Autonomous Edge Resource Organization with Smallcell Integration in 5G

Ahsan Malik, Xun Xiao, Ramin Khalili,
Zoran Despotovic and Artur Hecker
German Research Center, Huawei Technologies
München, Germany
{ahsan.n.malik, xun.xiao, ramin.khalili,
zoran.despotovic, artur.hecker}@huawei.com

Mayutan Arumaithurai and Xiaoming Fu
Institute of Computer Science
Georg-August-University of Göttingen
Göttingen, Germany
{mayutan.arumaithurai, fu}@cs.uni-goettingen.de

***Abstract*—In the upcoming 5G era, many new use cases (e.g. emergency situations, crowded sports event coverage, etc.) are expected to be supported. Additionally, to improve latency and jitter, 5G is likely to support edge computing capabilities, e.g. in an NFV and MEC fashion. These features and requirements call for support for autonomous networking, where links, nodes and their services cannot be presumed permanent, but rather have to be dynamically used, when available and suitable. In this paper, we propose a new design framework, named Aerosmith, which provides autonomous edge resource organization with smallcell integration and topology handling. We motivate our design, explain the key architectural decisions and our prototype implementation, and provide initial evaluation results.**

***Index Terms*—Autonomous, Small Cells, MEC, 5G Networks**

I. Introduction

In the previous generations of mobile networks, the connectivity provision to the base stations (BSs) mostly relied on available copper and optical means or used microwave or satellite backhauls. In 5G [1], the upcoming next generation, new use cases have been identified (such as support for emergency situations, crowded sports event coverage, etc.) in 3GPP SA1 [2]. Those new use cases require a rapid, sometimes even on-demand, rollout or extension of a cellular network into a new area, often with guarantees, e.g. enough capacity, high reliability or low latency.

Radically novel in these emerging use cases is the unforeseeable usage patterns at the moment of network deployment: in 5G, both the network topology (BS placement, their interconnection, etc) and the network services will change during network lifetime. Thus, additionally to long-term network planning, rapid operational decisions will have to be used: as backhauls and network topologies cannot be planned cost-efficiently in this situation, the future mobile networks must be more adaptive and malleable. For instance, small mobile cells could be placed to reach BSs with good backhaul connectivity. Similarly, an available macro-cell fronthaul can be used as aggregate backhaul. In the newest products, basic provisions for both BS-to-BS connectivity (e.g. X2 interface) and fronthaul-backbaul integration (e.g. Huawei's X-HAUL[1]) are available via pre-configuration.

[1]http://www.telecomtv.com/articles/5g/huawei-launches-x-haul-mobile-bearer-solution-for-5g-networks-15883/

The pre-provisioning approach is problematic due to the fact that the connection of a BS (both to the mobile core network services and to the public data networks) might be provisioned over an unstable path, subjected itself to control and management decisions of the (same) operator. Specifically, redirecting flows triggered by traffic engineering due to load fluctuations can easily lead to control path instability or its loss. Consider e.g. a scenario, where an operator downgrades or blocks forwarding on a port of one BS, currently relaying control plane communications of some other BS; today, that would lead to a full disconnect of the relayed BS from the mobile core network. Coping with such situations from a central point of view of an operator requires exact understanding of the current, albeit dynamic, network topology and a precise schedule of operations, where commands must follow specific order to avoid lockouts. This is difficult to implement in a scalable way.

In addition, to improve latency and jitter, 5G is likely to support edge computing capabilities, e.g. in NFV [3] and MEC [4] fashion. This requires dynamic deployments of network functions (NFs) in 5G BSs, which allows to deploy core network services directly on the edge. Hence, 5G BSs should be also able to discover, select and connect to the most suitable NF instances. In fact, 3GPP SA2 has recently defined a service-based architecture (SBA) with a service-based interface (SBI) between NFs to support similar provisions in the core [5], [6]. Clearly, a mechanism is needed to allow MEC-enabled 5G network nodes (e.g. BSs) to dynamically handle their participation in the SBI as well.

