Coordinated Control and Spectrum Management for 5G Heterogeneous Radio Access Networks

Grant Agreement No.: 671639
Call: H2020-ICT-2014-2

Deliverable D5.1
Draft specification and implementation of the algorithm from programmable Radio Access Networks

Version:	1.0
Due date:	30.07.2016
Delivered date:	16.08.2016
Dissemination level:	PU

The project is co-funded by

Authors

Roberto Riggio (CREATE-NET, Editor)
Fang-Chun Guo (EICT)
Kostas Katsalis (EURECOM)
Shaoteng Liu, Dejan Kostic (SICS)
Robert Norgren Erenborg (SICS)
Tao Chen (VTT)
Antonio Cipriano (TCS)
George Agapiu (OTE)
Brian Meads, Dimitri Marandin (CMA)

Coordinator

Dr. Tao Chen

VTT Technical Research Centre of Finland Ltd

Tietotie 3

02150, Espoo

Finland

Email: tao.chen@vtt.fi

Disclaimer

Acknowledgement

This report is funded under the EC H2020 5G-PPP project COHERENT, Grant Agreement No. 671639.

Version history

Version	Date	Remarks
1.0	4/4/2016	ToC created.
1.1	11/5/2016	ToC Finalized
1.2	25/6/2016	Initial introduction draft
2.0	4/7/2016	Integrated Contributions from SICS, CNET, and EUR
2.1	13/7/2016	Revised section 2 in order to make it more compliant with the COHERENT Architecture
2.2	14/7/2016	Integrated contributions from VTT
2.3	15/7/2016	Revised section 2
3.0	18/7/2016	Integrated contributions from CMA, integrated revision of section 2 by SICS
4.0	26/7/2016	First complete draft ready for review
5.0	3/8/2016	Integrated reviews from CMA, OTE, and EICT
6.0	16/8/2016	Integrated reviews from EURECOM and VTT

Executive summary

In this deliverable we report the preliminary results on algorithms for programmable Radio Access Networks (RAN). The scope of the WP5 encompasses RAN sharing, traffic engineering, and network resiliency. The results presented in this document build upon the radio and MAC abstractions devised in WP3 and WP4, and the preliminary COHERENT C3 Implementation and SDK provided by WP2.

More specifically we report on:

1. Implication of the CAP (Consistency, Availability, and Partition tolerance) theorem on the mobile network control plane (COHERENT Objective 4).
2. Network graph based mobility management and energy optimization in dense small cell networks (COHERENT Objective 5).
3. RAN sharing aspects, including performance isolation and revenue maximization in multi-network operator sharing environments with bandwidth and latency SLAs (COHERENT Objective 6).

The current outcomes show promising results as far as the benefits expected from the COHERENT control and coordination abstractions and the SDK are concerned. With regard to business aspects, an initial analysis has been provided for selected project use cases, namely Network Function Virtualization (NFV) and RAN Sharing.

Moreover, from the scientific dissemination standpoint the results are encouraging. In particular, within this reporting period a total of five papers (four conferences and one journal) have been published [Chen2016, Riggio2015A, Riggio2015B, Riggio2016, Katsalis2016].

The next steps for the validation of the COHERENT abstractions and of the SDK call for further experimentation within individual research areas but also cross-research areas in order to better serve the different use cases, in which the COHERENT solutions belonging to different research domains need to operate simultaneously, e.g. traffic engineering and spectrum management. This larger experimentation will reveal the expected results and will also demonstrate the enhancements provided by COHERENT and the SDK.

Note that not all the novel solutions presented in this document have been or will be actually implemented and tested in the integrated demonstrator. On the contrary, we plan to prioritize the solutions to be implemented and validated on the testbed according to *both* their significance in terms of market impact and their suitability to be supported in an effective and efficient manner by the SDK and by the available prototypes. This is done in order to maximize the technical impact of the project given the amount of resources available.

Table of contents

List of abbreviations

2G	2nd Generation Mobile Networks
3GPP	Third Generation Partnership Project
4G	4th Generation Mobile Networks
5G	5th Generation Mobile Networks
5G PPP	5G Infrastructure Public Private Partnership
API	Application Programming Interface
BS	Base Station
C3	Centralized Controller and Coordinator
CAPEX	Capital Expenditure
CEPT	European Conference of Postal and Telecommunications Administrations
CN	Core Network
CNV	Centralized Network View
CoMP	Coordinated Multipoint
D2D	Device-to-Device
eNB	evolved Node B
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
GW	GateWay
GWCN	Gateway Core Network
IaaS	Infrastructure as a Service
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
LTE	Long-Term Evolution
LTE-A	Long-Term Evolution Advanced
MAC	Media Access Control
MEC	Mobile Edge Computing
MIMO	Multiple-Input Multiple-Output
MOCN	Multi-operator Core Network
MTC	Machine-Type Communication
MNO	Mobile Network Operator
MVNO	Mobile Virtual Network Operator
NaaS	Network as a Service
NFV	Network Function Virtualization
NGMN	Next Generation Mobile Networks Alliance
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	Orthogonal Frequency Division Multiple Access

OPEX	Operational Expenditure
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
RAT	Radio Access Technology
RRM	Radio Resource Management
RT	Radio Transceiver
SA	Service and System Aspects
SC	Small Cells
SDK	Software Development Kit
SDN	Software Defined Network
SD-RAN	Software-Defined Radio Access Network
SLA	Service Level Agreement
TR	Technical Report
TN	Transport Node
UE	User Equipment
VNF	Virtual Network Function
WiFi	Wireless Fidelity
WWLAN	Wireless Local Area Network
WP	Work Package, Working Party

1. Introduction

Fifth generation mobile networks are expected to deliver dramatic improvements along several dimensions including, but not limited to, capacity, coverage, latency, and service deployment time. While significant boosts to the system performances are expected to be delivered by new advanced physical layer technologies, such as CoMP and MIMO, a key role will also be played by innovative control and coordination schemes and by network programmability.

In particular, the envisioned widespread deployment of dense and heterogeneous radio access networks is expected to raise several research challenges as summarized in [Andrews2012]. Moreover, the massive CAPEX investments which are necessary to be considered in order to provide such capillary network infrastructure changes, will make it even more challenging for Mobile Network Operators, (MNOs), to deploy 5G networks. In this context, the Network–as–a–Service, (NaaS), business model shall allow MNOs to tap into new revenue streams by further abstracting the physical network into service specific slices possibly operated by different Mobile Virtual Network Operators (MVNOs).

The envisioned vertical applications range from high–definition video delivery to machine–to–machine and from e-health to public safety. Such use cases clearly impose different requirements on the network in terms of bandwidth, latency and resiliency. As a result, in order to cope with the diverse range of requirements that arise for such use cases, MNOs will further rely on virtualized resources and on dynamic service orchestration.

Within WP5, the Software Development Kit (SDK) developed in WP2 and the radio and MAC abstractions devised in WP3 and WP4 will be used to design and implement novel solutions to be integrated into the COHERENT demonstrator. Notice, however, that not all the novel solutions originating in WP5 will be actually implemented and tested in the integrated demonstrator, on the contrary we will prioritize the solutions to be implemented and validated on the testbed according to *both* their significance in terms of market impact and their suitability to be supported in an effective and efficient manner by the SDK and by the available prototypes. The selected solutions will then be passed as software packages to WP6 for integration and testing in the final COHERENT demonstrator.

In this deliverable we report on some preliminary results on three key aspects of programmable Radio Access Networks, namely network resiliency, traffic engineering, and RAN sharing. More specifically we report on:

4. Implication of the CAP (Consistency, Availability, and Partition tolerance) theorem on the mobile network control plane (COHERENT Objective 4).
5. Network graph based mobility management and energy optimization in dense small cell networks (COHERENT Objective 5).
6. RAN sharing aspects, including performance isolation and revenue maximization in multi-network operator sharing environments with bandwidth and latency SLAs (COHERENT Objective 6).

1.1 Outcomes

The work achieved in the framework of WP5 during the first phase of the project contemplates several preliminary results covering different individual research areas. The current outcomes show promising results as far as the benefits expected from the COHERENT control and coordination abstractions and the SDK are concerned. With regard to business aspects, an initial analysis has been provided for some of the project use cases, namely Network Function Virtualization (NFV) and RAN Sharing.

Moreover, also from the scientific dissemination standpoint the results are encouraging. In particular, within this reporting period a total of five papers (four conferences and one journal) have been published [Chen2016, Riggio2015A, Riggio2015B, Riggio2016, Katsalis2016].

The next steps for the validation of the COHERENT abstractions and of the SDK imply further experimentation within individual research areas but also cross-research areas in order to better serve

the different use cases, in which the COHERENT solutions belonging to different research domains need to operate simultaneously, e.g. traffic engineering and spectrum management. This larger experimentation will reveal the expected results and will also demonstrate the enhancements provided by COHERENT and the SDK.

1.2 Organization of this deliverable

This deliverable is organized as follows.

- In Section 2 we tackle the control plane resiliency problem by studying the implication of the CAP theorem on centrally controlled heterogeneous wireless networks. We also introduce a system architecture for resilient network nodes and we report on some preliminary results leveraging on a real-world prototype.

- In Section 3 we study the challenges on performance isolation in NFV environments. The section reports on a novel Virtual Network Function (VNF) placement problem formulation and on its implementation and evaluation over a real-world prototype.

- In Section 4 we present a generic framework developed for the multi-operator/multi-service provider case operating over the virtualized Mobile Edge Computing (MEC)-enabled LTE network. The framework is designed in such a way as to be applied to different kind of RAN/infrastructure sharing systems.

- In Section 5 we build upon the network graph in order to globally optimize the energy consumption of a heterogeneous mobile network.

- Finally, Section 6 provides a consolidated overview of results for this first phase and outlines the future work as far as validation and business impact analysis are concerned.

2. CAP trade-offs in programmable RANs

During the last decade, *Radio Access Networks* (**RANs**) have seen a significant increase in terms of both number of Wireless Clients and usage [Gudipati2013]. This tendency leads to a significant increase in cellular data usage and has called for a densification and optimization of the current RAN deployment in order to meet the user demand. Densifying the RAN deployment raises several challenges, e.g., increased interference [Chen2016, Fontes2015] and an overall increased need for inter-cell coordination. The densification of today's RANs calls for intelligent coordination between the different RAN elements. However, traditional RAN deployments are manually configured to ensure system-wide availability and consistent policy enforcement. This method has several limitations: it lacks flexibility, programmability and a global view over resource allocation.

Logically centralizing the control and coordination of the RAN is a promising approach to overcome these limitations. Logically centralized control and coordination can provide several advantages, e.g., effective load balancing and reduction of unnecessary hand-overs (e.g., ping-pong behavior), as well as interference relation mapping and mitigation. In COHERENT, the logically centralized control and coordination task resides in the C3 entity which collects the Centralized Network View (CNV) from the network nodes and exposes it to the control applications. Notice how with the term network node we address any network element in the COHERENT infrastructure plane, namely: Transport Node (TN) and Radio Transceiver (RT). It is also worth stressing that we use the term RT to address both Wi-Fi Access Points and LTE eNodeBs. More details about the COHERENT architecture are in D2.2.

In its most general form the C3 entity is composed of several C3 Instances each of them in charge of a portion of the RAN. Notice that, the task of defining how large is the geographical area under the control of a single C3 Instance as well as defining of how large is the entire geographical area under the control of a C3 Entity is outside the scope of this document. Moreover, due to its preliminary nature, in this deliverable we will account for a network configuration where the C3 Entity is composed of a single C3 Instance. We leave the generalization of the results presented in this document to multiple C3 Instances as future work. Notice how in the rest of the section we will use interchangeably the terms C3 Entity, C3 Instance, and just C3 to referent to the node executing the control and coordination functions.

As it can be imagined, the scheme of letting a set of nodes to be controlled by a single C3 Instance increases the risk for single point of failure due to the disconnection or failure of the controller. For example, during natural disasters such as an earthquake, it is possible that the wired backbone (connecting cell towers) is partitioned into one or more isolated network partitions, thus separating portions of the RAN from their C3 Instance.

In the following, we focus on solving the node failure/network partition issues in the logically centralized programmable RAN. Our work especially tackles the scenarios occurring during emergency situations, e.g., following a natural disaster. These cases can result in massive failures and creation of one or more isolated network partitions. Thus, we may face reliability issues like control node failure/partition and normal network nodes failure/partition. We present a solution to handle these reliability issues in a logically centralized programmable RANs, with an emphasis on the *Consistency, Availability and Partition tolerance* (CAP) trade-offs.

2.1 Intuition and motivation

2.1.1 Problem statement

2.1.1.1 Leader election

During catastrophic scenarios, it is very important to provide at least some basic availability. It means that the functioning radio cells should be given the ability to serve users within the partition or at least within the cell itself.

Fig. 2-1 illustrates an example of maintaining availability in case of control node failure/partition in a programmable RAN. Here, we define the ***leader*** as the node that runs the control function (i.e., the C3

Instance). We also name a node that is under the control of the leader node as a ***follower***. Both the leader and its followers form a group. Each follower can be the leader candidate and can run the control function. This is due to the fact that our goal is to tolerate N-1 node failure, thus we require that each node has the ability to boot up the control function. In the worst case, a partition only contains 1 node, and this node should have the ability to boot up the control function and offer services (perhaps in a limited fashion) to UEs. Of course, as in [Katta2015], we can also limit the number of leader candidate node in order to, for example, reduce the amount of signaling. However, in this case we cannot guarantee that the worst case partition can still functioning.

Notice how in this scenario we have two types of control channels:

1. The network control channel. This is essentially the logical Southbound Interface (SBi) between a TN or a RT node and the C3 instance. This channel is not depicted in Fig. 2-1.

2. The control network. This is between the nodes within the same group (leader and followers). We name this channel Consensus Controller East-west Interface (EWi).

Notice how such interfaces are **logical** and can be carried over the same transport network used for the data-path (in-band signaling) as well as over different transport networks (out-of-band signaling). It is also possible in principle to have three separate transport networks: data, SBi, and EWi. However, in this document we address the most likely scenario whereby a single network is used to transport both user traffic (the data plane) and other two control channels.

As Figure 2-1 shows, initially there are 6 nodes inside a RAN. One node is the leader and controls the other five. Suppose during an earthquake, the network is partitioned into two partitions. For partition 1, the leader node is unreachable. If partition 1 needs to continue to offer radio access service to UEs (user equipment), a new leader node must be elected. In addition, the newly elected leader can run the controlling entity (C3) and have enough information to make control decisions. The restored operation is depicted in Fig. 2-2. As it will be clearer later, the Consensus Controller East-west Interface (EWi)

At the distributed system level, we can view network partition and node failure as the equivalent problem. On one hand, we can treat network partition as node failure. For example, in Figure 2-1, since from the views of the two nodes in partition 1, they cannot detect the other four nodes in partition 2, they can consider the other four nodes are failed. On the other hand, if a node is failed, it can be regarded as partitioned outside the current group.

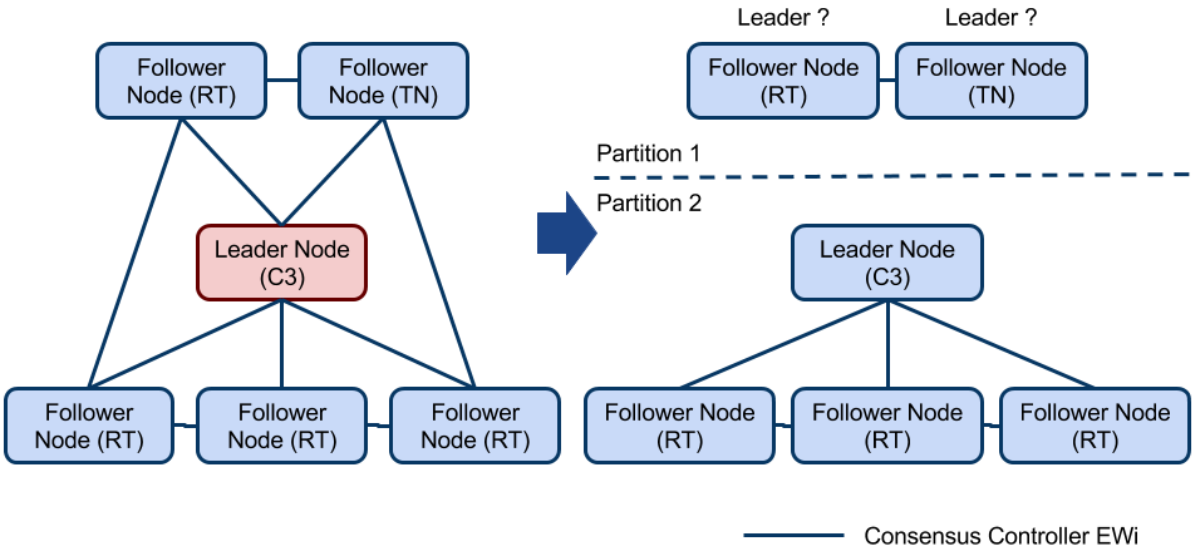


Figure 2-1 Illustration of RAN backhaul network partition.

