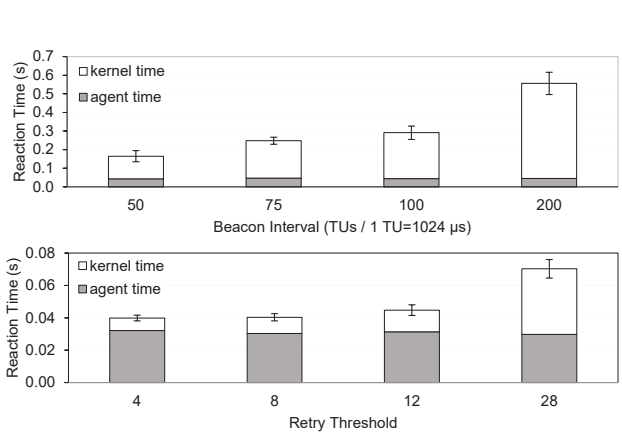
B. Benchmarking Link Failure Detection Time

We prototype SODALITE in Linux based devices, using a software switch as SDN agent, and a custom user-space program as FLRR agent. Then, we modify the mac80211 driver in Linux to implement the link break detection schemes described in Section IV-C. To measure the delay required by a SODALITE node to detect a link break, we configure two wireless devices in the NITOS testbed, hereafter referred to as node A and node B, configured in mesh mode. These nodes are equipped with Qualcomm 802.11n wireless NICs, our FLRR agents and custom *mac80211* driver to support wireless SDN [19]. Then, we proceed to shut down the radio interface in node A and measure in node B the time at which B detects the link break, t_{break}, and the last time B confirmed that the link with A was active, i.e. t_{lastOK}. We evaluate the two mechanisms for link break detection described in Section IV-C, namely, missed beacons detection, and detection of excessive number of retransmissions. Note that t_{lastOK} is

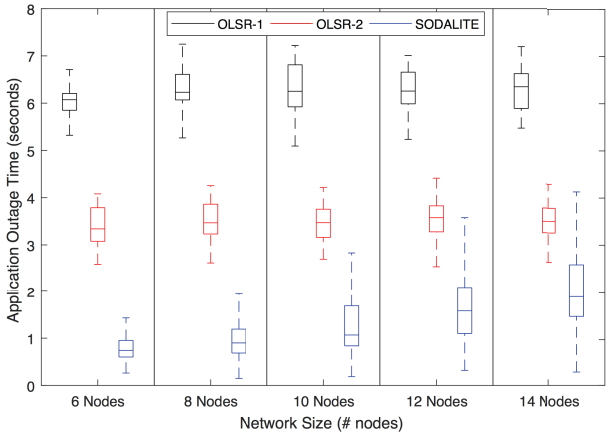


(a) $K = 10$ (b) $K = 20$ (c) $K = 100$

Fig. 8. Number of accepted flows and reliability for sequential and joint path allocation policies in randomly generated topologies for different values of K.



(a) Beacon (upper) and retry (lower) based detection with 95% confidence intervals.



(b) Application recovery time: median, first and third quartiles (lower/upper box limits) and minimum and maximum values (whiskers).

Fig. 9. Benchmarking SODALITE fast recovery.

computed differently in both schemes. With missed beacons, t_{lastOK} corresponds to the last time node B received a beacon from node A, whereas in the case of retransmissions, t_{lastOK} corresponds to the last time when node B received an acknowledgment from node A. The link break detection time is measured as $t_{detection} = t_{break} - t_{lastOK}$. Hence, the results reported in this section have to be understood as an upper bound for the true link break detection time. In the beacon-based detection experiments we fix the threshold of missed beacon frames at 2, and do not transmit any traffic between A and B during the experiments. In the retry-based detection experiment we set up an *iperf* UDP transfer saturating the wireless channel, and test different configurations regarding the retry threshold. In order to provide statistically meaningful results, each experiment is repeated 50 times and the average is reported in Fig. 9a.