Beyond the well-known radio coordination problems, the resulting trend to denser but more dynamic small cells in 5G raises new questions with regard to the integration, connectivity and coordination of the numerous BSs and the whole system architecture. Overall, these features and requirements call for support for autonomous networking, where links, nodes and their services cannot be presumed permanent, but have to be dynamically used, when available and suitable. This is radically novel in the mobile network area, rather characterized by strong operator involvement, network planning activities and reliance on session-oriented protocols used over guaranteed and operator-provisioned connectivity. To the best

of our knowledge, we are not aware of any previous work that solves both a dynamic interconnection of many small cells in the mobile network RAN and allows their autonomous establishment and maintenance of an SBI.

In this paper we present Autonomous Edge Resource Organization with SMallcell Integration and Topology Handling (*Aerosmith*) solution. More concretely, this paper makes the following contributions:

- We design a novel, dedicated and simple, architecture to support fully autonomous resource integration, connectivity and backhaul maintenance to support rapid (e.g. mobile BS) deployment.
- We propose necessary mechanisms to enable an autonomous, i.e. even core-less, BS operation in the future generations of mobile networks, where BSs participate in an existing or span a new SBA support layer.
- We implement our proposals using real networking stacks and evaluate the proposed mechanism. Our results show the feasibility of our solution and provide insights about its design.

Our proposal is complementary to SA2 work in 3GPP, providing a solution for SBI. With Aerosmith, the 5G system could be living on the edge of the small cells.

The rest of the paper is organized as follows. In Section II, we describe our formalization of the problem, i.e. our modeling based on the above observations. In the Section III, we present technical details of our solution. After that, simulation results are presented in Section IV. Finally, we survey some related work and conclude the paper.

II. System Model

In our system model, we assume a mobile RAN covering a geographical area. Formally, we denote the mobile RAN infrastructure as $G := \langle S, E \rangle$ where S is the set of BSs, in which every s_i can have networking, compute and storage capabilities (but with different and varying resource because of allocations). E is the link set, in which every link e_{ij} is a bidirectional communication link between two BS nodes s_i and s_j. Over the network infrastructure G, BSs need a control plane interconnectivity so that NFs running across the infrastructure can communicate with each other to provide network services.

Since the network topology is dynamic, both the BS set and the link set E are not static and could change due to the network churn (e.g. any possible node/link failures, node joining/leaving, and/or port blocking by NFs). As a result, the control plane connectivity could also be affected and network services could be broken. Our solution provides a self-organized control plane establishing such interconnectivity and supporting service discovery. The control plane constructed spans all available resources within a control realm. We will introduce details of our solution in the next section.

Note in the previous paragraph an important detail related to the terminology that we use in this paper: We use the term *resource control plane* (or just control plane) to denote a set of resources (or agents that manage them, see next section), along with a set of protocols to establish and maintain interconnectivity among them, as well as provide basic storage capabilities. The later term, control plane namely, will be more frequently used, i.e. whenever there is no danger of confusing it with the control plane of a mobile network.

III. Our Proposal

Our solution features a zero-configuration, self-bootstrapping and self-maintaining communication service supporting different types of resource control plane communications. It supports both resolution of end-points, node to node path setup and maintenance and piggy-backing messages of other protocols on top of the control plane (e.g. OpenFlow or network management protocols like NETCONF etc). The rationale behind our design is that 5G, and future networks in general, should exhibit more autonomies, i.e. giving away management and adopting more control. For example, failures in the network should not be handled through an ensemble of the tasks such as network planning for resilience and network management, but rather resolved by the network itself through its self-healing properties. That should also, in our opinion, resonate better with the SB architecture of 3GPP [5], [6].

A. Infrastructure Requirements

There are two requirements on the infrastructure for the proposed solution to work:

1) Every node in the infrastructure must execute a *resource control agent* (RCA).
2) Every RCA on an infrastructure node must have a local connectivity to at least one other RCA.

An infrastructure (resource) node can be any node in the network, e.g. a BS, a server hosting a specific network function or a data center that hosts a multitude of them.[2] The RCA is a piece of dedicated software designed for and running on the BS as implemented by the vendor or integrator of the resource. As a rough example, RCA can be compared to standard agents (e.g. OpenFlow or SNMP) in the network appliances.

The execution of the RCA is just the representation of the controllability of the resource and can be compared to the presence of OpenFlow clients in OpenFlow switches. As for the connectivity requirement, it means that we do not consider lower layer channel/transport issues here; the channel can be a physical medium (layer 2 link) or a virtual channel, spanning uncontrollable resources (public paths, VPN, etc). This also answers the question whether everything needs to be controlled. If the condition 2 is not fulfilled, then the mobile RAN essentially falls apart in two (smaller) networks. Interestingly, each part will remain controllable, as long as there is a control end point within it.