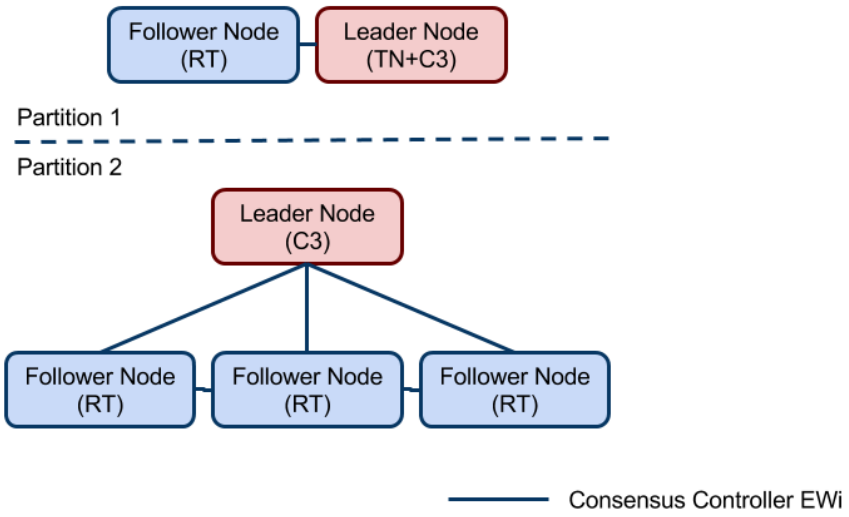


Figure 2-2 Illustration of RAN backhaul network partition with the newly elected leader.

To deal with the network partition problem in a RAN, a leader node needs to be selected for each partition to run the control task. The concept of (re)electing the controller at the network nodes in case of failure/partition might at first glance appear to be fairly straightforward, e.g., elect a suitable leader and let it resume the control operations within the partitioned network. However, closer examination reveals several obstacles. Firstly, the RAN may become unstable when the partition/failure happens. For example, during an earthquake, physical components of a wireless network may become highly unstable due to power fluctuation, causing the communication between nodes to be intermittent. Secondly, in the worst case, there is only one node in a partition, which means that our system should be able to tolerate N-1 node failures (N is the number of nodes initially forming a group). Thirdly, it is hard to know beforehand how long a partition will last. For example, partitions due to device failure inside a core area of the backhaul network may last for several hours while partitions due to a link failure may be resolved quickly if a new path can be found to re-route the traffic. Thus, the system should always be prepared to deal with recovered partitions.

2.1.1.2 Resource allocation in a partitioned network

Figure 2-3 illustrates the resource allocation problems when network partition happens. As shown in Figure 2-3, the network is partitioned into the partition A and the partition B, which are controlled by leader nodes L_A and L_B, respectively. The radio transceivers $\{RT_1, \dots, RT_6\}$ are geographically located close to each other, but divided into two partitions. Due to the broadcast nature of the wireless medium, the radio transceivers could interfere with each other. As shown in Figure 2-4, leader node L_A has no knowledge of the latest network information of partition B when the network is partitioned. Therefore, it is hard for the leader node L_A to coordinate the transmission power for the radio transceivers RT_1, RT_2 and RT_3 without the information of radio transceivers RT_4, RT_5 and RT_6 for offering better availability for the wireless clients (UEs) and less interference among radio transceivers.

We believe that the post-failure resource allocation could solve the problem. More specifically, based on the history of network statistics, a set of likely partition scenarios are pre-determined. Accordingly, the resource allocation could be proactively calculated for each partition scenario. When the partition happens, the new leader could determine the current partition scenario and then the corresponding resource allocation could be applied. Note that in this report, we focus on the work of leader election problem described in previous section. The pre-calculated resource allocation for all partition scenarios would be our future work.

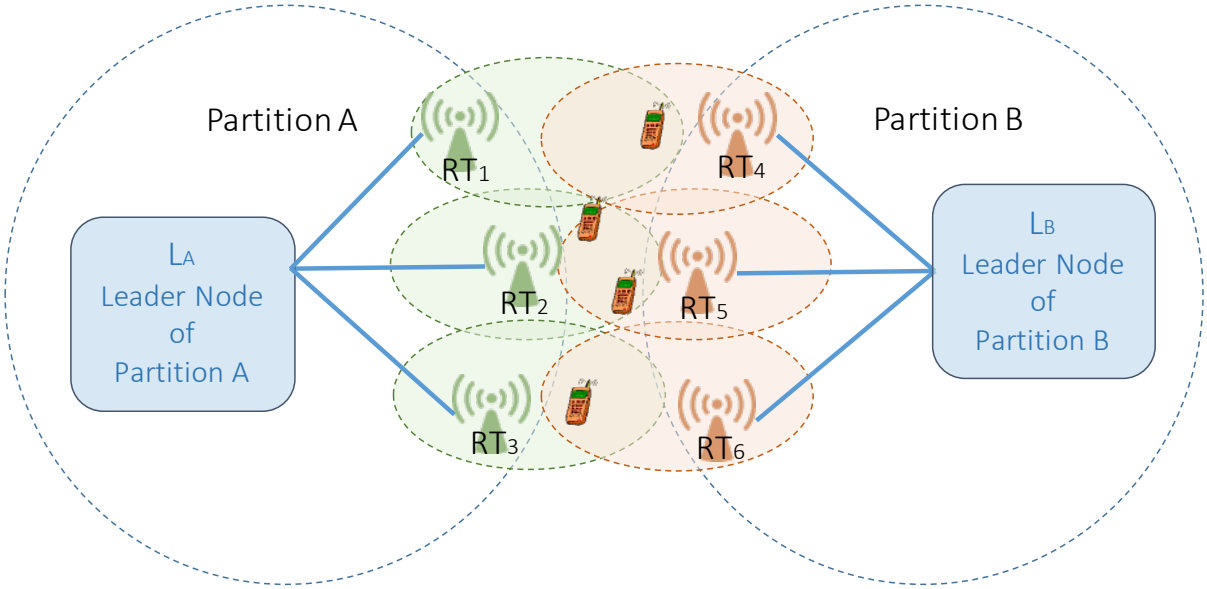


Figure 2-3 Resource allocation problem in a partitioned network.

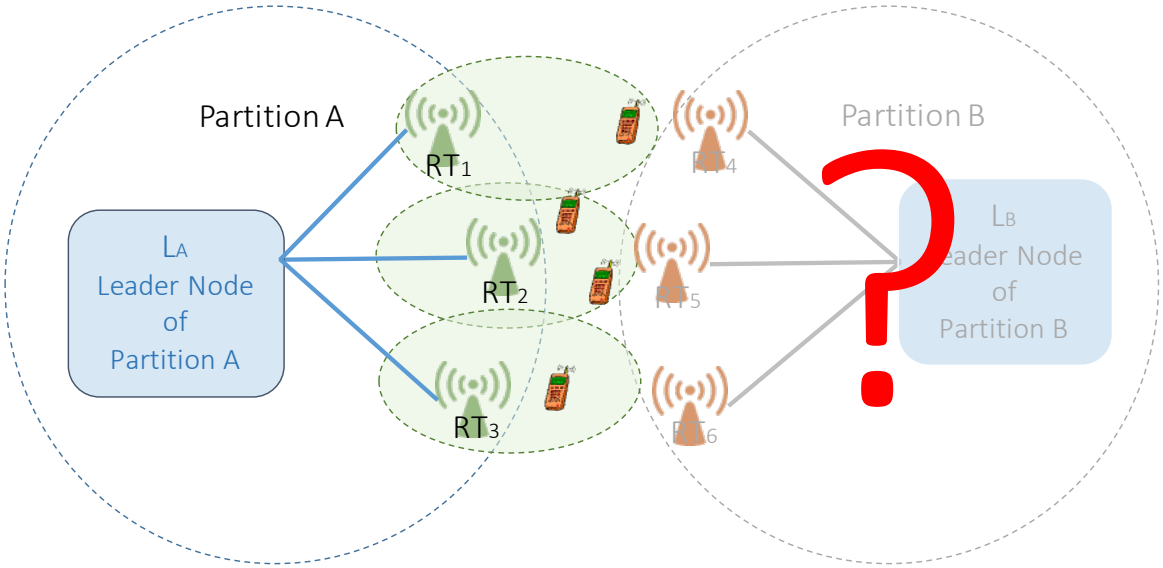


Figure 2-4 Resource allocation problem in a partitioned network: network view of partition A.

2.1.2 Motivation

Therefore, to ensure the service availability in a partition, we need to guarantee that a leader can be elected and have the ability to resume the control task. Intuitively, the system needs to satisfy the following requirements:

1. The leader node can replicate its status and the network state to its followers.

2. Each node (TNs and RTs) can have the ability to become the leader and resume the control task based on the replicated information.

3. A distributed algorithm should ensure that new leader can be elected in a new partition, and the leader can manage the group membership, e.g., it can decide when a follower can join or leave.

4. Discovering and re-unifying recovered nodes needs to be handled by the leaders.

However, one dilemma is that we cannot achieve service availability and information consistency at the same time. On the one hand, since it is impossible to synchronize each leaders' modifications on network states across partitions, the more leaders, the more risk of inconsistency. On the other hand, to guarantee the service availability, each new partition should elect a leader as soon as possible.

Actually, this dilemma has been clearly stated by CAP theory, which says that a distributed application can at most satisfy two out of Consistency, Availability and Partition Tolerance. The CAP Conjecture and its trade-offs has previously been studied in the literature [Gilbert2002, Panda2015]. In our case, since we focus on availability and partition tolerance, we have to compromise consistency to a certain extent.

Based on the CAP considerations, our aim is to design a framework that ensures service availability in programmable RAN during partitions while keeping as much consistency as possible. To this end, we design a framework that consists of three key components. Firstly, a failure detector (***FD***) can detect the failure/recovery of nodes/partitions. Secondly, a group member management and leader election (***GMLE***) algorithm can elect a leader quickly for a newly-formed partition, as well as merge leaders quickly when partitions are recovered. Thirdly, a replicated state machine (***RSM***) can replicate the leader's information to its followers. The RSM should guarantee the consistency of information within a group as much as possible.

Notice that, in our framework each node is a leader candidate (according to requirement 2). This is due to the consideration that in the worst case, a partition contains only 1 node (N-1 node failure situation). However, there is a trade-off between failure tolerate ability and the signaling cost. To tolerate more failures, we should let more nodes to be the leader candidates. Nevertheless, this increases the signaling cost between candidates. For example, failure detection cost increase as the number of leader candidates increases. As a result, we need to reduce the signaling cost as much as possible, e.g., in our current design for failure detection, each signaling message is designed very concise (just the node ID). We can further optimize the FD and GMLE algorithm to reduce the signaling cost in the future.

Moreover, we want to mention that our framework can handle both leader node and the follower node failure. In case of a follower node failure, the leader node just needs to be notified by its FD and updates the group membership. In our consideration, the most lethal failure is the leader node failure, since it would affect the service availability of a number of nodes. Thus, in the following discussion, we mainly focus on how to deal with partitions with leader node failure.

2.1.3 Challenge and related work

One prominent challenge for our framework design is the algorithm for group membership management and leader election (GMLE). Intuitively, the GMLE algorithm in our case has three requirements: 1) tolerate up to N-1 node failures, 2) work in unstable network situations, and 3) manage the group members and the merge of leaders.

A similar challenge also exists in reliable Software Defined Network (SDN) design, since leader re-election is also required in case of the control node failure. This issue has been studied in several works. In [Bote2014] and [Bote2013], each leader candidate keeps querying a data store (behind a server) to find out which node is the current leader. However, they do not consider that the server front end can also fail. [Katta2015] uses Zookeeper [Hunt2010] to elect a new leader. However, Zookeeper prioritizes C (consistency) over A (availability) in case of P (partitions). The leader election algorithm in Zookeeper selects the leader based on majority votes (called quorum), which guarantees the consistency by ensuring that there is always no more than one leader in the entire system. As a result, if a partition has less than N/2+1 nodes (we name this kind of partition as ***Minority Partition***), no leader cannot be elected and thus service is stopped (this is highly undesirable in our scenario). [Desai] builds a framework similar to ours. It uses a Paxos algorithm for replicating states and a separate eventual leader election algorithm to elect leaders. However, it does not consider partition recovery and group merging.

2.2 Background on CAP Theorem

Originally stated by Brewer [Brewer2000] and later formalized and proven by Gilbert and Lynch [Gilbert2002], the CAP theorem states that distributed application can at most satisfy two out of the following three properties: Consistency, Availability and Partition Tolerance. The definition of these three properties relies on the application and for the theorems formalized by Gilbert and Lynch and refers to a distributed database application.

Originally, consistency refers to atomic consistency, or the linearizability of database operations (e.g. read/write); meaning a read operation should always return the most recent written value. That is, provided a set G of servers all with an initial state V_0 at time t_0, if a value V_1 is written to server $A \in G$ at time t_1, this means that any server $s \in G$ should return V_1 at time $t_1 + \Delta$. Availability means that all servers $\in G$ should always accept and process incoming requests - within a finite amount of time. Partition tolerance means that the system should be able to cope with any arbitrary network partition, meaning that any message sent, at any time may get lost.

The CAP definition may vary according to the application context. For example, in [Panda2015], CAP is applied into network and communication area, consistency is then redefined as *Correctness*, meaning the ability to always satisfy a particular network policy.

In our case, we also vary the definition of Consistency and Availability a little bit from the origins. Consistency here regards the consistently replicating the state of the leader. The consistency includes that 1) Sequential consistency: updates of the state will be seen in the same order by the followers as by the leader. 2) Atomicity: updates either succeed or fail, there is no partial result. 3) Timeliness: The follower's replications of the leader's state are guaranteed to be up-to-date within a certain time bound. Besides, the availability in our concern focuses on the ability to offer radio access service to the UEs.

2.3 Proof-of-concept design

We use a proof of concept design to illustrate our ideas for the new framework. In the following parts of this section, we will firstly present an architectural overview of a node. Then, we focus on the three key components of the architecture with more detailed explanations.

2.3.1 Architectural overview of a resilient node

Here we name resilient Radio Transceiver (rRT) and resilient Transport Node (rTN) the network nodes that run our network resiliency mechanism. This essentially implies that RT and TN are enhanced with computational and possibly storage capabilities. We illustrate the architecture of rRT node in Fig. 2-5. Notice how the picture represents a situation where a generic functional split is used. However, the same considerations apply also for scenarios where no functional split is used. Similarly, the architecture of a rTN is depicted in Fig. 2-6.

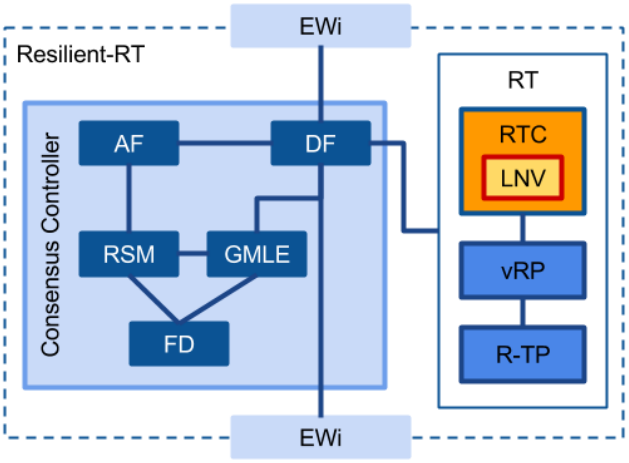


Figure 2-5 The architecture of a Resilient RT Node.

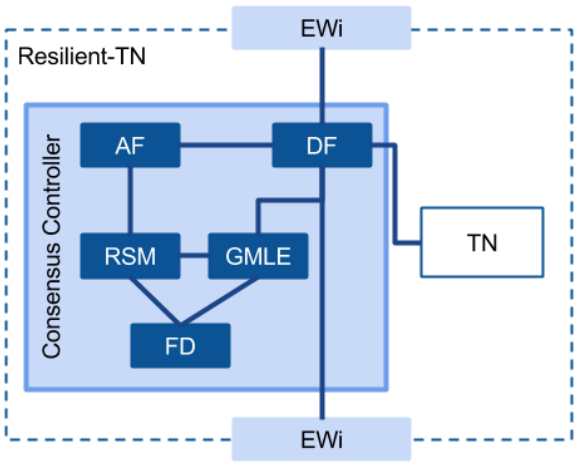


Figure 2-6 The architecture of a Resilient TN Node.

Generally speaking, each node has two modules, which are the Consensus Controller (***CC***) module and the RT module (or the TN module). The CC module also contains several sub-modules, which are called components. Their functionalities are briefly introduced as following:

RT module. The Radio Transceiver implements the physical point of attachment for the wireless terminals. An RT coincides with a WiFi AP in 802.11 network or an LTE eNodeB in a cellular network. As described in D2.2, several functional splits can be envisioned for the RT node.

The Real Time Controller (RTC) is in charge of implementing low-latency operations such as radio resource scheduling, MCS selection, and power control. According to the COHERENT Architecture, the RTC can be either embedded within an RT or decoupled and controlling several RTs at the same time. Currently, in our proof of concept design, the RT module simulates UE association behaviors. The RT will simulate UE movement for set of UEs and sends RT report to the AF of the leader node periodically, we name this period as RT report period.

CC module. The Consensus Control module is the backbone of the framework and contains a vital set of components to ensure leader election, state machine replication, and failure detection. It is composed of five components, which are: Replicated State Machine (RSM), Group membership management and leader elector (GMLE), Failure Detector (FD), Allocation Function (AF) component and the Dispatch Function (DF) component. Currently, in our proof of concept design, the RT module simulates UE association behaviors. The RT will simulate UE movement for set of UEs and sends RT report to the AF of the leader node periodically, we name this period as RT report period

RSM has the sole purpose of replicating its NoSQL-like key value store across the nodes of a group. We choose this relatively simple data store because it can capture the essence of resource allocation: association of UEs to RTs. The general functions of GMLE are leader election and group membership management. The function of FD is to detect node failures and recoveries. We will give more detailed descriptions of these three components in section 2.3.2.