The upper part of Fig. 9a depicts the results of the beacon based experiment for a BI varying between 50 ms and 200 ms. The depicted link break detection time is broken into two components: i) *kernel* time is the time required by the *mac80211* kernel module in node B to detect that 2 beacons are missing from node A, and ii) *agent* time is the time required by the FLRR agent to process the notification from the kernel module. As expected, we see in the upper part of Fig. 9a that *kernel* time scales, on average, as $2.5 \times BI$, where BI is the configured beacon interval. The reason is that, in our implementation, the kernel module in node B piggybacks on its own beacon transmissions to check whether the time since the last beacon received from a peer node has exceeded the configured threshold, which results in a detection time $t_{detection} = 2 \times BI + \phi$, where ϕ is the random phase between the beacon generation times of nodes A and B. The lower part of Fig. 9a depicts the same *kernel* and *agent* times for the retry-based experiment. The tested configurations are obtained by varying the retry limit to 4, 8, 12 and 28 retransmissions. After discarding a packet, the radio module generates a report to the kernel module, which then evaluates if the number of retransmissions has exceeded the configured threshold. Looking at *kernel* time in the lower part of Fig. 9a, we see that it only significantly exceeds 10 ms due to retransmissions and contention backoffs when the retry limit is configured at 28, due to the binary exponential backoff used by IEEE 802.11, in which case, *kernel* time is around 40 ms. Regarding *agent* time, it is always around 30 ms in our experiments, which is a result of our current implementation based on user-space scripts. Finally, it is worth highlighting that no false positives were experienced throughout the experiments. The results of these experiments demonstrate that SODALITE nodes can provide resiliency in a carrier-grade wireless backhaul network.

C. Evaluating SODALITE Impact on Application Delay

In this section, we study the impact of the FLRR agent on the delay perceived by the application layer in case of link failure. In particular, we compare the delays introduced by FLRR with the ones experienced if a distributed protocol

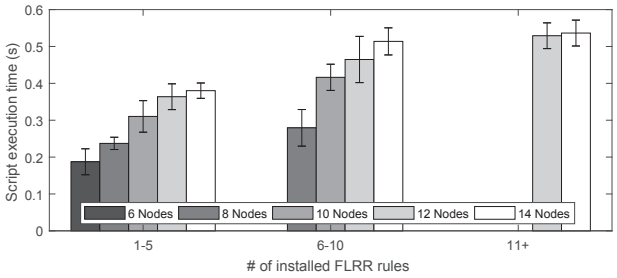


Fig. 10. Average delay T_R between detection of an active switch rule and activation of the backup path for different number of nodes and installed FLRR rules. Note that the amount of rules depends on the number of active flows, which is limited by the size of the scenario: 11+ rules for 12 and 14 nodes, up to 7 rules in 8 node scenarios, and up to 5 rules in 6 node scenarios.

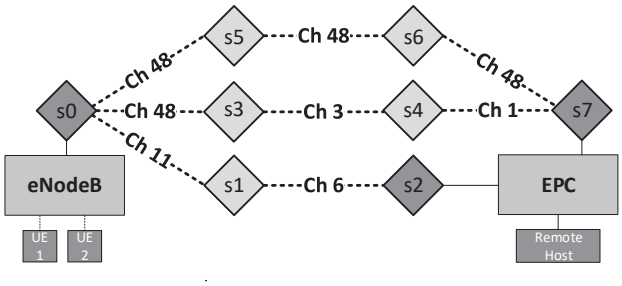


Fig. 11. The network topology used in the NITOS testbed. Node s0 is the entry point of the network and serves as connection to the eNB, whereas nodes s2 and s7 are gateways towards the EPC.

is used. For this purpose, we reuse the random topologies introduced in Section VI-A, and build a virtual network emulator using network namespaces [28] containing our full implementation of SODALITE, including the SDN agent, the FLRR agent and the controller. As a distributed protocol we use OLSR, available in Linux as part of the package *olsrd2*. In order to measure the impact of link failures on application delay, in each random topology a *ping* application is launched between each non-gateway node and its closest gateway with an interval of 10 ms between packets. Then, we break one of the links and measure the time gap caused by the link break, until the flow of ICMP packets resumes. For each topology, this process is repeated choosing a different link to break, until a link break has been caused for all links of the topology.

Fig. 9b depicts the results of our experiment. The convergence of OLSR heavily depends on its timers and, hence, two different OLSR configurations are used. *OLSR-1* uses a Hello and Validity intervals of 0.5 s and 1.5 s, whereas *OLSR-2* uses 1 and 3 s, respectively. These two configurations trade off convergence speed with signaling overhead, and are based on the recommendations in [29]. We can see in Fig. 9b how, for OLSR, the application outage time in case of link failure, lies around 3 s and 6 s for the *OLSR-1* and *OLSR-2* configurations, respectively, which is twice the Validity interval. Instead, SODALITE delivers application outage times of 2 s in the worst case, without the need of trading off the signaling overhead and convergence, thanks to the fact that in SODALITE backup paths are pre-provisioned.