RCA needs some initial configurations. The self-bootstrapping allows to minimize that configuration to the bare minimum: every RCA only needs to have an

[2]We will often use the term BS to mean an infrastructure node. There is nothing radio access specific in our usage of the term base station.

identifier and a security association (SA). An identifier can be derived from the resource (e.g. the MAC address of the management port, etc) and depends on the available SA: in practice, we expect it to be attached to a private/public key pair. The SA must have a control realm name. This name can be explicitly given as a string, or it could result from the signatory information in the certificate (e.g. X.509) that might be used to confirm the public key to *id* binding.

In our solution, the RCA acts both as a local resource control element and as a control plane peer. The local resource control part mainly provides access to the controllable objects of the resource. RCA internally links the local control and the *RCA peer*. For instance, it uses local log files, errors and alarms to reconfigure the constructed control plane in case of problems, enabling local and, thus, fastest detection. For the latter as an RCA peer, the RCA implements the resource-to-resource (R2R) protocol suite, enabling it to exchange messages with other RCAs.

B. RCA Peer Operations

Interactions between RCA peers are the main activity constituting the control plane. In an RCA, a Message Processor is responsible for the internal treatment of the R2R protocol and the delivery of the contained messages. R2R protocol runs when the RCA starts up, it periodically runs Friend Discovery (FD), neighbor selection (NS) and Routing modules. The output of FD module is made available to the NS module. The output of NS module is made available to Routing module. Additionally to periodic execution, the RCA runs these modules in case of alarms. We here further detail the activities in different phases.

1) Friend Discovery: When a node of a mobile RAN (i.e. a BS) initially comes up, its RCA needs to find other RCAs from the same control realm. This is usually referred to as bootstrapping. The only input to this process is the initial configuration and, if available, the friend list from previous operations. RCA should use a mix of different approaches to find friends (UPnP, mDNS, SDP, DNS-SRV, special nodes, etc) depending on infrastructure capabilities and the nature of the network element. The overall active friend list is regularly updated and made available to the NS module.

2) Neighbor Selection: Given a list of active friends, the decision becomes necessary, which of them an RCA will connect to, i.e. which of these it will consider as control plane routing next hops. Assuming that freedom of choice exists in the infrastructure, that decision is paramount because the choice of the nodes defines the overlay structure of the control plane. That structure has a major impact on the resulting per-node state and possible communications in the control plane as well as the resilience of the whole. We engineered the control plane to fulfill QoS and resilience requirements in a pragmatic manner, in particular without overloading resources.

Different strategies for neighbor selection typically lead to various types of overlay structures from graph-theoretical perspectives. If most of the nodes select several specific nodes in the network as friends, then such a selection policy will lead to a centralized topology. Oppositely, if everyone randomly selects its friends, the resulting topology will be more random. In between, the topology will present scale-free properties. Alas, one realizes that a perfect structure does not exist. Higher degree structures exhibit a better resilience and good QoS posture but induce a high maintenance cost and scale badly. Scale-free networks have very good scaling properties but rely on high degree nodes that are easy targets for an intelligent attacker. Balanced solutions such as uniform degree networks or generic small worlds represent trade-offs: while their resilience is worse than that of random networks or full meshes, their QoS guarantees are lower than what can be achieved with full meshes or scale-free networks.

Since no structure exhibits all desired properties, the neighbor selection phase makes the topology of the constructed control plane be structurally adaptive. We refer to this process as rewiring: following a trigger, the RCA is able to find other neighbors in order to adopt the appropriate system-level shape, following the service degradation doctrine. Instead of guaranteeing all properties at all times, this idea advocates guaranteeing some properties depending on the context. For instance, in normal operational conditions, good QoS and low node foot-print would appear as primary objectives. However, in case of repetitive faults and errors, the system can go into a degraded or protected mode, where resilience prevails on quality, and this up to a point where the communication becomes restricted to only essential commands.

In dire straits, what matters is that the operator/decision layer can still reach the RCA, so as to be able to get status reports and/or to apply a different configuration.