The AF component takes the responsibility of making resource allocation decisions and configures the nodes that are under the leader's control. The AF achieves these goals by spawning a local instance of the C3 controller (see Fig. 2-7). Therefore, inside a group of nodes, only the leader can have its AF running while in follower node the AF is disabled (see Fig. 2-8). The function of AF is to collect information from the RTs belonging to the group members and respond with resource allocation decisions. When a decision is made by the C3 Entity or new information are received from an RT, the AF stores changes in the NoSQL database. The state of the database is then synchronized by the RSM component which replicates the leader's state with all the followers. Currently, in our proof of concept design, the C3 instance can make UE association decisions with simple strategies: e.g., when a UE sends the association request, the C3 responds with an association decision.

The function of the DF component is to deliver network status update report originated by the RT to AF of the current leader and to dispatch resource allocation decisions taken by the leader AF to the other followers RT elements.

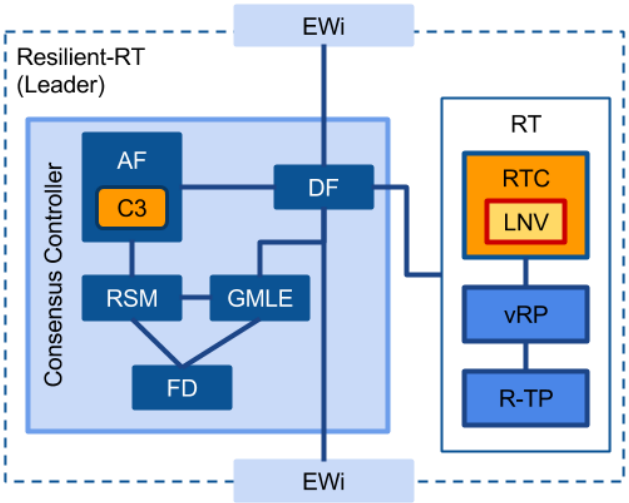


Figure 2-7 The architecture of a Resilient RT Leader Node.

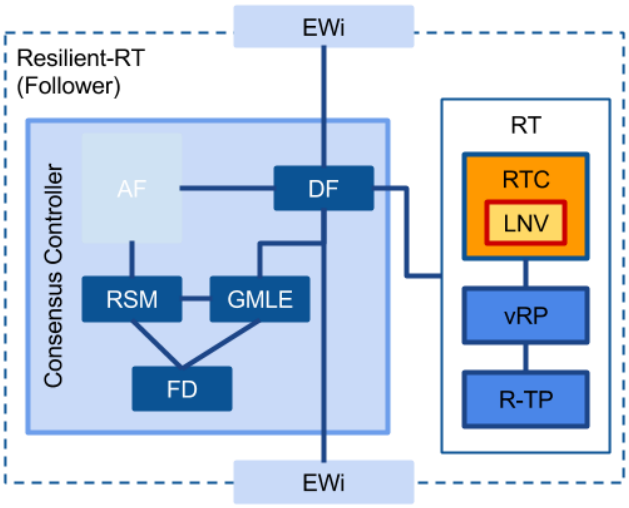


Figure 2-8 The architecture of a Resilient RT Follower Node.

As pointed out before, in our framework, each node can have three states: *leader, follower,* and *idle*. If a node is the leader, it activates its AF component and configures the DF to route the RT southbound interface to the local AF. If a node is a follower its AF component is deactivated and the DF is configured to route the RT southbound interface to the DF inside the leader node (same consideration apply also to TN nodes). When a follower loses its leader, it becomes idle. In an idle node, both the AF and the DF are not functioning. Its CC module tries to decide whether the idle node will join a leader, or become a leader as soon as possible.

2.3.2 Detailed descriptions of the key components

The key components of this framework are the FD, GMLE and RSM. In the following, we present detailed functionalities and algorithms contained in these three components.

2.3.2.1 Failure Detector

A failure detector is inside every node. The type of failure detector we use is called eventual perfect failure detector (<>P). The safety property of this failure detector states that every non-faulty node eventually suspects every faulty node. The liveness property of this failure detector states that eventually, non-faulty nodes suspect no correct node. When applied in our case, intuitively, when the network is synchronous, the failure detector can detect failed nodes and gives no false suspects. Nevertheless, when the network is asynchronous, the failure detector may suspect a correct node and cab thus generate false suspects.

The algorithm of the failure detector has been documented in [Kshemk2011, Chandra1996]. Briefly speaking, the failure detection mechanism is based on heartbeat messages: each FD component periodically (this period is referred as $t_{heartbeat}$) floods the networks with its heartbeat messages to all other nodes, and periodically checks (this period is referred as t_{check}) whether heartbeat messages (that should come from other nodes) are missing. If the heartbeat message from a certain node is missing in a check round, then this node is suspected as crashed.

With this failure detector, each node can keep updating its own view about the suspected nodes and the alive nodes. Each FD notifies the local GMLE within the same node about the failure and recovery of nodes. When the FD suspects a previously alive node to have crashed, it sends a “suspect” message to the local GMLE; when FD receives heartbeat message from a previously suspected node, it sends a “recover” message about it.

2.3.2.2 Group membership management and leader elector

The essential purpose of the GMLE is to control the transition between the *leader*, *follower* and *idle* states of a node. To this end, we design the GMLE as a Finite State Machine (FSM). Such FSM has three states. When a follower detects that its leader is giving up its leadership, or that the leader has crashed, then it transitions into the "*idle" state*. When an idle node receives invitations from some leaders, it can choose a leader to join and change its state to "*follower"* state. Nevertheless, if the idle node finds out by itself that it is the most suitable node to be the leader, it changes its state to "*leader"* state. When a leader node receives join invitations from other leaders, it means that some partitions have been recovered. Then the leader may choose to give up its leadership, join another leader, and change its state into "*follower" state*.

To control the "join" process, we intentionally introduce a sub-component in GMLE called Local Decision Maker (***LDM***) that can intervene the "join" process, as illustrated in Fig. 2-9. Any node who wants to join a leader must ask for permission from its LDM first. In our proof of concept design, the LDM runs a very simple policy on join decision making: upon receiving join invitation from a leader, it waits for a while and sees if there are more invitations from other leaders. Then, it generates a join decision, which chooses the highest-ranking node among all the inviters as its leader. In the future, we may deploy more advanced policies that choose the leader node not only based on the static ranking, but also takes network status estimation, leader node failure probability, and the effort for keeping information consistency into consideration.

The GMLE algorithm is summarized in Algorithm 1.

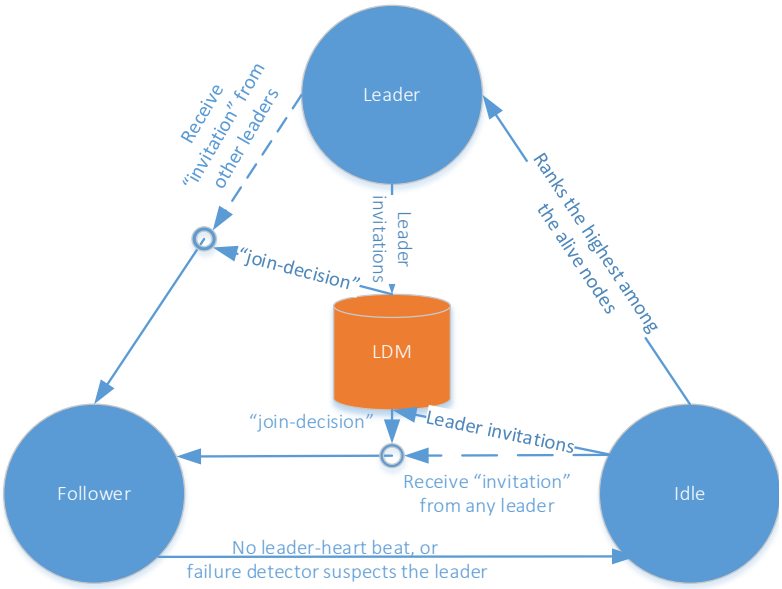


Figure 2-9 The state transition controlled by GMLE

Algorithm 1 A condensed summary of the GMLE algorithm

A condensed summary of the GMLE algorithm

Assumption:
All the nodes are ranked statically. Each node knows the rank of itself and the rank of any other node.

Definitions:
suspected nodes: **A set of nodes which are suspected to be crashed by the FD**
alive nodes**: A set of nodes which are not suspected to be crashed by the FD (*alive nodes = {all nodes – suspected nodes}*)**
followers: **A set of nodes that the leader considered under its control.**
outliers: **A set of nodes that in the alive-nodes but not in the *followers* (*outliers = {alive nodes – followers}*)**
Note: The FD keeps updating both *suspected nodes* and *alive nodes*.

Initial:
- **Check whether itself has the highest ranking node among all the nodes**
  - **If yes, change state to "leader". Set *followers = {all nodes}, outliers = {}***
  - **If no, change state to "follower". Set *followers = {}, outliers = {all nodes}***

In leader state:
Do the following tasks periodically, with a period t_{leader}
- **Send "leader-heartbeat" messages to all the nodes in *followers*.**
- **Send "invitation" messages to the nodes satisfying the following conditions: 1) have a lower ranking than itself, and 2) belong to the *outliers*.**
- **Check whether it has received "invitation" messages from other leaders.**
  - **If yes, report to its local decision maker about which nodes sent the invitations.**
- **Check whether it has received the "join-decision" message from its local decision maker.**
  - **If yes, change the state to "follower". Set *followers = {}*. Update *outliers* according to its definition. Send "ack-join" to the decided leader, which is encapsulated in the "join-decision" message.**

Response to the following messages immediately:
- **Upon receiving "ack-join" message from a node: add the node into *followers*.**
- **Upon receiving "nack-join" message from a node: ensure that the node stays in the *outliers*.**
- **Upon receiving "suspect" message from the local FD:**
  - **If the suspected node is in the *followers/outliers* set, remove it from the *followers/outliers*.**
- **Upon receiving "recover" message from the local FD:**
  - **Add the recovered node to the *outliers*.**

In follower state:
Do the following tasks periodically, with a period $t_{follower}$
- **Check whether have received "leader-heartbeat" message from its leader recently.**
  - **If yes, clear the received "leader-heartbeat" message.**
  - **If no, or if leader is in *suspected nodes*, change the state to "idle".**

Response to the following message(s) immediately:
- **Upon "invitation" message from a node: reply "nack-join" message to the node.**

In idle state:
Do the following tasks periodically, with a period t_{idle}
- **Check if it has received "invitation" messages from some nodes.**

- **If yes, report to its local decision maker about receive which nodes sent invitations.**

- **Check if it has received the "join-decision" message from its local decision maker.**
  - **If yes, change the state to "follower", and send "ack-join" to the decided leader encapsulated in the "join-decision" message.**
- **Check if itself has the highest rank among all the *alive nodes*.**
  - **If yes, change the state to "leader".**

Response to the following message(s) immediately:

- **Upon receiving "leader-heartbeat" message from a node: Reply "nack-join" message to the node.**

Regarding this algorithm, we have several points to clarify or emphasize. Firstly, the GMLE algorithm ensures that in any situation and at any time, a node can be and must be in one of the three states. Secondly, the GMLE algorithm guarantees that in any situation a node can find a leader. In the worst case, a node becomes its own leader. Thirdly, the GMLE algorithm is designed to mitigate the inconsistency due to multiple leaders. It only creates multiple leaders when the network is partitioned. Moreover, the algorithm can merge the leaders of partitions when the partitions are recovered.

The GMLE also controls the state of a node by sending state update messages to other components of the node. For example, when a node becomes as the leader, GMLE sends out a message to activate the AF component.

2.3.2.3 Replicated state machine

The main responsibility of the RSM module is to: i) ensure that the leader state (and related updates) is replicated amongst the follower nodes and ii) provide the ability to read the old state of a previous (failed) leader. The motivation for this is twofold. The first one is to ensure that any node elected as the new leader will initially have the same state as previous leader before it failed, making the leader transition smoother. The second motivation is a result of enabling the ability to assess the failure scenario and make appropriate post-failure resource allocation decisions. This means that, to determine the current failure scenario, the old leader and network (pre-failure) state is required. This is then to be compared with the current (post-failure) state.

The underlying state replication mechanisms is based on a protocol called Raft [Ongaro2014] The Raft protocol is, like Paxos, a protocol based on the notion of consensus, and is a simplified alternative to Paxos. The Raft protocol can be used as a replicated state machine that guarantees consistency. In our proof of concept RSM design, we include a third party software tool called LogCabin [Logcabin] that implements the Raft protocol.

Furthermore, although Raft protocol also provides the functionality of leader election and group membership management, it suffers from the same limitation as the Zookeeper [Hunt2010], in that it does not allow a minority partition to elect a leader. We modified the LogCabin code used in our framework so that it can be controlled and reconfigured by the GMLE. The GMLE then assigns the leader and maintains group members for LogCabin.

2.4 Case Study: Generic Network

This section presents a case study of the framework by giving intuitive examples explaining the collaboration among the components inside the framework. The study explains the cases of partition and recovery in the network topology illustrated in Fig. 2-10. The topology depicts a network with seven nodes, with node 1 as the current leader. The following parts describe the framework operations during network partition and recovery.

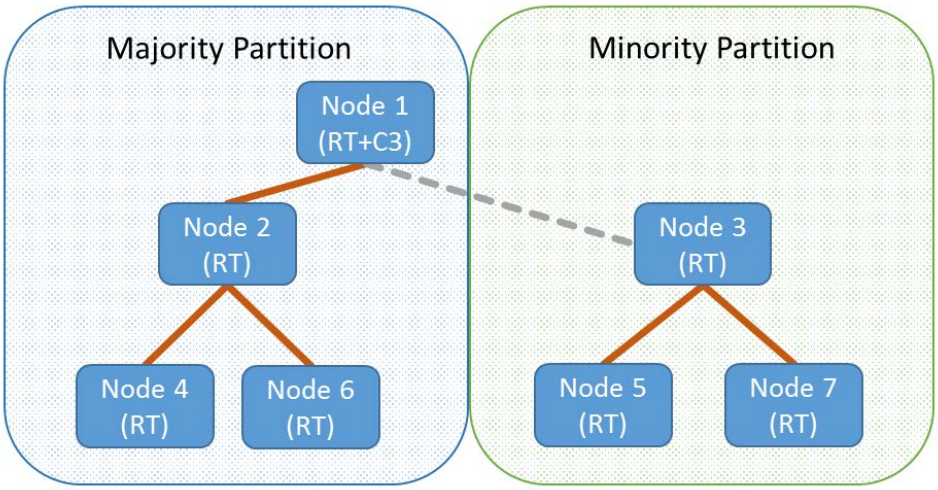


Figure 2-10 A case study.

2.4.1 Normal situation

The following describes the cooperation between components during normal situation, which is without any node failures and recoveries. In this situation, node 1 is the leader and is running the AF. Node 1 controls all RTs contained in node 1 to 7. That is, node 1 is also running a local RT instance.

The DF routes messages between the RT(s) and the AF (present in the current leader). The RT sends the so called *RT Reports* to the AF. The report contains the current internal state and wireless surroundings of the RT, e.g., associated or in-range UEs. The C3 instance in the AF makes control decision, e.g. resource allocation, based on the received reports and (when needed) sends resource allocation decisions (called commands) to the RT(s).

All decisions that change the state of the network are committed to the RSM. Each entry being committed has one of two possible states: *issued* or *confirmed*. An entry is declared *issued* once the RT has received a command. When the RT has fulfilled the command, the entry is declared *confirmed*.

2.4.2 Network partitioning

The dashed (grey) line in Fig. 2-10 illustrates the topological location of a link failure. The failure will cause the network to split, creating two isolated network partitions: one majority partition (having more than N/2 nodes), and one minority partition (having less than N/2 nodes). In the majority partition no leader election process is required (since the leader is still alive). The GMLE of the leader node thus simply notifies the AF about the updated group membership. However, in the minority partition, a leader needs to be (re-)elected. The leader (re-)election is performed in three steps, i.e., in the minority partition.

First, the FD of nodes 3, 5 and 7 suspects node 1, 2, 4 and 6 to have failed, and consequently report the failures to GMLE of nodes 3, 5 and 7, which then updates its group membership. Nodes 3, 5, 7 will all realize the lost connectivity to their leader (node 1).

Second, node 3 will claim itself to be the rightful leader (having the lowest observed ID in the group). The new leader (node 3), advertises invitations to all nodes. Idle nodes (5 and 7) then check for invitations from the new leader (node 3). Node 5 and 7 hence join the leader and transition into *Followers*, once the invitation is received and approved by their individual LDM. If a node becomes a follower, its GMLE notifies its DF about new leader. If a node becomes the leader, its GMLE notifies the DF about the new group membership. The DF in a new leader node (node 3) subsequently activates the AF. Group membership updates from GMLE are forwarded to the AF, to allow for resource allocation for the member nodes.

Third, the AF will enter the ***re-spawn*** sequence. This includes: loading the state of previous AF (from the RSM); querying/retrieving *RT Reports* from group member RTs; comparing the old state with the new RT Reports and updating the resource allocations, followed by sending these updates to the RTs. In parallel with the activation and re-spawning process of the AF, the GMLE of the new leader sends additional group membership and leader updates to the RSM. The RSM receiving these updates will reconfigure the underlying Raft server cluster (its internal group membership) to match the current leader and group membership.

2.4.3 Partition recovery

The same dashed (grey) line in Fig. 2-10 that was disabled in previous section is now restored. The restoration of the link will cause the two groups to merge. The following describes the cooperation between components to merge the two groups.