It is worth highlighting that the delay experienced by FLRR in this experiment was limited by the performance of the server hosting the network emulation, and therefore we expect significantly lower delays in practice, where FLRR agents run each on a different hardware (node) rather than aggregated on a single server. To validate that the server is artificially increasing the outage delay for SODALITE, we measure the delay T_R between detecting a packet matching a switch rule, and the reprogramming of the rule to redirect a traffic flow, as the number of FLRR rules installed in a node grows. In a practical setting, T_R would not depend on the size of the network. However, Fig. 10 depicts the average T_R values measured throughout the experiments, revealing that, due to the performance limitations of the server when emulating all nodes on a single machine, T_R increases with the scenario size, even though the number of FLRR rules per node are the same. Hence, in practice we expect application outage times well below 1 second (6 Nodes in Fig. 9b).

D. E2E SODALITE Evaluation in the NITOS Testbed

In order to validate SODALITE in a realistic setting, especially the FLRR agent, we perform a set of experiments in the LTE-enabled NITOS testbed [24]. The experiments show how SODALITE applies forwarding policies to LTE traffic and uses FLRR to react to different critical situations. The focus of our experiments is on the S1 interface, which would be equivalent to the F1 or N3 interfaces in a 5G network.

1) Experimental setup: The *RF-isolated indoor testbed* in NITOS is equipped with a SC, an EPC, and different nodes that can be used as SODALITE nodes or LTE-enabled UEs. We configure eight of those nodes to form a wireless backhaul. Each node in the backhaul is equipped with up to two IEEE 802.11g/n interfaces, thus supporting simultaneous communications on up to two channels. The topology used for the experiments and the channel configuration are shown in Fig. 11. The backhaul includes an entry node (s0) that connects the eNB with the backhaul and two gateway nodes (s2, s7) that connect the backhaul to the EPC.

The SDN controller managing the data plane of the backhaul runs on a remote virtual machine connected to the backhaul via a VPN instantiated on the gateway node s7. The sequential policy is applied in the controller, with a channel occupancy threshold u_{THR} = 0.6 (c.f. Section IV-A). As destination or origin for the LTE traffic generated during the experiments, we set up two nodes as LTE UEs and a node connected to the wired backbone behind the EPC serves as destination or origin of data for communications with the UEs.

As initial step of the experiment, we set up the LTE connections of the two UEs. In this process, the control traffic between the eNB and the EPC traverses our backhaul over a pre-provisioned in-band control path. As soon as the UEs establish their connection, each UE is assigned a unique GTP TEID for its corresponding uplink and downlink flows.

With the knowledge of the GTP TEIDs for each of the UEs, the SODALITE controller assigns main and backup paths for the correspondent end-to-end flows. These paths start at the entry node (s0) and go to any of the gateways (s2, s7) for the uplink traffic and the opposite direction for the downlink traffic. Following the sequential policy, the controller assigns the main path for both uplink flows to the lower branch (s0-s1-s2) and the backup paths over the upper branch (s0-s5-s6-s7).

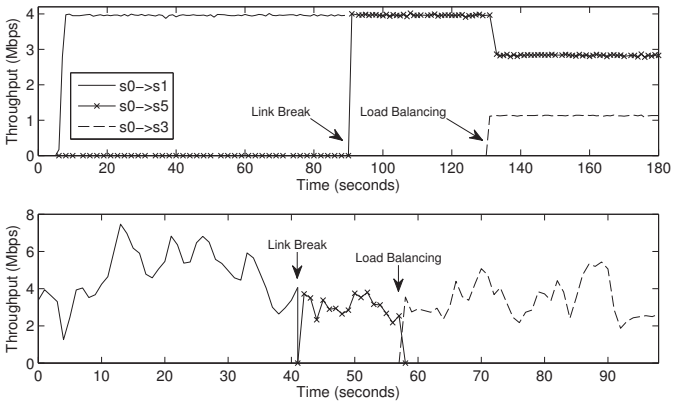


Fig. 12. Aggregate throughput transmitted over s0 towards neighbor nodes of the backhaul over the course of the UDP (top) and TCP (bottom) experiments with two (UDP) and one (TCP) flows, respectively.