3) Routing: Once all RCAs select their friends, the RCAs can run separate, classic distributed routing protocols on the available neighbors so as to decide how to reach each destination (e.g. distance vector or link state protocols resulting in the next hop neighbor for any destination). At least two alternatives to this are available:

- One could bind the routing to an infrastructural metric instead and only use neighbors, which are closer to the RE of the RCA in the transport infrastructure.
- One could bind the routing directly to the neighbor selection by passing the message from one RCA to another according to a common metric, e.g. distance from the intended recipient to *id*s of known neighbors. This would allow to construct a distributed resolution service.

We choose the latter way. We use the network *Linearization* algorithm proposed in [7], an *id*-based structured routing protocol, for the routing module. Linearization is a simpler form of what has been proposed in [8]. [8] also presents a nice discussion on why id-based routing is at all a good choice for routing. We point the interested reader to that discussion, instead of repeating it here.

Along with routing, it is straightforward to organize a distributed storage across the BS nodes, since the stored objects and the RCA *id*s can be easily projected to the same space using a hash function, similar to distributed hash tables (DHTs) [9]. This is even easier in our implementation thanks

to the choice of the *id*-based structured routing. With this, it is straightforward to store objects at a BS node closest to the object. For higher availability, redundant storage can also be easily organized, e.g. using several hash functions. This storage is currently used as strictly internal to control plane to allow to store global status and configuration information, but can be easily extended to serve as an efficient implementation of the Universal Storage (USM) and Unstructured Data Storage (UDSF) of 3GPP TS 23.501 [5].

C. Remarks

Aerosmith (especially, the constructed CP) is designed as a multi-dimensional and dynamic interconnection of BS resources within the same control realm. We delegate the construction and maintenance of a CP to the RCA running at every BS. This is a paramount design principle: first, doing so has the advantage of being capable to react to local events in the fastest manner; moreover, with smart algorithms, this provides a good level of isolation from the decision layer: even if decisions are disastrous with respect to the immediate situation of a resource, they can be locally corrected so that at least the logical binding to the CP and, hence, the decision layer is always maintained.

As a simple example, if an SDN controller (i.e. a control purpose NF running on a resource node) sends a flow rule to a controlled switch s_1 to block forwarding on a port currently used for control plane communications of some other switch s_2, in todays SDN that would lead to a full disconnect of s_2 from the controller. This is particularly critical in OpenFlow-based networks with an in-band control plane (or called control channel). However, in our solution, the local RCA on s_2 will reconnect to a different suitable RCA (e.g. to the one on s_1). Once reconnected in such an indirect way (RCA on s_1 acts as a proxy for the RCA on s_2), s_2 will notify the responsible controller of the situation. The controller now can raise an alarm or stop the responsible control application or restore the previous state.

In the examples above, one could argue that the problem can be centrally considered and solved at a network control application. However, requiring correctness there would result in a considerable complexity and unnecessary customization of network control applications, increasing the development effort and impeding their portability. The reasons are as follows. First, the control graph details are irrelevant to a casual developer and, ideally, should not play any role, so that network applications can be reused in other control realms or in changed infrastructure conditions. Second, this requires instantaneous knowledge of the overall infrastructure situation, which is a very hard requirement for a distributed system at hand. Therefore, requiring correct decisions from a decision layer is an unrealistic assumption and a bad design choice. It would strongly limit the admissible infrastructure dynamics, since non-local decision points cannot be informed in zero time and might make wrong decisions. Further, it would heavily shrink the scalability of the solution by imposing tight synchronization requirements. Most importantly, it would strongly increase the cyclomatic complexity of all network applications, which contradicts the ease of programmability.

The proposed solution solves the issues on control plane establishment and particularly its maintenance later on. With a distributed system operating between RCAs and embedded in the network infrastructure, it abstracts the controlled BSs, integrates autonomously newly deployed BSs, keeps all BSs connected, and makes the resources on the BSs always available to the developers of the network. The solution is one of the key components enabling a fully programmable but transparent network infrastructure for future mobile networks.

IV. Evaluation

In this section, we show the evaluation results of different aspects of RCA peer operations (cf. Section III). Every node executes an RCA as per explanation above (cf. Section III-B).