When the link is restored, the group members of each partition will detect the nodes of the other group. The FD in each node will subsequently notify the GMLE about the recovered nodes. With two network partitions, each containing a separate group membership, there will be two leaders (node 1 and node 3). Since Node 1 has the higher ranking of the two it should thus become the resulting leader of the merge operation.

Both leaders (node 1 and node 3) will then send invitations to the members of the other group. Since node 1 has the higher ranking of the two, it should thus become the resulting leader of the merge operation. So, let's consider the case of node 1. The GMLE will advertise invitations to the newly detected nodes (3, 5 and 7). Since node 5 and 7 already have a leader (node 3), they will disregard the advertisement. However, Node 3 will realize the authority of Node 1. It will join (and merge with) Node 1 and change its state to "follower".

Once the merge between node 1 and node 3 has been initiated, the GMLE of node 1 notifies the AF to initiate the merge sequence. This includes loading the state of the merged leader (node 3), merging the two states and selection of precomputed resource allocation that match the current scenario.

The followers of the group lead by Node 3 are now without a leader. Once transitioned to *idle* state they will be invited and thus join node 1.

2.5 Case Study: Mobility Management in Enterprise WLANs

In this section we will present a case study of the framework for the specific case of an Enterprise WALN. Notice how with the term Enterprise WLAN we refer in general to a network under the administrative domain of a single administrative entity and operated by a centralized network controller.

A wireless client can join a WiFi network using either a proactive or a reactive approach. In the reactive approach, wireless clients listen for Beacons messages transmitted by WiFi APs. Beacons are transmitted periodically at the lowest rate and advertise the availability of a certain AP. In proactive mode, wireless clients first send a broadcast message called Probe Request soliciting APs within decoding range to reply with a Probe Response message. At this point the wireless client is aware of the available APs in its neighborhood and can associate to any of them (prior proper authentication) using an Association Request message. In the remainder of this section we shall focus only to the proactive mode. The reason is that most APs are configured to send beacons with a very high period (tens of seconds), forcing clients to use the proactive approach for association.

2.5.1 Light Virtual Access Point Model

In Enterprise WLANs it is common to have multiple APs under the control of a single WLAN controller. Such controller, whose functions ca be played by the COHERENT C3 Entity, is responsible for various tasks ranging from basic authentication, to policing and billing. In such architecture, WiFi APs forward Probe Request messages to the Controller for further processing. The controller can use this information to build a global network hearing map which is essentially a simple network graph

describing by which AP the probe request of a given wireless client has been received. This information can be enriched also with the Receiver Signal Strength Indicator (RSSI) measured by the AP.

Starting from this hearing map a WLAN controller can selectively ask a particular AP to send the Probe Response message. Notice that, in a scenario without WLAN controller, all APs within decoding range of the Probe Request message will reply with a Probe Response leaving the task of selecting the best AP to the wireless client itself. Such an approach is suboptimal especially in very dense network in that wireless clients typically select the best AP according to RSSI measurements without considered the load of the AP itself (in terms of number of attached clients and/or traffic). On the contrary a centralized WLAN controller can effectively implement sophisticated load balancing strategies.

We can conceptually model the wireless client association process in the following way:

1. Upon receiving a certain number of Probe Request messages from a given wireless client the controlling entity (C3) generates a Light Virtual Access Point (LVAP) at a particular AP.
2. The LVAP encapsulate the state of a particular wireless client in terms of authentication and association status, encryption keys, amount of traffic exchanged, etc.
3. Only the AP hosting a client LVAP can generate messages for the client, i.e. only this AP can reply to Probe Request messages.
4. The LVAP can be moved between one AP to another.
5. Moving an LVAP from on AP to another essentially results in a handover.
6. Each AP will host as many LVAP as the number of wireless clients in communication with it.

The LVAP abstraction provides a high–level interface for wireless client's state management. The implementation of such an interface handles all the technology–dependent details such as association, authentication, handover, and resource scheduling, i.e., the LVAP abstraction captures the complexities of the IEEE 802.11 protocol stack such as handling of client associations, authentication, and handovers.

More specifically, every newly received probe request frame from a client at a AP is forwarded to the controller which may trigger the generation of a probe response frame through the creation of an LVAP at the requesting AP. The LVAP will thus become a potential candidate AP for the client to perform an association. The controller can also decide whether the network has sufficient resources left to handle the new client and might suppress the generation of the LVAP. Similarly, each AP will host as many LVAPs as the number of wireless clients that are currently under its control. Such LVAP has an ID that is specific to the newly associated client (in a WiFi network the LVAP can be thought as a Virtual AP with its own BSSID). Removing a LVAP from a AP and instantiating it on another WTP effectively results in a handover.

This model is equivalent to the normal operation of a standard WiFi network but comes with the additional property of providing us with a powerful abstraction to describe and manipulate the status of the network. Notice how the LVAP model is already part of the COHERENT SDK for WiFi Networks described in D2.3. Technical details about how to implement the LVAP model can be found in [Riggio2015B].

Let us consider the scenario depicted in Fig. 2-11. The figure depicts a simple WLAN consisting of 5 APs and 2 transport nodes (i.e. the switches). The lines interconnecting the nodes represent wired media carrying both control and data traffic (i.e. in-band signaling). The network has 7 follower nodes (in blue) and one leader node (in red).

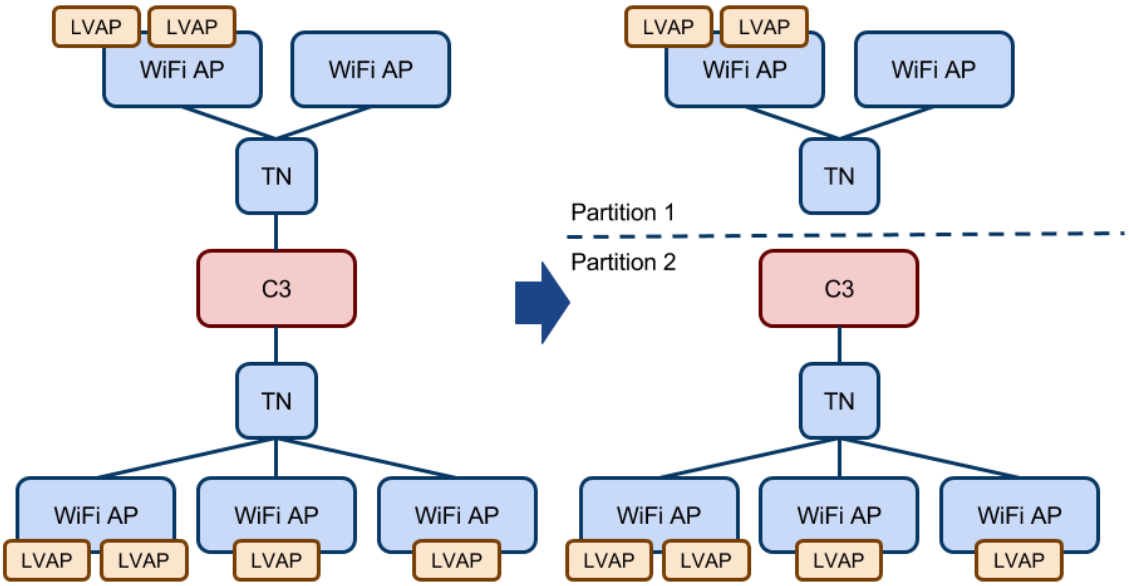


Figure 2-11. A simple failure scenario.

2.5.2 Normal Operation

During the normal network operation (i.e. before the partition), the C3 can leverage on the Network Graph in order to perform mobility management decisions. For example, the downlink and the uplink RSSI maps can be used to estimate the average Modulation and Coding Scheme (MCS) that can be used for the communication. Similarly, the empirical traffic matrix (another form of network graph) can be used jointly with the RSSI Map in order to optimize wireless client's handover. An example of RSSI Map and Traffic Map is reported in Figure 2-12.

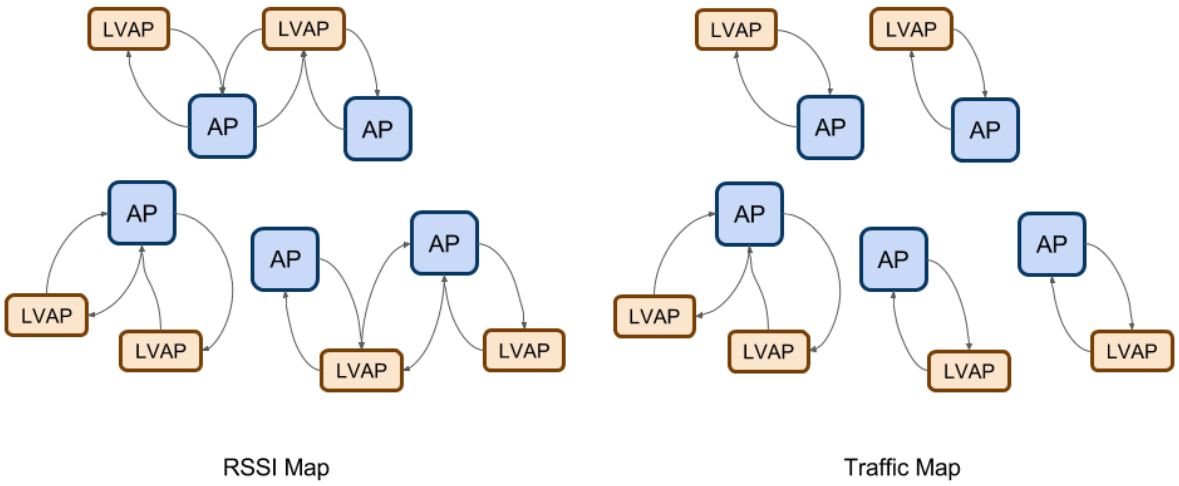


Figure 2-12. An example of RSSI and Traffic Maps.

Notice that in order to improve readability we did not plot also the edge weights. However, edges in the RSSI Map are annotated with the RSSI measured at the received. Similarly, edges in the traffic map

are annotated with the amount of traffic exchanged in the uplink and in the downlink directions (e.g. in Mb/s). Both types of network graphs can be easily stored as (sparse) matrices.

In this scenario we can easily formulate the mobility management problem as an optimization problem taking as objective function parameters such as global network throughput, airtime utilization, or energy consumption. The formulation of such an optimization problem is outside the scope of this deliverable.

2.5.3 Network Partition

Let us consider now the simple backhaul failure scenario depicted in Fig. 2-11. In this case the failure of a single wired link leaves one majority partition with 5 nodes (one of these is a leader node running the C3 controlling entity) and one minority partition with 3 nodes and without a leader. In the picture we also show the LVAPs hosted by each AP. We remind the reader that in our model each LVAP is a wireless client associated to a particular AP.

In this case the C3 entity in the majority partition can still continue to operate and to implement load-balancing strategies within its partition. However, as wireless clients start moving toward the edges of the majority partition an optimal solution to the optimization problem is not guaranteed. The reason for this is that the optimal handover opportunity maybe at an AP that is not anymore within the control of the controlling entity. The controlling entity is aware of the wireless clients that are not anymore under its control. Such clients can be removed from the state of the C3 entity. Nevertheless, it is worth noticing that, due the broadcast nature of the wireless medium, clients in the minority partition may still be within the decoding range of the APs in the majority partition. As a result, the C3 Controller in the majority partition may still be able to build a partial RSSI map.

In the minority partition a new leader will be elected. The leader will start with the same network view of the C3 controller in the majority partition. This is due to the fact that the status of the Leader is replicate at all the followers. The newly elected leader will face the same resource allocation challenges of the leader in the majority partition, i.e. lack of global network view and inability to globally optimize the network.

New wireless clients attempting to join the network will proactively send a Probe Request message in order to discover the available WiFi networks in their neighborhood. However, as a result of the network partition it is not possible for the two controlling entity to coordinate their operations. Different strategies could be used in this case. For example, the leaders could use a probabilistic approach in order to decide whatever they should reply to the new wireless client. Probabilities could be in this case linked to the size of the partition.

2.6 Evaluation

2.6.1 Test scenarios

We evaluate our proof of concept design using a series of experimental tests. Notice how in our current implementation the RT module only simulates wireless clients' association. More specifically, every 3-4 seconds the RT will simulate movement for a set of wireless clients, causing some of them to go out of range, and some to come in range (of the particular RT). All currently associated and/or in-range wireless clients are bundled into a RT report which, subsequently is sent to the AF (described below). Similarly, also C3 implementation has only rudimental functionalities, such as wireless client's association decisions based on a pre-computed allocation table.

In the setting of leader election and network partitioning there are four classes of network partitions, namely: 1) a minority partition including the (pre-failure) leader; 2) a majority partition including the leader; 3) a minority or 4) majority with no current leader (i.e., initially). In our evaluation, we mainly consider class 3) and 4), since in 1) and 2) there is no need for leader election.

To this end, we designed three test scenarios each with different partition size (number of nodes) and different partition class focus. Each test scenario also has two phases: network partition and network

recovery. In the partition phase, the link (singular) connecting the pre-failure leader will be removed causing the perception of leader failure for the now partitioned set of nodes (which are in need of a new leader). Reversely, in the recovery phase, the link will be recovered causing two partitions to merge. The three scenarios are:

Scenario 1: 16 nodes in the beginning. After the partitioning, we focus on the evaluation of a minority partition with 5 nodes and without the leader (class 3 partition).

Scenario 2: 16 nodes in the beginning. After the partitioning, we focus on the evaluation of a majority partition with 10 nodes and without the leader (class 4 partition).

Scenario 3: 16 nodes in the beginning. After the partitioning, we focus on the evaluation of a majority partition with 15 nodes and without the leader (class 4 partition).

Each scenario will from this point be referred to as the number of nodes in the considered network partition, i.e., 5 *nodes*, *10 nodes* and *15 nodes*. A node is a virtual host running the proof of concept node architecture, and will be running the RT with the capability of running the control functions (if elected leader), in addition to the GMLE, RSM and FD modules.

2.6.2 Environment and parameter settings

The nodes running the framework are emulated using a modified version of the *MiniNet* network emulator [Mininet] version called *mininet-wifi* [Fontes2015], which allows for in-band control of OpenFlow switches. Due to restrictions of mininet-wifi, the UEs are not emulated but instead simulated inside the framework nodes. The link latency between any two neighboring nodes is set to 5 ms, giving a 20ms RTT between any node inside the post-failure partition. Pre-failure RTT between the leader and any follower is 30ms. The three previously mentioned post-partition sizes (5, 10, and 15 nodes) will test both leader election and group membership merge. Thus, there are 6 test cases in total, each of which runs 100 times in our evaluations. Due to the restrictions of the mininet-wifi, the UEs are not emulated but instead simulated inside the framework nodes.

The experimental emulation is run on a *Lenovo ThinkPad T450s* with an *Intel(R) Core(TM) i7-5600U* (4-core, 64-bit, 2.6-GHz) processor and with 3*4 GB, 1.6-GHz memory. Being an emulation, the framework doesn't scale well on a single laptop, i.e., running multiple nodes (10 and beyond) will overtax the system's resources. For this reason, many internal task periods were set intentionally high to lower the load on the system. The following task periods settings have been used:

- Failure Detection: $t_{heartbeat} = 300ms,\ t_{check} = 600ms$
- GMLE: $t_{leader} = 1200ms,\ t_{follower} = t_{idle} = 1800ms,\ t_{LDM} = 300ms$
- RT Reports: 3500 – 4000 ms

In the following parts of the evaluation, we refer to these parameter settings as *settings 1*.

2.6.3 Results

During each of every test scenario the considered partition is first partitioned (from the leader) and then merged. The former requires the group membership update and a leader election. The latter requires the two groups to be merged into one.

2.6.3.1 Evaluation results on GMLE

Graphs (a), (b) and (c) in Fig. 2-13 show CDFs with 50th and 95th percentile (horizontal lines) over the delay for various key events following the partitioning event. The different delay measurements are: *Failure Detection*, the delay for all nodes in the partition to detect the failed leader; *New Leader Elected*, delay for the node with lowest ID to become leader; *Follower Join Leader* and *Leader Accepts Follower* show the delay for all nodes to join the newly elected leader and for the leader to accept these nodes as followers, respectively. In general, it takes around 0.8 seconds for failure detection, another 1.2 second for leader election and another 3 seconds for the nodes in the new partition to join the new leader.

Graphs (d), (e) and (f) in Fig. 2-13 show CDFs of the delay for various key events following the recovery of partitions. The different delay measurements are: *Recovery Detection*, delay for the nodes of the partition to recognize the node as alive; *Leader Merge* delay for the post-failure partition leader to merge with the pre-failure leader; *Join New Leader*, delay for all nodes in the partition to join the pre-failure leader. In general, it takes around 1 seconds for recovery detection, 2 seconds for the leader merge and another 3 seconds for all the recovered nodes to join the new leader.