The downlink flows are assigned to use the middle branch (s7-s4-s3-s0) as main paths and the upper branch as backup paths, respectively. To initially allocate the paths, a load of 1 Mbps for each end-to-end flow is assumed.

2) UDP-based traffic experiment: As performance metrics, we measure how the overall throughput evolves before, during, and after the link break. Based on the initial setup described earlier, the first experiment consists in launching two UDP-based *iperf* data streams, one from each of the UEs, towards the server located behind the EPC. The first flow (f_1) is set to 1 Mbps, and the second flow (f_2) to 3 Mbps. The sum of the weight of the flows is intended to lie below the actual capacity of the wireless links, being limited on the sender side by the *iperf* application; both flows start at second 5. In the upper part of Fig. 12, the aggregate traffic sent over the radio by s0, the entry node of the uplink flows, is visualized. Initially, a total of 4 Mbps is maintained over the lower branch (s0-s1-s2).

After 90 seconds, we interrupt the link between s1 and s2 by shutting down the wireless transceiver on s2. The FLRR agent detects the link break and redirects the flows, as described in IV-B. The effects can be seen in the upper part of Fig. 12 as the measured load switches from the link s0-s1 to the link s0-s5. The link break detection and traffic redirection over the backup path takes less than 1 s. In this process, we observe that the throughput during one *iperf* report interval of 1 s drops slightly to 3.8 Mbps, after which the full throughput of 4 Mbps is reestablished, thus proving the minimal disruption introduced by the FLRR agent.

At this point of the experiment, f_1 and f_2 both have switched from the broken main path on the lower branch to the backup path on the upper branch. The links on the backup path are on the same channel and, as a result of both flows using the same backup path, the channel occupancy increases drastically up to 71%. This exceeds the maximum allowed channel load threshold of 60% configured in the SODALITE controller and triggers a new path computation. At second 130, the controller decides to reallocate f_1 over the middle branch of the topology, separating the two end-to-end flows. The moment at which the controller takes this decision depends on the periodicity of the network status check, the actual bitrate of the links in the network and the filtering algorithms applied by the controller to avoid ping-pong effects. As a result, the peak channel load observed over the upper branch is reduced to 54%, with only one active flow. Notice that f_1 is not relocated immediately, because a 15 second interval is used to collect OF statistics and the controller applies filtering to the received statistics to avoid ping-pong effects. Given that FLRR quickly resolves a link break, a larger statistics polling interval is considered appropriate to reduce signaling overhead. However, these values can be configured more aggressively if a faster overall optimization of the network is required.

3) TCP-based traffic experiment: The second experiment is performed with the same base setup, but using a single TCP Cubic flow launched from the first UE towards the server behind the EPC. In this case, *iperf* does not limit the throughput, which depends on the rate adaptation mechanisms of TCP and on the variable bandwidth in the LTE link and the wireless backhaul links. Following the sequential policy, the controller allocates this flow on the lower branch and assigns the backup path on the upper branch. The transmission lasts for 41 seconds, after which the link between s1 and s2 is manually broken by shutting down the wireless transceiver of s2. As shown in the lower part of Fig. 12, which plots the throughput measured at the node s0, the FLRR agent reacts immediately by redirecting the flow over the backup path (from s0-s1 to s0-s5). The variations of throughput measured over the course of the experiment are caused by fluctuations in the LTE link and in the backhaul links. Note that the UDP experiment does not show those fluctuations since the links were not saturated (traffic limited on the application side). Since each of the three hops on the backup path now is on the same radio channel, the achievable end-to-end throughput is reduced as the wireless transceivers of s0, s5, and s6 are competing for the channel access. Further, since the channel load now exceeds the maximum threshold of 60%, the SODALITE controller reallocates the flow to the middle branch, via s3, on second 57. Like in the UDP-based experiment, the exact moment this reallocation decision is taken may vary depending on when the last network status check was performed and other internal parameters of the controller.

The reallocation from the upper to the middle path happens almost immediately, a small reduction of the throughput is only observed during seconds 57 to 58. Immediately after that we can see that the average throughput of the end-to-end flow increases, as the links of the middle branch lie on separate channels. However, shortly after, the newly allocated path also suffers from time variation, which is however not due to the backhaul but due to quality fluctuations in the LTE link.