To check the feasibility of Aerosmith, as well as assess the costs associated with its creation and operation, we have a running, proof of concept implementation of the system. Our resource is a Linux host, running an OpenVSwitch (OVS) [10] as the networking module and using the local file system for storage purposes, implemented by 1000 lines python code. The largest part is for routing module. The RCA design is modular so that other protocols(e.g. RSTP or TRILL) can be easily integrated as routing modules.

A. Evaluation Environment

Our RCAs run as OVS extension under Linux. Besides, the same RCA is used on the node running the control purpose NF (we use FloodLight controller[3]). In our evaluation environment, each extended OVS represents a dynamically deployable 5G small cell.

For topology deployment, we use a slightly altered version of Mininet[4], which permits us to run big topologies of such initially unconfigured N OVS nodes and 1 controlling nodes. No routing and no other special provisions are implemented (no routing protocol, no STP or other L2 auto-organization solutions; the nodes even have no initial IP configuration, no controller configuration, etc). The node degree in our generated random topologies is 3 on average.

On start-up, our local RCA installs initial basic flow rules in the OVS so as to capture all R2R traffic; it also starts sending out R2R packets over its local OVS using PACKET_OUT. Again: note that by the nature of RCA, there is no controller yet at this stage. An RCA can only follow the procedure in Section III-B to discover other RCAs in its neighborhood and to actually construct and maintain the control plane.

We are interested to study the system bootstrapping time, convergence time, and resuming time (these terms will be explained in details in the following sections) on topologies of different sizes (e.g. 50, 100, 150 and 200 nodes). We compare the performance of two alternative schemes used by the routing module in the proposed solution. The first is our choice (i.e. Linearization algorithm [7]) and the other one is the default choice in the current OVS implementation (i.e. STP).

[3]http://www.projectfloodlight.org/floodlight/

[4]http://mininet.org/

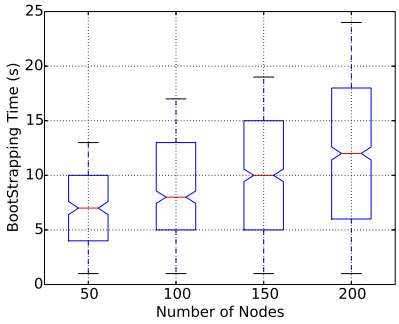

Fig. 1: Bootstrapping time of [7]

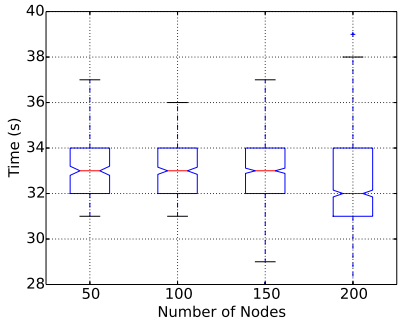

Fig. 2: Bootstrapping (Convergence) time of STP

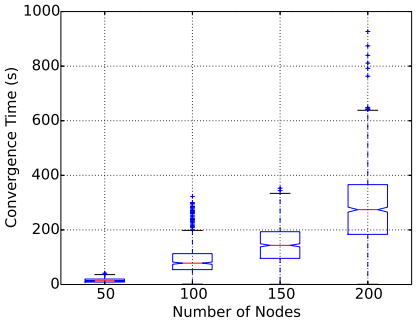

Fig. 3: Convergence time of [7]

B. Control Path Bootstrapping Time

In this experiment, we are interested in the control path bootstrapping time, i.e. time till RCAs are able to forward the control plane messages to a selected endpoint (e.g. a controller running on a specific network node), so that the TCP control connection can be established from the OVS to those endpoints in the network.

In the Figure 1, we can see that Linearization algorithm bootstraps all control paths after 7 seconds (resp. 8, 10 and 13 seconds), for a network of size 50 nodes (resp. 100, 150, and 200 nodes), while STP as shown in Figure 2 requires 33 seconds (resp. 33, 33 and 32 seconds) to establish the control plane connectivity. Hence, STP is a factor 4-7 times slower than Linearization algorithm to bootstrap the network.

This experiment illustrates how long different choices for the routing module take so that every nodes can connect to a specific node in the network. It is quite relevant to the future CP/DP-decoupled network architecture where every network node does not have to connect to every other nodes but only a controlling endpoint of the network.