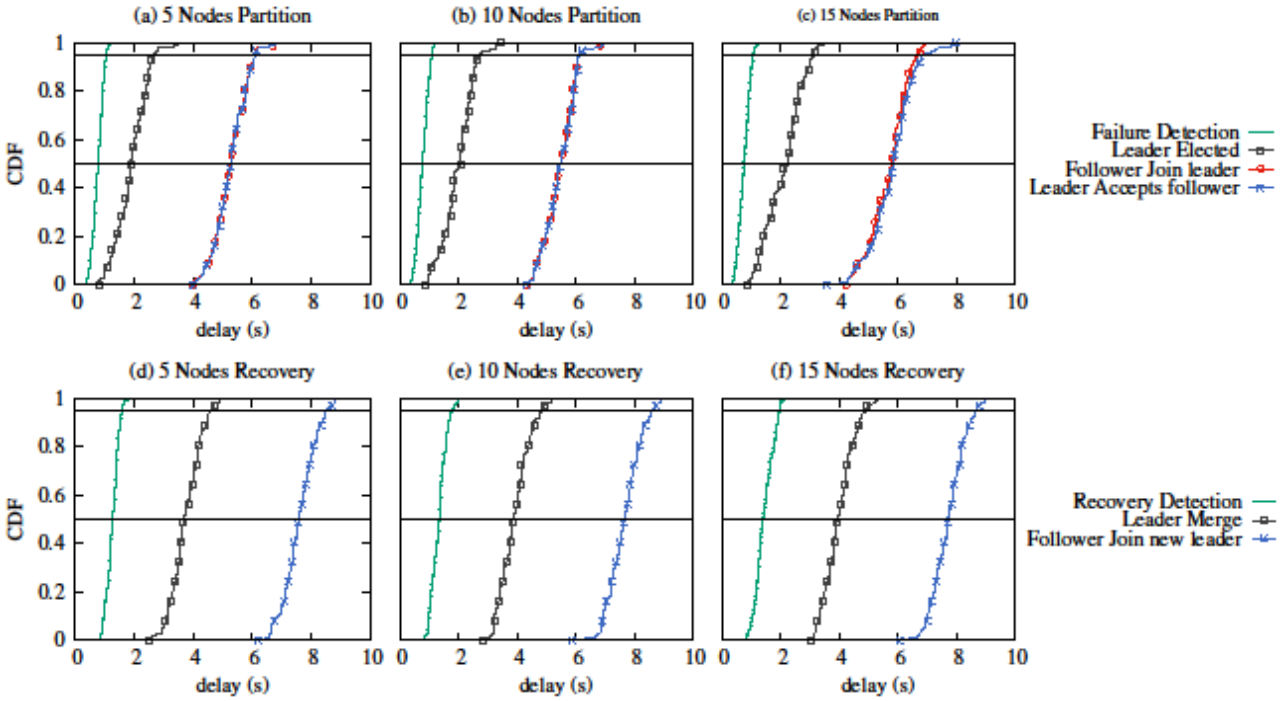


Figure 2-13. Partition and merge delays in GMLE

The delays shown by the graphs in Fig. 2-13 are closely related to the framework parameter settings, i.e., task periods in FD and GMLE. Table 2-1 aims to showcase the relationship between the observed delays and said settings. The table shows the median values for the key event delays present in the graphs of Fig. 2-13. We compare the delays for a 5 nodes partition but with two different settings. Setting 1 is the setting described in 2.6.2. Setting 2 divides all the parameters in setting 1 by three. The results show that the delay when using setting 1 is roughly 3 three times that of setting 2.

Table 2-1 The delays under different parameter settings

Test	Link Failure			Link Recovery	
	FD median	LE median	Stable median	RD median	Stable median
Settings 1	769	1298	3232	1272	7531
Settings 2	253	411	1105	810	2397

2.6.3.2 Evaluation results on RSM

The replicated state machine (RSM) uses the *LogCabin* [Logcabin] Raft server service to replicate the leader state. Fig. 2-14 illustrates the delay per entry, for read and write operations. The read and write requests considered are those related to reading or rewriting the entire state. Thus, each data point is the average delay for reading or writing multiple entries. E.g., in the 5 nodes scenario, the median read delay per entry is about 1.2 ms and the write delay per entry is around 13 ms. It can be observed that

the read delays are significantly lower than those of the write. This is due to reads are performed based on the node-local Raft server process, while write requests are replicated throughout the Raft cluster.

Furthermore, we can observe that the average (per entry) delays for both read and write operations are decreasing with the size of the partition. Increasing partition sizes (more nodes) results in larger state (number of key-value store entries). The reason for the trend of decreasing delays is the overhead introduced by calling the Raft server process coupled with the larger state sizes. Thus the results are mostly affected by the overhead of calling the process and not directly in replicating the state.

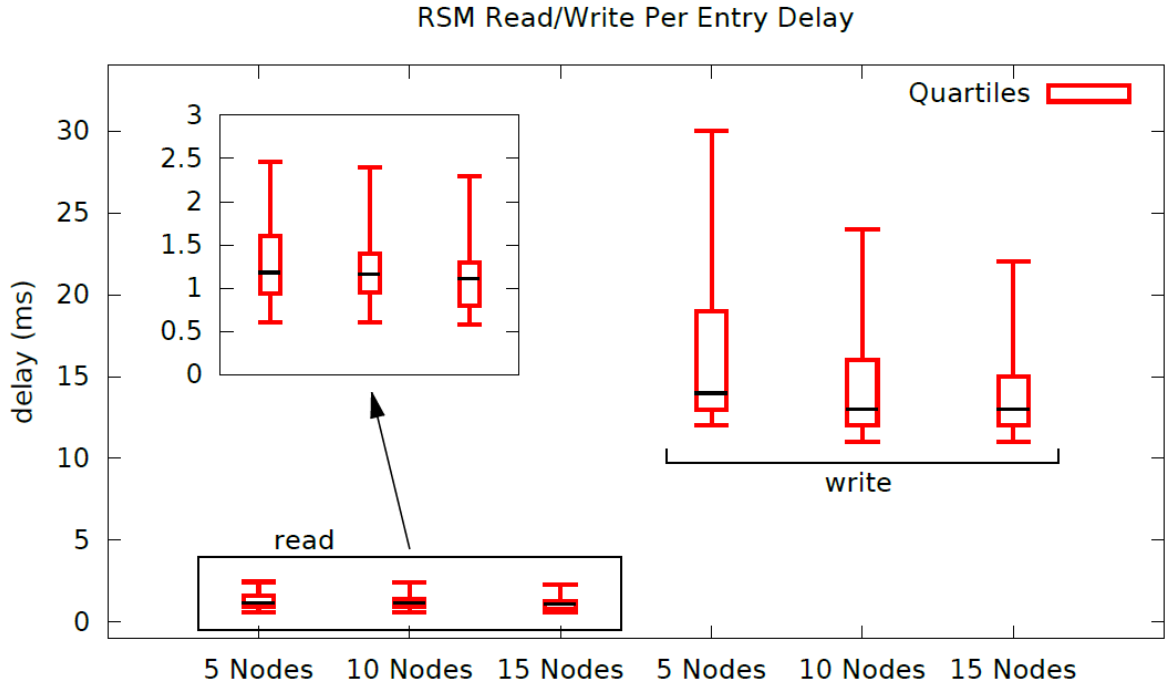


Figure 2-14. RSM read/write delay.

2.6.3.3 Evaluation results on AF

Fig. 2-15 shows the delays for different tasks in the AF (re)spawn sequence. That is, after the cluster has elected a new leader it spawns the resource allocation functions. This sequence includes: fetching old state from RSM, collecting RT reports to get the current network state and pushing the new resource allocation made by C3 to the RTs. Graph (a) shows the delay for fetching and parsing the state of the pre-failure leader. One could have expected this number to be larger, but the low delay (10-12 ms of the median) is due to the fact that the state is loaded from Raft into the RSM module as soon as the leader is detected to have failed. Since having a larger state for loading, the partition with 15 nodes appears to have a longer delay. Graph (b) shows the delay of pushing the new resource allocations to the RTs in the new group membership. The median is around 21 ms. There is a tendency of slightly increasing delays for increasing group membership sizes. Graph (d) and graph (c) reveals the bottleneck of the (re)spawn sequence. The reason for the high delays for collecting the RT reports is due to the RTs periodic nature of sending reports. To reduce complexity, the AF doesn't query the RTs for the reports, but instead waits for the periodically-sent reports. That is, the RTs are periodically sending these reports as a part of normal operation. According to setting 1, this is done approximately every 4 seconds. On average the reports should be received after roughly 2 seconds, as graph (c) indicates. In the future, we can design a RT Report request response mechanic to reduce these delays.

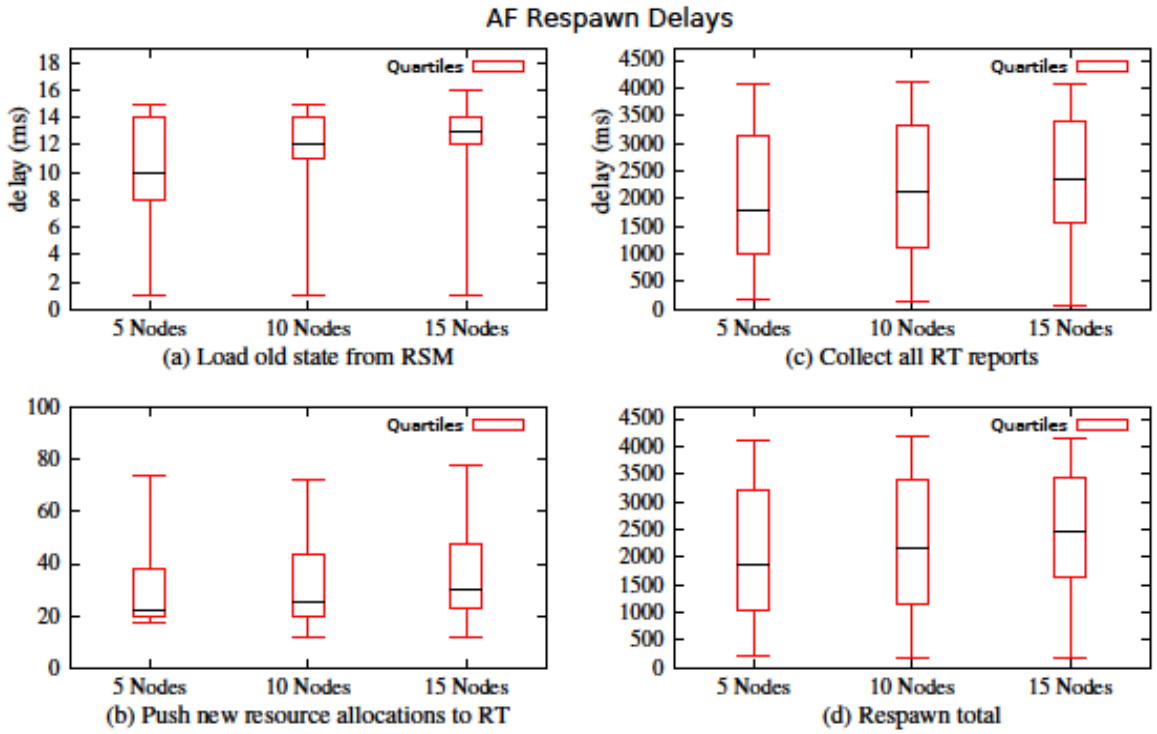


Figure 2-15 AF respawn delay decompositions.

2.6.3.4 Evaluation of the entire framework

In this section, we try to explore the total delay required for a new partition without leader to resume (correctly functioning) service. Graph (a) of Fig. 2-16 depicts the UE association service in a partition case. It shows the number of new UE association per 500 ms averaged over 100 experiments for each test scenario. These associations numbers only consider associations done by the nodes inside the considered partition. Time (0) is the time of the network partitioning happening, and in the figure there is an instant drop of served new associations. This is due to the fact that the old leader is no longer present and the new leader has not yet been elected and made ready. There will be a period during which no new UEs association requests can be served. We call this period as *service restart delay*. Under our evaluations settings 1, the service restart delay takes about 4 seconds, after which the new associations start increasing since the service is resumed by the new selected leader. However, if we scale down the parameters in settings 1, we can get such delay reduced linearly.

Graph (b) in Fig. 2-16 shows the association delays in two different phases, which are: normal (pre-failure) operation phase, and partitioned (post-failure) phase. The delay is defined as the time between the UE is detected to the time it is associated. This includes notifying the AF that ultimately makes the decision to associate the UE with the RT. One notable trend from Graph (b) is that the delay decreases in the partitioned phase, when compared with the normal phase. This is likely due to the leader moving closer to the followers in the new partition, causing the AF-RT delay reduced by approximately 10 ms.

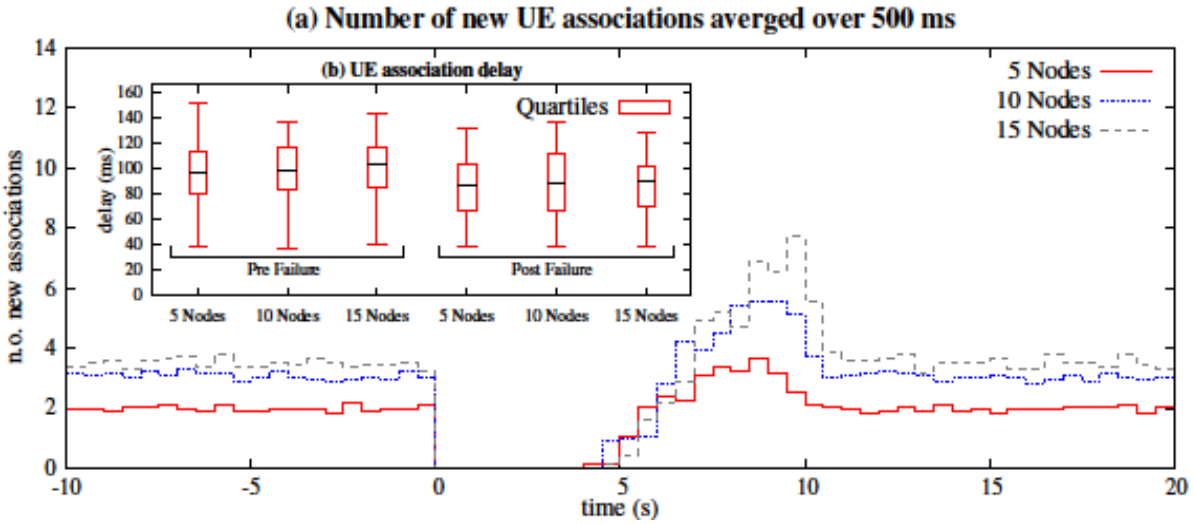


Figure 2-16 The delay to restart service in a partition without leader.

2.7 Conclusions and Future Work

We proposed a framework that can ensure service availability of a logically centralized programmable RAN in extreme situations such as network partition, which may occur during natural disasters. In this framework, a leader node runs the control entity and replicates its states and network information to other nodes in its group. Then, we designed an algorithm that can elect a new leader to assume the network control responsibility when the old leader has crashed or is outside the partition. The algorithm can also allow groups to merge when partitions are resolved. The design of the algorithm follows the CAP conjectures. The principle is to ensure service availability (A) during partitions (P) while keeping as much consistency (C) as possible.

We presented the framework by using a proof of concept design, which reveals the overall architecture and the key components to guarantee service availability. We also used a case study to describe how components in this architecture collaborate in different situations. We evaluated the design under several different parameter settings and particular partition scenarios. The evaluation results highly depend on our parameter settings of the system. Generally speaking, it may take around a few hundreds of millisecond to a few seconds to resume the service in a new partition that loses contact with the previous leader.

In the future, we can improve the framework in several aspects. Firstly, current design uses a third party software called LogCabin (implementing the Raft protocol) for state replication. However, the Raft implementation is not fast and efficient enough for our use cases. We plan to replace Raft with our own, tailored replicated state machine design in the future. Secondly, our framework can be further optimized to reduce the service restart delay in case of leader node failure. Thirdly, we might investigate and deploy more advanced partition merging polices in LDM. Fourthly, we might collaborate with other partners to apply advanced dynamic resource allocation algorithm in the C3 instance of AF and apply more sophisticated RT.

3. Performance isolation in virtualized RANs

Network Function Virtualization (NFV) sits firmly on the networking evolutionary path. By migrating network functions from dedicated devices to general purpose computing platforms, NFV can help to reduce the cost to deploy and operate large IT infrastructures. In particular, NFV is expected to play a pivotal role in mobile networks where significant cost reductions can be obtained by dynamically deploying and scaling virtual network functions (VNFs) in the core network. However, in order to achieve its full potential, NFV needs to extend its reach also to the radio access segment. Here, mobile virtual network operators (MVNOs) shall be allowed to request radio access VNFs with custom resource allocation solutions. Such a requirement raises several challenges in terms of performance isolation and resource provisioning.

In this work, we formalize the wireless VNF placement problem in the radio access network as an integer linear programming problem and we propose a VNF placement heuristic, named wireless network embedding (WiNE), to solve the problem. Moreover, we present a proof-of-concept implementation of an NFV management and orchestration framework for enterprise WLANs. The proposed architecture builds on a programmable network fabric where pure forwarding nodes are mixed with radio and packet processing capable nodes.

3.1 Motivation

NFV promises to reduce the cost to deploy and operate large networks by migrating network functions from dedicated hardware appliances to software instances running on general purpose virtualized networking and computing infrastructures. This, in time, shall improve the flexibility and the scalability of mobile networks in that the deployment of new applications and services will be quicker (software vs. hardware development life–cycles) and different network functions can share the same resources paving the way to further improvements of economies of scale. This progressive process of network softwarization is set to play a pivotal role in fifth generation mobile networks. In this context, the Network–as–a–Service, (NaaS), business model shall allow operators to tap into new revenue streams by further abstracting the physical network into service specific slices possibly operated by different MVNOs. The envisioned vertical applications range from high–definition video delivery to machine–to–machine applications. In order to cope with the diverse range of requirements that sprout for such use cases, future wireless and mobile networks will further rely on virtualized resources and on dynamic service orchestration.