VII. Conclusions

Innovative wireless backhaul technologies are required to foster the adoption of massive deployments of SCs, which are a key component of future 5G networks in dense scenarios. In this paper, we have demonstrated SODALITE, a novel system that follows the SDN architecture, and integrates with 4G and 5G Mobile Networks in order to provide per-subscriber/application policies in the wireless backhaul segment. SODALITE is composed of a centralized controller, which derives per-session main and backup paths, and a

distributed agent acting locally on each node, which ensures fast recovery in case of link failure. Through an extensive experimental evaluation, we have benchmarked the performance of SODALITE using randomly generated topologies, and a testbed composed of an LTE network and eight wireless backhaul nodes. Through mathematical analysis and using traces from a mobile network operator, we also show the scalability of SODALITE both in terms of its negligible signaling overhead and the limited number of flow rules it requires at the backhaul nodes. The thorough analysis of SODALITE shows that it provides resiliency and fast reaction times necessary for a carrier grade wireless backhaul network. As future work we consider the study of enhanced path-allocation policies that accomodate application priorities and leverage traffic predictors to proactively alter backhaul paths.

Finally, although the system designed in this paper has been based on devices using IEEE 802.11 interfaces, SODALITE is amendable to other types of radio technologies, such as mmWave. In addition, the FLRR scheme can be extended to consider more than one backup path in case of multiple failures. We consider this as part of our future work.

ACKNOWLEDGEMENTS

The authors thank Ferran Quer i Guerrero for his contributions. The research leading to these results has received funding from the European Union under grant agreements 762057 (H2020 5G-PICTURE), the Spanish Ministry of Economy and Competitiveness (MINECO), through projects TEC2016-76795-C6-2-R, RYC-2013-13029 and FEDER.