C. Full Convergence Time

We now measure the convergence time, i.e. the time it takes from the start-up of a completely unconfigured network until each and every OVS can directly communicate with everyone else. In other words, the convergence time is the time from the startup until all RCAs reach a stable state in a given network. In this experiment, during and after the convergence, the network is not structurally altered (no new nodes are added, no nodes are removed, no new links are added/removed).

Full convergence is a very conservative estimate: usually, nodes can communicate to the controlling node much earlier as shown in the previous experiment, using another RCA to forward their messages to some other RCA, until these finally reach the controller[5].

[5]Recall that constructed control plane using Linearization algorithm is not a single-rooted tree (like STP/RSTP) but can actually construct structurally different topologies, always supporting path diversity. Therefore, its scalability for forwarding of the traffic is much better; this however comes at a price of higher yet still realistically usable convergence times.

In Figure 3, the RCAs reach a stable state after about 20 seconds (resp. 70, 130, and 150 seconds) for a network of 50 nodes (resp. 100, 150, and 200 nodes) when the Linearization algorithm is used. With STP whose convergence time equals bootstrapping time, RCAs reach a stable state after 32 seconds on average (ref Figure 2). After this time, the network has fully converged, i.e. all keep-alive messages exchanged after the convergence are without effect, as long the topology remains unchanged.

D. Handling Network Link Failure

We also tested the capability of the proposed solution to handle network dynamics, e.g. due to link failures. Specifically, we tested how much time it takes to resume the control plane connectivity for those nodes affected by link failures occurred in the network. In this case, we compare the performance of the Linearization algorithm with RSTP [11], which is an extension of the original STP proposed to handle network dynamics.

We first report the evaluation results for RSTP from section 2.1 of [12]. They show that in a small network, with 4 to 20 nodes, and when only a single link failure occurs, it takes 1 to 5 seconds to resume the connectivity of the affected nodes. According to our literature study, no results are available for multiple link failures when RSTP is used, but we expect the time to be larger than what is reported in [12]. This would be evaluated in our future study through measurements.

To perform our test on the Linearization algorithm, rather than breaking only a single link, we randomly selected a number of links, removed them from the network, and then measured the time until the control plane connectivity is fully re-established, referred to as resuming time. We measure the resuming time for two different network sizes, 100 and 150 nodes, and for different fractions of link failures of the links, namely 10% and 20%. The results are depicted in Figure 4. We observe that, in the median, less than 8s are needed with 10% link failures in both network size cases to fully resume the control plane communication, while with 20% link failures, around 10s are needed for both network size cases.

These results indicate that by selecting an appropriate routing scheme, we can provide fast bootstrapping time and

fast resuming time, even when multiple link failures happen. We should mention that further study is required to completely understand the impact of the selected routing mechanism on these performance metrics.

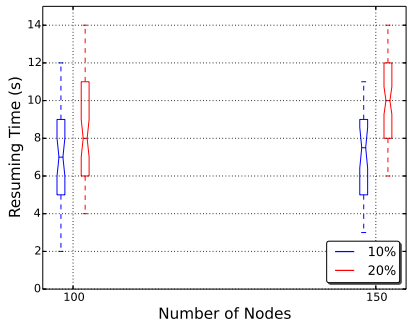

Fig. 4: Resuming time (in seconds) with network link failures of different sizes using Linearization in [7]

V. Related Work

Establishing a CP for a mobile network is challenging. In current generations, the network infrastructure is rather static and CP setup is done by management with professional expertise. If network failures and mistakes happen, usually on-site repairing and re-configurations are required. This problem becomes more severe if we have much denser small cells in 5G [13]. In order to support various network services, network protocols have to be pre-installed on the network devices. For example, OSPF [14] and BGP [15] have to be configured on routers. According to specific requirements, engineers have to prepare the configurations for the devices. In future, radically novel in emerging use cases is the unforeseeable service patterns at the moment of network deployment in 5G. This makes pre-planning and static methods inefficient.

Programmability of a network infrastructure facilitates rapid deployments of network services and fine-grained network flow control. However, a critical prerequisite is the availability of a functioning control network. Such a control network so far is not autonomous and cannot be established without human intervention. Hence, in 5G with high dynamics, CP's autonomy is even more critical than before.