Although a rich body of literature exists on VNF placement [Moens14], virtual network embedding [Fischer13], and component placement [Jennings15], most of these works focus on the problem of mapping an input virtual network request (often in the form of a VNF Forwarding Graph) onto a physical virtualized network substrate (often offering computational as well as networking resources). However, these works implicitly assume that once a VNF is mapped on a node, the virtualization layer (i.e. the hypervisor) will take care of scheduling the various VNFs ensuring both logical isolation and an efficient use of the substrate resources [Ghaznavi15]. Such an assumption does not hold anymore if radio nodes are added to the set of virtualized resources available in the substrate network (alongside computational and networking resources). In this case, in fact, the amount of resources available at each substrate radio node is a stochastic quantity depending on both channel fluctuations and end–users distribution. In this work, we investigate the VNF placement and scheduling problems in the Radio Access Network (RAN) domain. In this scenario we expect MVNOs to specify their requests in terms of a VNF Forwarding Graph. Such VNFs can include functions such as load–balancing and firewall, as well as virtual radio nodes. Moreover, in order to satisfy the diverse requirements imposed by future applications and services, MVNOs shall be allowed to deploy custom resource allocation schemes within their network slice. At the same time, the underlying system shall both enforce strict performance isolation between MVNOs and ensure efficient resource utilization across the network in spite of the non–deterministic nature of the wireless medium.

The contribution of this work is twofold:

- we formalize the VNF placement problem for radio access networks
- we propose a slice scheduling mechanism that ensures resource and performance isolation between different slices.

The proposed solutions work jointly, i.e. performance isolation is ensured if slices are accepted under the constraints imposed by our VNF placement problem formulation. The results presented in this section have been published in [Riggio2015A, Riggio2016].

We would like to notice that in this work we will use a very general network model that can be applied to the COHERENT Architecture as well as to other scenarios. The mapping between the generalized RAN network model used in this work with the COHERENT Architecture will be provided in the following sections.

3.2 Network Model

In the VNF placement problem, the input consists of Service Function Chains (SFC) composed by a variable number of VNFs, whereas the substrate network, called Network Function Virtualization Infrastructure (NFVI), provides the physical constraints in terms of bandwidth and capacity [ETSIVFV]. In this context, the term capacity is not related only to pure computational resources, such as number of CPU cores, memory, and/or storage. Instead it refers also to packet forwarding and radio processing capabilities. Before introducing the proposed, solution we need to detail specific notations for the NFVI and the SFC requests.

3.2.1 Network Function Virtualization Infrastructure Model

Let $G_{nfvi} = (N_{nfvi}, E_{nfvi})$ be an undirected graph modeling the physical network, where N_{nfvi} is the set of $n = |N_{nfvi}|$ physical nodes that compose the substrate network and E_{nfvi} is the set of edges or links. An edge $e^{nm} \in E_{nfvi}$ if and only if a point–to–point connection exists between $n \in N_{nfvi}$ and $m \in N_{nfvi}$. With respect to the physical network, links are actual wiring media, e.g., an Ethernet cable interconnecting the two nodes. Four weights, $\omega_c^s(n)$, $\omega_m^s(n)$, $\omega_s^s(n)$, $\omega_r^s(n)$, are assigned to each node $n \in N_{nfvi}$: $\omega_{c,m,s}^s(n) \in \mathrm{N}^+$ and $\omega_r^s(n) \in \mathrm{R}^+$, $0 \leq \omega_r^s(n) \leqslant 1$ representing the packet and radio processing resources available on that node. Nodes with all weights equal to 0 (zero) are assumed to be pure packet forwarding nodes. Nodes with $\omega_c^s > 0$, $\omega_m^s > 0$, $\omega_s^s > 0$, and $\omega_r^s = 0$ are assumed to be pure packet processing nodes. Finally, nodes with $\omega_c^s = \omega_m^s = \omega_s^s = 0$, and $\omega_r^s > 0$ are assumed to be pure radio access nodes. Another weight $\omega_e^s(e^{nm})$ is assigned to each link $e^{nm} \in E_{nfvi}$: $\omega_e^s(e^{nm}) \in \mathrm{N}^+$ representing the capacity of the link connecting two nodes. In order to avoid exceeding the nominal capacity of the substrate links, traffic shaping is implemented at the nodes with packet and/or radio processing capabilities. Finally, let P_{nfvi} be the set of all substrate paths and $P_{nfvi}(s,t)$ the shortest path between nodes s, t $\in N_{nfvi}$. Table 3-1 summarizes the NFVI parameters.

Variable	Description
G_{nfvi}	Substrate network graph.
N_{nfvi}	Substrate nodes in G_{nfvi}.
E_{nfvi}	Substrate links in G_{nfvi}.
$\omega_c^s(n)$	Available CPU resources at node $n \in N_{nfvi}$.
$\omega_m^s(n)$	Available memory resources at node $n \in N_{nfvi}$.
$\omega_s^s(n)$	Available storage resources at node $n \in N_{nfvi}$.
$\omega_r^s(n)$	Available radio resources at node $n \in N_{nfvi}$.
$\omega_e^s(e^{nm})$	Available resources of link $e^{nm} \in E_{nfvi}$.
$\Lambda_{c,m,s,r}(n)$	Cost for each unit of node resources, $n \in N_{nfvi}$.
$\Lambda_e(e^{nm})$	Cost for each unit of link resources, $e^{nm} \in E_{nfvi}$.

Table 3-1 Substrate network parameters.

The weights $\omega_{c,m,s}^s(n)$ associated with the packet processing nodes represent, respectively, the amount of CPU, memory, and storage resources available on that node, while the weights $\omega_r^s(n)$ are assigned

specifically to the radio access nodes and represent the normalized amount of wireless resources available at that node. Notice that with the term radio access nodes we refer to the generic nodes providing end-users terminals with wireless connectivity and that this model makes no assumptions on the type of resources that can be available at radio nodes. For example, in a 802.11 based network $\omega_r^s(n)$ could model the amount of airtime available at a certain Access Point (AP). Similarly, in an OFDMA–based network (e.g., LTE), $\omega_r^s(n)$ could be used to model the available radio resources in time and frequency at a certain eNodeB (eNB). For the sake of simplicity and without any loss of generality in this work we assume that all radio–enabled nodes initially have the same amount of resources $\omega_r^s(n) = 1, \forall n \in N_{nfvi}$.

A sample substrate network is sketched in Fig. 3-1. The network is composed by 8 nodes interconnected together. In order to improve readability link weights have been omitted. The substrate network in this example consists of 4 radio access nodes (at the bottom of the picture), and 4 switches, 2 of which supporting just basic forwarding capabilities. Notice that Radio access nodes can be either full Radio Transceivers (RT) or they can be just R-TP. In the former case the Processing nodes will serve Mobile Edge Computing (MEC) running standard VNFs, such as Firewall. In the latter case, the processing nodes may also run the Virtualizaed Radio Processing (vRP) instances associated to one or more R-TP.

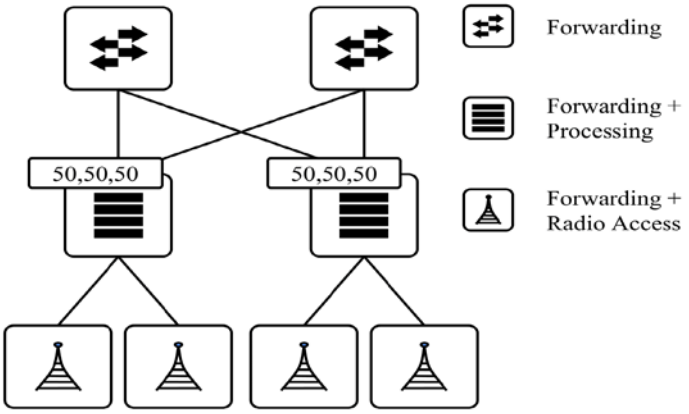


Fig 3-1. NFVI network model. The figure shows the three basic virtual resources: forwarding, packet processing, and radio access.

3.2.2 Service Function Chain Requests

Users are allowed to request SFCs as a *directed* graphs $G_{sfc} = (N_{sfc}, E_{sfc})$. Where N_{sfc} denotes the set of nodes (i.e. VNFs) and $E_{sfc} \subseteq N_{sfc} \times N_{sfc}$ denotes the set of virtual links. An edge $e^{nm} \in E_{sfc}$ if and only if the packets from VNF $n \in N_{sfc}$ must be forwarded to the VNF $m \in N_{sfc}$. Notice that as opposed to the previous NFVI model, nodes in SFC requests represent virtual network functions through which packets must undergo before leaving the network. Packet processing nodes and links in the SFC request shares the same weights as for the NFVI substrate network (ω_c^v , ω_m^v , ω_s^v , ω_e^v). On the other hand, provisioning of radio resources can be made either in terms of fraction of available radio resources (ω_r^v) or in terms of bandwidth (ω_b^v). In the former case, the users will specify the percentage of radio resources they want to be assigned to a certain node, while in the latter case the users will specify the amount of bandwidth assigned to the node. A single SFC request can mix bandwidth–based and resource–based provisioning models.

Due to their stochastic nature, available bandwidth is a time–varying quantity in wireless networks. Channel fading and distribution of end–users, can greatly influence the network performance. For example, users at the center of the cell can, in general, use more efficient modulation and coding schemes thus achieving higher throughput for a fixed amount of radio resources than users at the edges of the cell. As a result, when the bandwidth–based provisioning model is employed, also the actual channel conditions experienced by the end–users must be taken into account.

Let us call $b(n)$ the actual aggregated throughput of the virtual radio node $n \in N_{sfc}$ in the fraction of resources currently assigned the node. We can then introduce an additional parameter named reference throughput $\Omega_b^v(n) \geqslant \omega_b^v(n)$ upon which the bandwidth reservation $\omega_b^v(n)$ is enforced. We can then define the effective target bandwidth $\widetilde{\omega}_b^v(n)$ for the virtual radio node n as follows:

$$\widetilde{\omega}_b^v(n) = \begin{cases} \omega_b^v(n) & if \quad b(n) \geq \Omega_b^v(n) \\ \omega_b^v(n)\frac{b(n)}{\Omega_b^v(n)} & if\ b(n) < \Omega_b^v(n) \end{cases} \qquad (1)$$

The parameter $\Omega_b^v(n)$ represents a threshold. When the actual throughput $b(n)$ of the virtual radio node n is above the threshold, then the bandwidth reservation is respected. Conversely, when $b(n) < \Omega_b^v(n)$ the requested bandwidth $\omega_b^v(n)$ is linearly scaled down. Choosing a small value for $\Omega_b^v(n)$ means that the network can utilize more resources in order to satisfy the bandwidth requirements, which in time could result in a higher pricing. Conversely, a high value for $\Omega_b^v(n)$ means that the network will try to satisfy the bandwidth requirements only for the tenants that are making a better use of the wireless spectrum. It is worth noticing, as it will be clearer in Sec. 3.6, tenants can make a poor use of the wireless spectrum also due to custom scheduling disciplines that favor fairness at the expense of the aggregated throughput. This could be a reasonable trade–off in network slices aimed to be applied at emergency response applications where it is critical that all mobile clients get the same share of resources even if this could lead to a sub–optimal spectrum utilization.

If we define $\widetilde{\omega}_r^v(n) = \frac{\widetilde{\omega}_b^v(n)}{b(n)}$, then, in order for the SFC request to be feasible, it must hold:

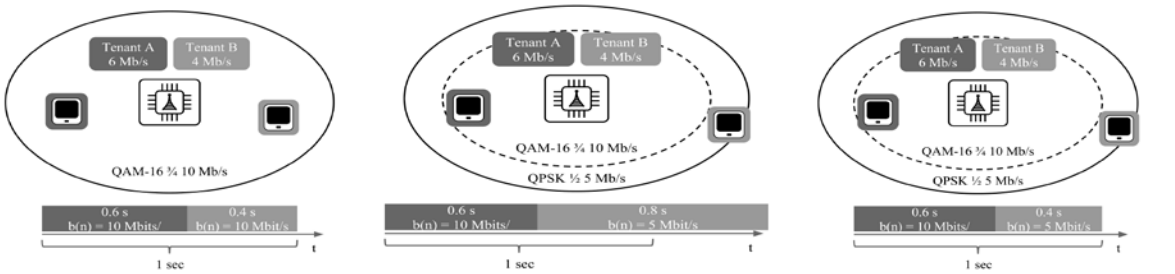
$$\sum_{n \in N_{sfc}^b} \frac{\widetilde{\omega}_b^v(n)}{b(n)} + \sum_{n \in N_{sfc}^r} \omega_r^v(n) \leq \omega_r^v(m) = 1 \qquad (2)$$

which means that the sum of the fractions of radio resources allocated to virtual nodes must be less than or equal to the resources available at the substrate node m. Table 3-2 summarizes the SFC request parameters.

Variable	Description
G_{sfc}	Service function chain graph.
N_{sfc}	Virtual nodes in G_{sfc}.
E_{sfc}	Virtual links in G_{sfc}.
$\omega_c^v(n)$	Requested CPU resources at node $n \in N_{sfc}$.
$\omega_m^v(n)$	Requested Memory resources at node $n \in N_{sfc}$.
$\omega_s^v(n)$	Requested Storage resources at node $n \in N_{sfc}$.
$\omega_r^v(n)$	Requested Radio resources at node $n \in N_{sfc}$.
$\omega_b^v(n)$	Requested Bandwidth at node $n \in N_{sfc}$.
$\Omega_b^v(n)$	Reference bandwidth at node $n \in N_{sfc}$.
$\omega_e^v(e^{nm})$	Requested resources (e.g. bandwidth) of link $e^{nm} \in E_{sfc}$.

Table 3-2. Service Function Chain Request Parameters.

For example, consider the case depicted in Fig. 3-2a. Here two tenants (A and B) requested two slices with aggregated capacity of, respectively, 6 Mb/s and 4 Mb/s, while the reference bandwidth (Ω_b^v) for both tenants has been set to 10 Mb/s. Moreover, for simplicity and without losing generality, let us assume that only one client is active in either slice. If the cell capacity $b(n)$ experienced by both clients is equal to, or higher than, 10 Mb/s it is easy to see how both slices can be accommodated by the system, thus in this case $b(n) \geqslant \Omega_b^v(n)$ and $\widetilde{\omega}_b^v(n) = \omega_b^v(n)$ for both slices.



(a) Good channel conditions **(b) Poor channel conditions, no isolation** **(c) Poor channel conditions, no isolation**

Figure 3-2. Impact of channel conditions on slice isolation. Bandwidth demands can be met when channel conditions are good for both slices (a). Conversely, bandwidth targets cannot be achieved when one of the tenants is experiencing poor channel conditions (b). Isolation can be preserved by linearly scaling down the bandwidth target for the tenant with adverse channel conditions (c).

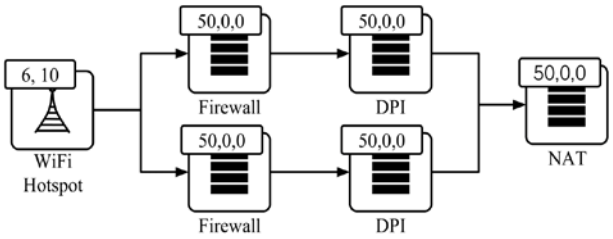
However, if the cell capacity experienced by the wireless client in the tenant *B*'s slice is reduced to 5 Mb/s, then the same request cannot be accommodated anymore. This is due to the fact that serving the same amount of data to the wireless client in the tenant *B*'s slice now requires twice the amount of radio resources it did before (see Fig 3-2b). In this case since $B(n) < \Omega_b^v(B)$, thus:

$$\tilde{\omega}_b^v(B) = \omega_b^v(B)\frac{b(B)}{\Omega_b^v(B)} = 4\frac{5}{10} = 2$$

This results in the resource allocation sketched in Fig. 3-2c which penalizes only the performance of the tenant B without affecting tenant A. It is worth noticing that this example can be easily generalized to an OFDMA system were resources are assigned across a time/frequency matrix.

A few sample SFC requests are sketched in Fig. 3-3. Notice that wireless terminals are not represented since they are outside the control of the orchestration framework. The linear SFC request in Fig. 3-3a consists of three VNFs including a WiFi hotspot, a firewall, and a load balancer. The WiFi hotspot request will also include parameters such as the name of the network and the authentication parameters (e.g. type of encryption, RADIUS server to be used, etc.) however these kind of information are purely functional and are thus omitted in this section. The SFC request in Fig. 3-3b implements a performance enhancing network service, namely duplicates filtering in dense urban scenarios. Finally, the SFC request in Fig. 3-3c accounts for an access enforcement scenario whereby multiple security related VNFs are deployed in parallel. Notice that all radio access VNFs use the bandwidth–based resource provisioning model.

(a) Radio access network sharing (Linear topology). **(b) Duplicate filtering (Branched topology).**



(c) Load-balancing (Cyclic topology).

Fig. 3-3. Sample SFC Requests. Notice that resource requests for all radio VNFs are bandwidth–based.