REFERENCES

[1] METIS-II: Mobile and wireless communications Enablers for Twenty-twenty (2020) Information Society-II (H2020-ICT-2014-2), "D2.1. performance evaluation framework," 2016.
[2] Ericsson, "Microwave outlook, trends and needs in the microwave industry," 2016.
[3] Y. Zeng, P. H. Pathak, and P. Mohapatra, "A first look at 802.11ac in action: Energy efficiency and interference characterization," in *2014 IFIP Networking Conference*, June 2014, pp. 1–9.
[4] M. S. Afaqui, E. Garcia-Villegas, and E. Lopez-Aguilera, "IEEE 802.11ax: Challenges and Requirements for Future High Efficiency WiFi," *IEEE Wireless Comm.*, vol. 24, no. 3, pp. 130–137, June 2017.
[5] H. Zhang, Y. Dong *et al.*, "Fronthauling for 5G LTE-U Ultra Dense Cloud Small Cell Networks," *IEEE Wireless Comm.*, vol. 23, no. 6, pp. 48–53, Dec 2016.
[6] P. Rost, C. J. Bernardos *et al.*, "Cloud technologies for flexible 5G radio access networks," *IEEE Comm. Mag.*, vol. 52, no. 5, pp. 68–76, May 2014.
[7] E. Garcia-Villegas, D. Sesto-Castilla *et al.*, "SENSEFUL: An SDN-based joint access and backhaul coordination for Dense Wi-Fi Small Cells," in *IWCMC 2017*, June 2017, pp. 494–499.
[8] K. Pentikousis, Y. Wang, and W. Hu, "Mobileflow: Toward software-defined mobile networks," *IEEE Comm. Mag.*, vol. 51, no. 7, pp. 44–53, July 2013.
[9] R. Misra, A. Gudipati, and S. Katti, "QuickC: Practical Sub-millisecond Transport for Small Cells," in *MobiCom 2016*, 2016, pp. 109–121.
[10] A. Detti, C. Pisa *et al.*, "Wireless Mesh Software Defined Networks (wmSDN)," in *WiMob 2013*, Oct 2013, pp. 89–95.
[11] H. Huang, P. Li *et al.*, "Software-defined wireless mesh networks: architecture and traffic orchestration," *IEEE Network*, vol. 29, no. 4, pp. 24–30, July 2015.
[12] A. Betzler, F. Quer *et al.*, "On the benefits of wireless SDN in networks of constrained edge devices," in *EuCNC*, June 2016, pp. 37–41.
[13] N. L. M. v. Adrichem, B. J. v. Asten, and F. A. Kuipers, "Fast Recovery in Software-Defined Networks," in *EWSDN 2014*, Sept 2014, pp. 61–66.
[14] J. Vestin and A. Kassler, "Resilient SDN based small cell backhaul networks using mmWave bands," in *IEEE WoWMoM*, 2016.
[15] 3GPP TS 23.501, "Technical Specification Group Services and Systems Aspects; System Architecture for the 5G system; Stage 2," 2018.
[16] 3GPP TS 38.331, "Radio Resource Control (RRC) protocol specification (Rel. 15)," 2018.
[17] D. C. Mur, P. Flegkas *et al.*, "5G-XHaul: Enabling Scalable Virtualization for Future 5G Transport Networks," in *IUCC-CSS*, 2016.
[18] Open Networking Foundation, "The OpenFlow Switch Specification." [Online]. Available: https://www.opennetworking.org
[19] A. Hurtado-Borrs, J. Pal-Sol *et al.*, "SDN wireless backhauling for Small Cells," in *IEEE ICC*, June 2015, pp. 3897–3902.
[20] R. Draves, J. Padhye, and B. Zill, "Routing in Multi-radio, Multi-hop Wireless Mesh Networks," in *MobiCom*, 2004, pp. 114–128.
[21] B. Stephens, A. Cox *et al.*, "PAST: Scalable Ethernet for Data Centers," in *ACM CoNEXT*, 2012, pp. 49–60.
[22] J. Yang, Y. Qiao *et al.*, "Characterizing User Behavior in Mobile Internet," *IEEE Tran. on Emerging Topics in Computing*, vol. 3, no. 1, pp. 95–106, March 2015.
[23] J. Huang, F. Qian *et al.*, "A Close Examination of Performance and Power Characteristics of 4G LTE Networks," in *MobiSys*, 2012, pp. 225–238.
[24] K. Pechlivanidou, K. Katsalis *et al.*, "NITOS testbed: A cloud based wireless experimentation facility," in *ITC 2014*, Sept 2014, pp. 1–6.
[25] I. Demirkol, D. Camps-Mur, and J. B. and, "5G Transport Network Blueprint and Dimensioning for a Dense Urban Scenario," in *EuCNC*, 2017.
[26] IEEE 802.11, "TGn Channel Models." [Online]. Available: https://mentor.ieee.org/802.11/dcn/03/11-03-0940-04-000n-tgn-channel-models.doc
[27] R. Patidar, S. Roy, and T. R. Henderson, "Technical report on validation of error models for 802.11n," University of Washington Seattle, Tech. Rep., 05 2017.
[28] Various authors, "LXC." [Online]. Available: https://linuxcontainers.org/
[29] C. Gomez, D. Garcia, and J. Paradells, "An OLSR parameter based study of the performance of real ad-hoc network environments," in *European Wireless 2005*, April 2005, pp. 1–6.

August Betzler is a research engineer at i2CAT in Barcelona, Spain. His research topics are SDN in wireless and mobile networks. He contributes to the standardization of new communication protocols for the Internet of Things within IETF. In 2010 he received his Diplom degree in computer science from the Technical University of Hamburg and in 2015 his Ph.D. from the Universitat Politècnica de Catalunya (UPC).

Daniel Camps-Mur currently leads the Mobile and Wireless Internet group at i2CAT. Previously, he was a senior researcher at NEC Network Laboratories. In 2004 he received a Masters degree and in 2012 a Ph.D. degree from the UPC. His research interests include mobile networks, SDN and communications protocols for the IoT.

Eduard Garcia-Villegas is an associate professor at the UPC and member of the Wireless Networks Group (WNG). He participates in the IEEE P802.11 WG and in the research developed within the i2CAT Foundation. His research interests include IEEE 802.11, radio resource management in wireless networks, and IoT enabling technologies. (sensor networks, mesh, multi-hop ad-hoc networks, etc.).

Ilker Demirkol is a Ramon y Cajal Research Professor in Dept. of Mining, Industrial and ICT Engineering at the UPC. His research focus is on communication protocol development and performance evaluation of wireless networks. He received his BS, MS, and PhD degrees in Computer Engineering from Bogazici University, Istanbul, Turkey.

Joan Josep Aleixendri is a research engineer at i2CAT in Barcelona, Spain. His research topics are software defined networking and wireless networks. In 2016 he received his Bachelor degree in computer science from the UPC.