IETF already recognized the CP connectivity issue [16]. It proposes *autonomic control plane* by including an existing IPv6 routing protocol. Its idea is to provide a hidden and logically separated control plane for traditional IP networks. In reality, the OVS implementation [10] also recognizes the CP establishment issue where standalone STP in the linux networking stack is used to establish an in-band CP connectivity. However, it does not achieve the CP's autonomy while out-of-band capability (i.e. linux stack) is implicitly utilized.

In [17], in-band control is built based on hybrid network nodes where DHCP is used to locate the controlling point. Restoration of the CP connectivity relies on calculating alternative paths, and protection of the control channel relies on pre-calculating one disjoint path for every controlled nodes. One feature of this work is that queuing mechanism is integrated so as to better serve the control traffic forwarding because of its high priority. Similarly, the work of [18] proposed to integrate an OSPF module aside in the switch.

VI. Conclusion

The envisioned 5G use cases make the network topology and network services unforeseeable. Control plane establishment and maintenance become thus the key challenge because any pre-planning and static configuration are inefficient and unrealistic. In this paper, we proposed Aerosmith, our solution to solve the problem. Aerosmith is responsible for handling the effects of network dynamics, it identifies network resources and abstracts their usages, particularly considering the programmability of the network. The proposed solution makes the difficulties of control plane establishment and maintenance transparent to the network developers. Our prototype validates our idea and provides insights for the resource control at the edge of a 5G mobile network.

Acknowledgment

This work was partly funded by the EU H2020 5G XHAUL project (H2020-ICT-2014-2 671551).

References

[1] A. Gupta and et al., "A survey of 5g network: Architecture and emerging technologies," *IEEE access*, vol. 3, pp. 1206–1232, 2015.
[2] 3GPP, "Service requirements for the 5G system; Stage 1," Technical Specification (TS) 22.261, 3GPP, 09 2017. Version 15.2.0.
[3] J. Martins and et al., "Clickos and the art of network function virtualization," in *Proceedings of the 11th USENIX NSDI*, pp. 459–473, USENIX Association, 2014.
[4] Y. C. Hu and et al, "Mobile edge computinga key technology towards 5g," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015.
[5] 3GPP, "System Architecture for the 5G System; Stage 2," Technical Specification (TS) 23.501, 3GPP, 12 2017. Version 2.0.1.
[6] 3GPP, "Procedures for the 5G System; Stage 2," Technical Specification (TS) 23.502, 3GPP, 12 2017. Version 2.0.0.
[7] M. Onus, A. Richa, and C. Scheideler, "Linearization: Locally self-stabilizing sorting in graphs," in *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, pp. 99–108, Society for Industrial and Applied Mathematics, 2007.
[8] M. Caesar and et al., "Virtual ring routing: Network routing inspired by dhts," in *SIGCOMM '06*, pp. 351–362, ACM, 2006.
[9] I. Stoica and et al., "Chord: A scalable peer-to-peer lookup service for internet applications," in *SIGCOMM'01*, pp. 149–160, ACM, 2001.
[10] B. Pfaff and et al, "The design and implementation of open vswitch.," in *NSDI*, pp. 117–130, 2015.
[11] W. Wojdak, "Rapid spanning tree protocol: A new solution from an old technology," *Reprinted from CompactPCI Systems*, 2003.
[12] A. Myers, E. Ng, and H. Zhang, "Rethinking the service model: Scaling ethernet to a million nodes," in *Proc. HotNets*, 2004.
[13] I. Hwang, B. Song, and S. S. Soliman, "A holistic view on hyper-dense heterogeneous and small cell networks," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 20–27, 2013.
[14] J. Moy, "Open shortest path first routing protocol (version 2)," tech. rep., RFC 1583, Proteon, March, 1994.
[15] Y. Rekhter and T. Li, "A border gateway protocol 4 (bgp-4)," 1995.
[16] M. H. Behringer and et al., "An Autonomic Control Plane," Internet-Draft draft-ietf-anima-autonomic-control-plane-03, IETF, July 2016. Work in Progress.
[17] S. Sharma and et al., "In-band control, queuing, and failure recovery functionalities for openflow," *IEEE Network*, vol. 30, no. 1, pp. 106–112, 2016.
[18] T. Omizo and et al., "Resilientflow: Deployments of distributed control channel maintenance modules to recover sdn from unexpected failures," *IEICE TRANSACTIONS on Communications*, vol. 99, no. 5, pp. 1041–1053, 2016.