3.3 Problem Formulation

In this section we shall provide the optimal ILP formulation for the SFC embedding problem, while in the next section we will present a scalable heuristic. The overall objective is to compute the optimal VNF placement based on the available radio resources. The chosen objective function is:

$$min\Big(\sum_{n \in N_{nfvi}} \sum_{n' \in N_{sfc}} (\omega_c^v(n')\wedge_c(n) + \omega_m^v(n')\wedge_m(n) + \omega_s^v(n')\wedge_s(n) + \omega_r^v(n')\wedge_r(n))\Phi_n^{n'} + \sum_{e \in E_{nfvi}} \sum_{e' \in E_{sfc}} \omega_e^v(e')\wedge_e(e)\,\Phi_e^{e'}\Big)$$

where, $\Phi_n^{n'}$, $\Phi_e^{e'} \in 0, 1$ are two binary variables indicating respectively if the VNF $n' \in N_{sfc}$ has been mapped to node $n' \in N_{nfvi}$ and if the virtual link $e' \in N_{sfc}$ has been mapped to the substrate link $n' \in N_{nfvi}$. A valid solution is the one where the resources utilized by the SFC request are at most equal to the available resources on the substrate network nodes and links:

$$\sum_{n' \in N_{sfc}} \omega_c^v(n')\,\Phi_n^{n'} \leq \omega_c^s(n) \qquad \forall n \in N_{nfvi} \qquad (3)$$

$$\sum_{n' \in N_{sfc}} \omega_m^v(n')\,\Phi_n^{n'} \leq \omega_m^s(n) \qquad \forall n \in N_{nfvi} \qquad (4)$$

$$\sum_{n' \in N_{sfc}} \omega_s^v(n')\,\Phi_n^{n'} \leq \omega_s^s(n) \qquad \forall n \in N_{nfvi} \qquad (5)$$

$$\sum_{n' \in N_{sfc}} \omega_r^v(n')\,\Phi_n^{n'} \leq \omega_r^s(n) \qquad \forall n \in N_{nfvi} \qquad (6)$$

$$\sum_{n' \in N_{sfc}} \omega_e^v(e') \Phi_e^{e'} \leq \omega_e^s(e) \qquad \forall e \in E_{nfvi} \qquad (7)$$

In this work we also assume that every VNF in the SFC request shall be mapped to a different substrate node:

$$\sum_{n' \in N_{sfc}} \Phi_n^{n'} \leq 1 \qquad \forall n \in N_{nfvi} \qquad (8)$$

Every VNF in the SFC request shall be mapped only once:

$$\sum_{n \in N_{nfvi}} \Phi_n^{n'} = 1 \qquad \forall n' \in N_{sfc} \qquad (9)$$

In terms of radio resources requirements, the following constraint, deriving from (1) and (2), enforces that for every radio processing node $n \in N_{nfvi}$ a feasible request has been made:

$$\sum_{n' \in N_{sfc}^b} \frac{\omega_b^v(n')}{\Omega_b^v(n')} \Phi_n^{n'} + \sum_{n' \in N_{sfc}^r} \omega_r^v(n') \Phi_n^{n'} \leq 1 \qquad \forall n \in N_{nfvi} \qquad (10)$$

Finally, the following constraint enforces that for each link $e^{nm} \in E_{sfc}$there must be a continuous path allocated between the pair of physical nodes on top of which the VNFs $n, m \in N_{sfc}$ have been mapped.

$$\sum_{j \in N_{nfvi}, j>i} \Phi_{e^{ij}}^{e^{nm}} - \sum_{j \in N_{nfvi}, j<i} \Phi_{e^{ji}}^{e^{nm}} = \Phi_i^n - \Phi_i^m \quad \forall i \in N_{nfvi} \, \forall e^{nm} \in N_{sfc} \qquad (11)$$

3.4 Wireless Network Embedding

The ILP formulation described in the previous sections cannot be applied to realistic scenarios due to its limited scalability. For example, embedding a 6–nodes linear SFC request over a k = 8 fat–tree substrate topology can take several hours on Intel Core i7 laptop (3.0 GHz CPU, 16 Gb RAM) using the Matlab ILP solver (intlinprog). Notice also that even a k = 8 fat–tree substrate topology is rather small if compared with realistic deployments where it is common to find k = 24 fat–tree networks, making the ILP problem formulation intractable. In this section, we present a heuristic, named WiNE, that can handle similar (and more complex) requests in tens of minutes.

The proposed heuristic is composed of three steps. In the first step, for each virtual node $n \in N_{sfc}$, the heuristic computes the list of candidate substrate nodes (see Alg. 3-1). These are the substrate nodes that can support the virtual nodes in the SFC request given the input capacity constraints. In the second step, the virtual nodes $n \in N_{sfc}$ are sorted in decreasing order according to the number of candidate substrate nodes (see Alg. 3-2). In the third step, the sorted list of virtual nodes is traversed starting with the virtual node with more embedding opportunities. For each candidate substrate node $p \in N_{nfvi}$ the heuristic considers all the neighboring nodes $m \in N_{sfc}$ of the virtual node n. The heuristic, then, assigns the node n to the substrate node $m \in N_{nfvi}$ with the lowest virtual edge mapping cost (see Algorithm 3-3).

Algorithm 1. Compute list of candidate substrate nodes

```
procedure FindCandidates(N_nfvi, N_sfc)
    for n ∈ N_sfc do
        for p ∈ N_nfvi do
            If ω^s_{c,m,s,r}(p) >= ω^v_{c,m,s,r}(n) then
                n.candidates.add(p)
            end if
        end for
    end for
end procedure
```

Algorithm 3-1. Compute list of candidate substrate nodes.

Algorithm 2. Sort list of candidate substrate nodes

```
procedure SortCandidates (N_sfc)
    sort(N_sfc)
end procedure
```

Algorithm 3-2. Sort list of candidate nodes.

Algorithm 3. Nodes and links assignment

```
procedure NodeAndLinkAssignment (G_nfvi, G_sfc)
    for n ∈ N_sfc do
        for p ∈ n.candidates do
            ff p.used then
                continue
            end if
            for m ∈ n.neighbors do
                if m.mapped then
                    cost = W(e^nm, p, m.mapped)
                else
                    cost = +∞
                    for q ∈ m.candidates do
                        cost = min(cost, W(e^nm, p, q))
                    end for
                end if
                mapping_cost(p)+ = cost
            end for
        end for
        p ← argmin(mapping_cost(p))
        n.mapped ← p
        p.used ← True
        for m ∈ n.neighbors do
            if m.mapped then
                Allocate path P_nfvi(n.mapped,
                m.mapped)
            end if
        end for
    end for
end procedure
```

Algorithm 3-3. Nodes and link assignment.

We define the virtual edge mapping cost W: $E_{sfc} \times N_{nfvi} \times N_{nfvi} \rightarrow$ R between a virtual edge $e^{nm} \in E_{sfc}$ and a pair of substrate nodes $p, q \in N_{nfvi}$ as follows:

$$W(e^{nm}, p, q) = \Lambda_{c,m,s,r}(p)\omega^{v}_{c,m,s,r}(n) + \Lambda_{c,m,s,r}(q)\omega^{v}_{c,m,s,r}(m) + \sum_{e \in P_{nfvi}(p,q)} \Lambda_e(e)\,\omega^{v}_{e}(e^{nm})$$

This represents the cost of embedding the virtual edge $e^{nm} \in E_{sfc}$ over the path $P_{nfvi}(p, q)$ between the substrate nodes $p, q \in N_{nfvi}$ given that virtual nodes n,m are mapped on, respectively, the substrate nodes p, and q. Minimizing the virtual edge mapping cost for node m essentially means that

substrate nodes that are far away from node's m embedding opportunities are penalized. This results in virtual nodes in an SFC request to be placed close to each other over the substrate network, which in time means that less substrate resources are needed to support a given number of requests.

3.5 Evaluation

The goal of this section is to compare the *relative* performance of the ILP–based placement algorithm with the performance of our placement heuristic using different synthetic substrate networks and different SFC requests. In this section, we shall first describe the simulation environment and then the performance metrics. Simulations are carried out in a discrete event simulator implemented in Matlab.

3.5.1 Simulation Environment

The ILP–based placement algorithm and the proposed placement heuristic are evaluated in three different scenarios. In the first scenario, linear VNF requests, similar to the one depicted in Fig. 3-3a, are considered. In the second scenario, branched VNF requests, similar to the one depicted in Fig. 3-3b, are considered. Finally, in the third scenario, VNF requests with loops, similar to the one depicted in Fig. 3-3c, are used. The number of VNFs in each SFC request as well as the actual amount of radio, computational, memory, storage, and link resources are randomly generated for each request.

The reference substrate network is a k–ary fat–tree with k = 4, 6, 8, where leaf nodes are WiFi APs rather than servers. This results in a total of, respectively, 16, 54, and 128 WiFi APs. The computational, memory, storage, radio, and link resources for the substrate network are initially all set to 100. The cost of using each unit of node $\wedge_{c,m,s,r}(n)$ and link $\wedge_e(e)$ resources is set to 1. The number of VNFs in each SFC request depends on the SFC type. In the case of linear and branched SFC requests the number is randomly picked in the set {3, 6}, while in the case of cyclic SFC requests the number is randomly picked in the set {4, 6}. The computational, memory, and storage requirements for each SFC requests are uniformly distributed between [5, 30], while the radio and link requirements are uniformly distributed between [5, 60].

The metrics used in this study are the standard ones adopted in several other related works [Chowdhury2009], [Yu2008], [Zhu2006]. For each scenario the number of accepted requests, the average embedding cost, the average node and link utilization, and the execution time using either the ILP–based placement or the proposed heuristic are considered. In this study we assume that a fixed number of SFC requests are embedded sequentially onto the substrate network. In particular in each run, the simulator tries to embed 30 randomly generated SFC requests. Reported results are the average of 10 simulations.

3.5.2 Simulation Results:

Figures 3-4 and 3-5 shows the percentage of accepted SFC requests for different substrate networks and the average embedding cost. As expected, the ILP–based placement algorithm is more efficient than WiNE in mapping the incoming requests. This can be seen in terms of both a higher number of accepted requests as well as a lower average embedding cost. Notice however that, at least for linear SFCs, WiNE actually has a lower embedding cost. This does not mean that WiNE is more efficient than the ILP–based algorithm but rather that, by accepting a lower number of SFC requests, WiNE also utilizes, on average, less substrate resources. It is also worth noticing that the efficiency of the proposed heuristic increases with the size of the substrate network. These hints are towards the fact that WiNE may be capable of closing the gap with the ILP–based placement algorithm for realistic substrate networks.

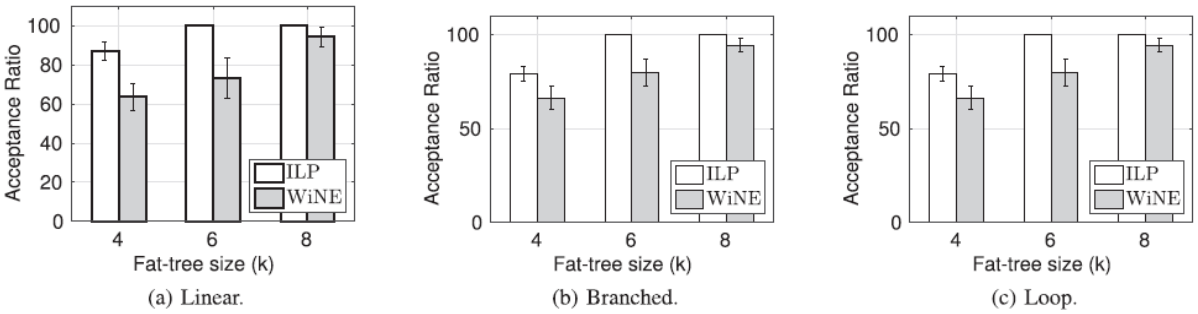


Fig. 3-4. Acceptance ratio using the ILP–based algorithm and the heuristics with different virtual and substrate topologies.

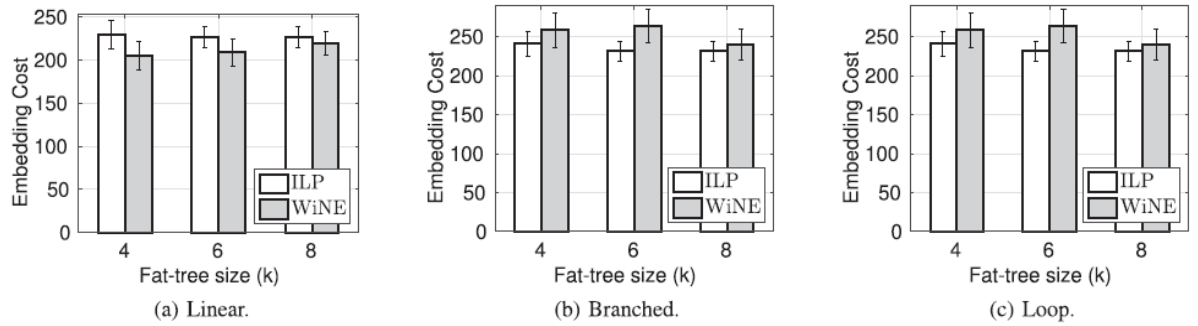


Fig. 3-5. Average embedding cost using the ILP–based algorithm and the heuristics with different virtual and substrate topologies.

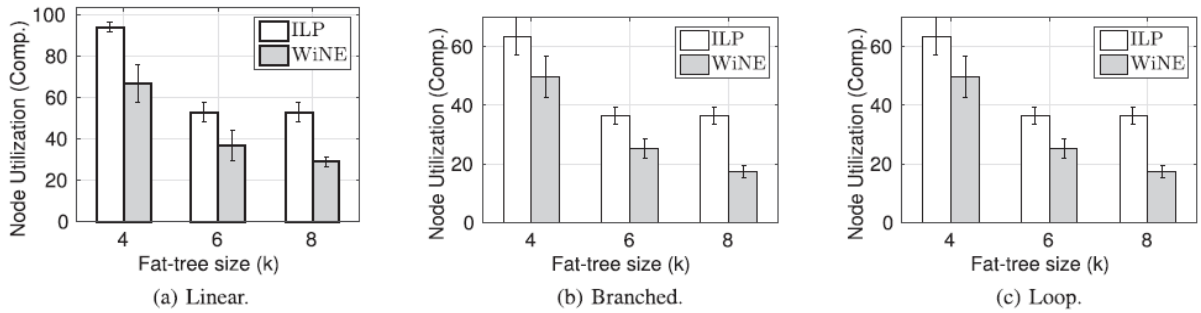


Fig. 3-6. Average computational nodes utilization using the ILP–based algorithm and the heuristics with different virtual and substrate topologies.

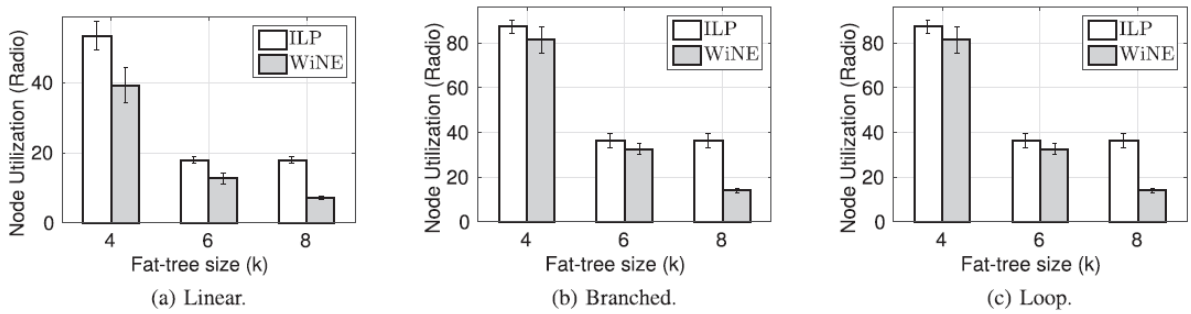


Fig. 3-7. Average radio nodes utilization using the ILP–based algorithm and the heuristics with different virtual and substrate topologies.

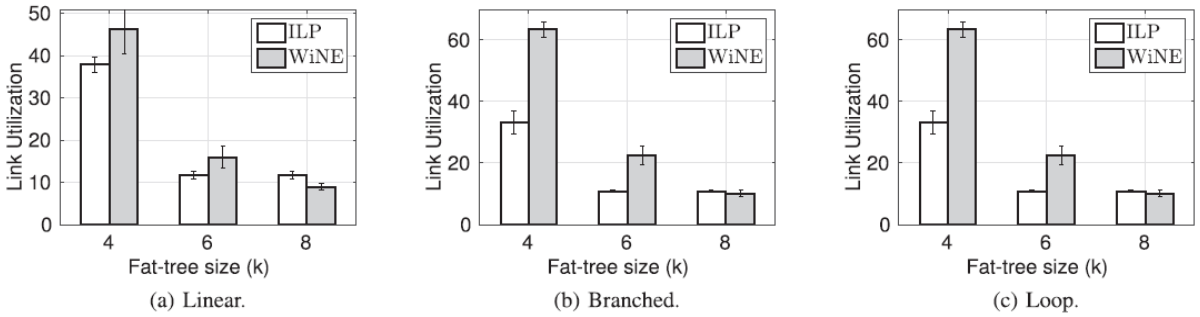


Fig. 3-8. Average link utilization using the ILP–based algorithm and the heuristics with different virtual and substrate topologies.

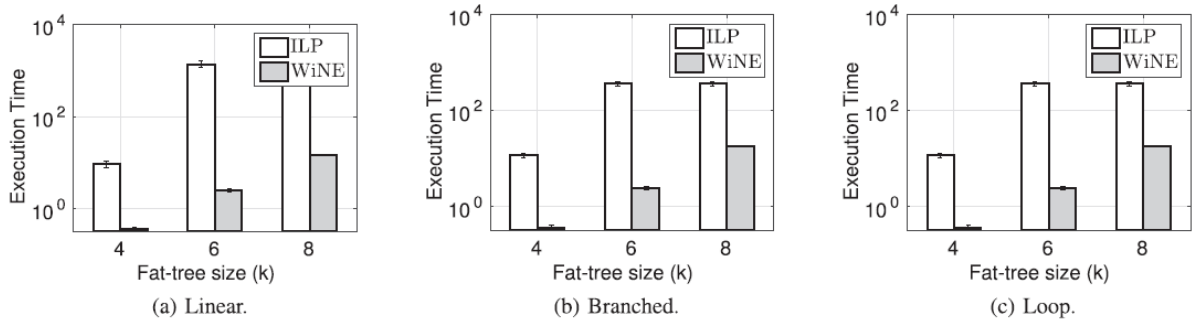


Fig. 3-9. Average execution time using the ILP–based algorithm and the heuristic with different virtual and substrate topologies.

Figures 3-6, 3-7 and 3-8 summarize the substrate resource utilization. As it can be noticed the ILP–based placement algorithm is characterized by a higher utilization ratio for both radio and computational nodes. This results in fewer substrate nodes being used to support the same number of SFC requests which in time could result in a more energy efficient operation if unused nodes can be switched off. Notice that this is further supported by Fig. 3-8 where the utilization of the substrate links is reported. As it can be seen, WiNE is characterized by a higher link utilization, which means that the proposed heuristic is less efficient in finding shorter paths between VNFs. However, it is also worth noticing that the gap between the ILP–based placement algorithm and WiNE gets smaller as the size of the substrate network increases.

Fig. 3-9 shows that the average amount of time required to embed a *single* SFC request using the ILP–based placement algorithm is significantly higher than the time required to embed the same request using WiNE. The ILP problem becomes essentially intractable for substrate networks with more than a few tens of nodes (irrespective of the number of VNFs in the request), while WiNE can effectively embed complex SFC requests on substrate networks with hundreds of nodes in a limited amount of time.

3.6 Implementation

3.6.1 Overview

We implemented the VNF placement and scheduling solution presented in this work in a proof–of–concept NFV management and orchestration framework, named *EmPOWER*. Notice that the prototype currently supports only RTs based on the 802.11 family of standards and, as a consequence, the applications described in the next section target Enterprise WLANs and Campus networks scenarios. Nevertheless, as seen in the previous sections, the provisioning model does not make any assumption about the particular link–layer technology and can be as well applied to any kind of radio access network including OFDMA networks such as LTE and LTE–Advanced.

Our proof–of–concept is loosely modeled by considering the ETSI reference NFV Architecture [ETSIVFV]. As it can be seen in Fig. 3-10, the architecture is conceptually divided into three layers. The bottom layer consists of the physical as well as the virtualized resources composing the NFVI layer. In the second layer we have the actual VNFs which are the software implementation of a particular network function capable of being executed over the NFVI. We remind the reader that in this work also radio access is treated as a VNF. Finally, in the third layer we have the Operational Support System (OSS) and the Business Support System (BSS) used by the network administrators to operate and manage their virtual networks. The Management and Orchestration plane covers the orchestration and the management of physical and/or virtual resources that support the NFVI as well as the life–cycle management of the VNFs, i.e. creation, configuration, monitoring, and destruction.

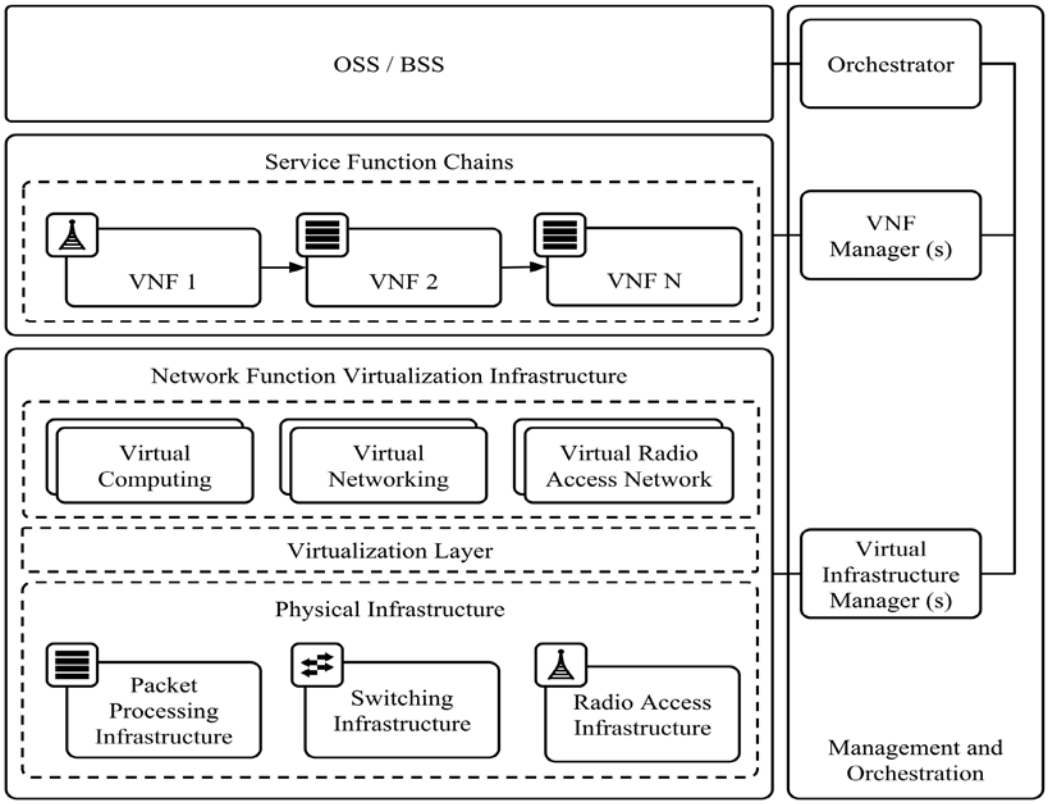


Fig. 3-10. Reference network function virtualization architecture [ETSIVFV].

3.6.2 Network Function Virtualization Infrastructure

Our architecture currently accounts for three kinds of NFVI resources, namely: basic forwarding nodes (i.e. OpenFlow switches), packet processing nodes, and radio access nodes. The latter, in addition to the features supported by the packet processing node, also embed specialized hardware in the form of one or more 802.11 Wireless Interfaces.

We name Wireless Termination Points (WTPs) the physical points of attachment in the RAN (e.g. WiFi APs or LTE eNBs) supporting virtualized radio processing capabilities. Conversely, the Click Packet Processors (CPPs) are the forwarding nodes with computational capacity. These nodes are essentially programmable switches running an embedded version of Linux and capable of performing arbitrary operations on the traffic, e.g. load–balancing, firewalling, deep packet inspection, etc.

As the name suggests the actual packet processing is performed by multiple instances of the Click Modular Router [Kohler2000]. Click allows building complex VNFs using simple and reusable components, called *elements*. Click includes over 300 elements supporting functions such as packet classification, access control and deep packet inspection. Elements can be composed in order to realize

complex VNFs. Finally, Click can be easily extended with custom processing elements making it possible to support features that are not provided by the standard elements.

Each WTP/CPP includes an OpenVSwitch instance, one or more VNFs, and one Agent. The latter is in charge of monitoring the status of each VNF as well as handling requests coming from the controller. In the current implementation the monitoring features include: number of packets/bytes transmitted and received as well as the amount of resources (cpu time, memory, storage) utilized by each VNF.

In the current prototype CPPs are built upon the Soekris 6501-70 platform consisting in single 1.6 GHz Intel Atom CPU, 2 Gbytes of SDRAM, and 12 Gigabit Ethernet Ports. CPPs run Ubuntu 15.04 Server as operating system. WTPs exploit the PCEngines ALIX (x86) embedded platform and run the OpenWRT operating system.

3.6.3 Virtual Infrastructure Managers

As Virtual Infrastructure Managers (VIMs) we use a combination of frameworks. Ryu [Ryu] is used to configure the switching fabric while for the NFVI we extended the *EmPOWER* controller presented by the authors in [Riggio2015B] in order to support also the generalized packet processing nodes. The controller supports multiple slices on top of the same physical infrastructure. A slice is a virtual network with its own set of WTPs/CPPs. The controller is responsible for the deploying the VNFs on the network devices.

3.6.4 Orchestrator

From an architectural standpoint, the VNF placement algorithm resides in the Orchestrator which is in charge of deciding whether a particular request can be accepted or if it must be refused. If a request is accepted, then the Orchestrator is in charge of mapping the request onto the substrate network, i.e., network resources must be allocated and configured on both the substrate nodes and the substrate links and the VNFs must be instantiated on the selected nodes. Notice how, our proof–of–concept does not impose a specific Orchestrator leaving network administrators free to use other currently available solutions, such as OpenBaton [OpenBaton].

3.7 Proof-of-concept evaluation

In this section we shall describe two SFCs implemented and tested over a small scale testbed deployed at CREATE–NET premises. The testbed consists of 2 OpenFlow–switches, 2 CPPs, and 20 WTPs.

3.7.1 Radio Access Network Slicing

In this use case we aim at demonstrating the performance isolation features enabled by our joint VNF placement and scheduling solution. Such use case is supported by introducing a MAC Hypervisor implementing the slice scheduling techniques described in the previous sections. It is worth noticing that the COHERENT Real Time Controller (RTC) is one possible approach for implementing such MAC Hypervisor. However, since the RTC is mainly envisioned for the LTE small-cell and since the RTC itself is still in development, the MAC Hypervisor presented in this section has been implemented using the Click Modular Router Framework [Kohler2000].

The term hypervisor is here used to refer to a network hypervisor. A network hypervisor is conceptually similar to a Virtual Machine Hypervisor. In fact, as for VM Hypervisors, a network hypervisor controls different virtual networks that are actually running on top of the same physical network. In this scenario, the network hypervisor is in charge of allocating resources to the various virtual networks or slices. An example of network hypervisor for wired networks is FlowVisor [FV2009]. An extensive survey on network hypervisors can be found in [Blenk2016] while for literature references, more related to the radio environment we refer the reader to [Katti2014, Gudipati2014].

It is worth noticing that the MAC Hypervisor presented in this section has not yet been merged with the COHERENT Architecture presented in COHERENT Deliverable D2.2. The reason being the concept of Radio Access Network slicing is still not completely explored in the literature. As a result we expect WP5 to provide WP2 with significant feedback regarding the architectural requirements for sliceable RANs. Such feedback will be integrated by WP2 and the outcomes will be reported in COHERENT Deliverable D2.2.

Fig. 3-11 sketches the internal architecture of a radio access node. Notice that not the entire WiFi MAC is virtualized. More precisely, WiFi control frames generation (e.g. ACK and RTS/CTS) as well as the actual physical layer are not virtualized due to their tight timing/computational constraints. On the other hand, each tenant can have its own Radio Resource Management (RRM) instance as well as its own management plane. In this work with RRM we refer to wireless clients scheduling and to rate adaptation, which can be defined on a per–tenant basis. For example, one tenant can use a wireless station scheduler aimed at ensuring fairness while another tenant could opt for a scheduler aimed at improving the aggregated slice bandwidth.

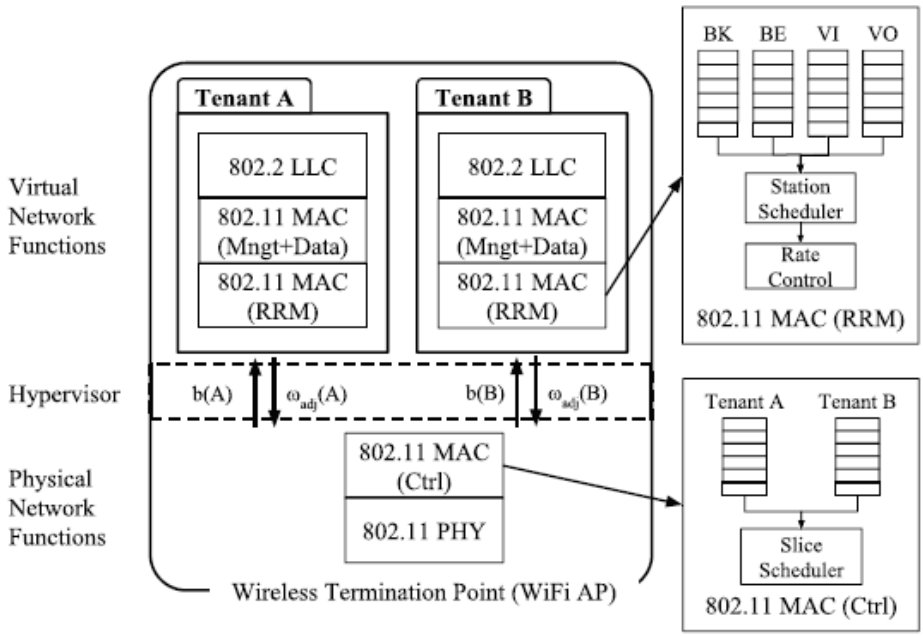


Fig. 3-11. Radio Access Network Slicing Application.

The MAC hypervisor monitors the channel utilization as well as the amount of traffic generated by each Tenant. Such information is used to compute the effective aggregated goodput of each slice (*b(n)*). It is worth stressing that both channel activity and goodput are two quantities that can be monitored by the MAC hypervisor with low overhead and without affecting the operation of the RRM policy implemented by Tenants. For example, a Tenant with wireless clients experiencing poor channel conditions will see its aggregated goodput reduced due to re–transmissions. Similarly, a Tenant implementing wireless clients scheduling policies aimed at maximizing fairness will also see a reduction in goodput if some of its wireless clients require less efficient modulation and coding schemes.

The network setup used for this set of measurements consists of two tenants A and B. The bandwidth requests coming from Tenant A and B are, respectively, 4 and 2 Mb/s, while the reference bandwidth (Ω_b^v) is 6 Mb/s for both tenants. A total of three wireless clients is active in the networks. Clients 1 and 2 belongs to Tenant A, while Client 3 belongs to Tenant B. Traffic is generated from a server sharing the same backhaul with the APs and consists of three UDP streams (one for each client). Each stream has constant inter–departure time and packet size resulting in a transmission rate of 10 Mb/s for each

stream. In order to simulate a hotspot with limited capacity the rate control algorithm used by the AP has been modified in order to always use the lowest transmission rate (6 Mb/s). Measurements have been carried in two different scenarios differentiated by the channel conditions experienced by client number 1 which is positioned in such a way as to experience channel conditions ranging from *Good* to *Poor*.

As it can be seen from Fig. 3-12a, when Client 1 is experiencing good channel conditions the proposed VNF scheduling method can satisfy the bandwidth request for both tenants (notice that Tenant A request of 4 Mb/s has been equality partitioned among the two clients). On the other hand, when Client 1 starts experiencing poor channel conditions (see Fig. 3-1b) the legacy resource provisioning mechanism allocates the same bandwidth to all the clients. This behavior, known as IEEE 802.11 performance anomaly [Heusse2003], allows a node which experiences poor channel conditions to monopolize the wireless medium lowering the performance of the whole system. Conversely, the proposed resource provisioning mechanism can meet the bandwidth reservations made by Tenant A by linearly scaling down the amount of resources allocated to Tenant B that is experiencing poor channel conditions.

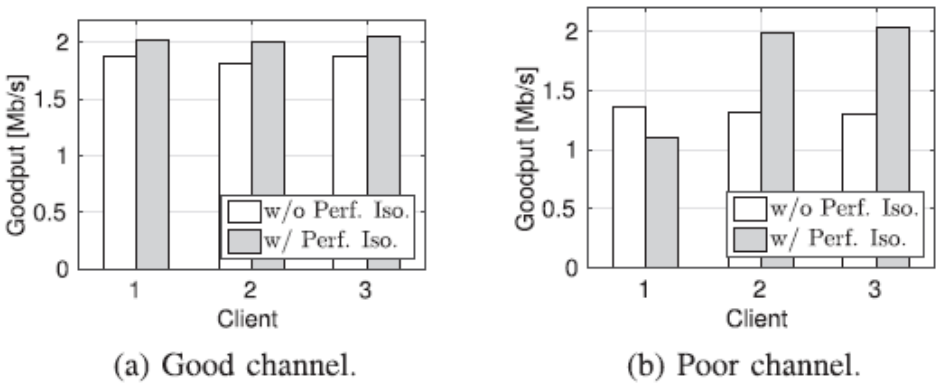


Fig. 3-12. Isolation across clients in different networks. Client 1 and 2 are in Tenant 1's network, while Client 3 is in Tenant 2's network.

3.7.2 Data/Management Plane Offloading

Wireless, and in particular mobile networks, have been so far designed around the requirements of the downlink, i.e. cell or AP selection is performed using downlink signal strength measurements. In the recent years, however, we have witnessed a mushrooming of new uplink–centric applications such as Machine Type Communications (MTC), Internet of Things (IoT), and Vehicle to Infrastructure (V2I) as well as of symmetric mobile applications. This calls for a paradigm shift where the traffic originated from a wireless client is received by one node while the traffic destined to the same client is transmitted by another node. This kind of network setup is usually referred to as uplink/downlink decoupling and, in its most general form, can consist of two possibly non overlapping sets of transmitting and receiving nodes.

The *EmPOWER* platform allows for network configurations where a wireless client can be attached to one AP in the downlink direction and to *one* or *more* APs in the uplink direction. This feature allows to exploit the broadcast nature of the wireless medium and to opportunistically receive the same transmission at multiple in–range APs. However, if not properly controlled such a feature can lead to an overload in the network core. For example, a wireless client scheduled on *N* APs in the uplink direction could increase the load on the network core by a factor of *N*. Moreover, a straightforward implementation of such a mechanism could generate a significant increase in the number of duplicate packets which in time could trigger anomalous behaviors at the transport layer.

As a result of the fact that the WiFi MAC is virtualized it is possible to offload parts of it to dedicated processing nodes as VNFs that can be then shared among several APs. In this use case we implemented

a VNF which filters out duplicate 802.11 frames based on their sequence number. Fig. 3-13 sketches a slightly more generic case were both data–plane (duplicates filtering) and management (Probe Response generation) functionalities are offloaded to two different *CPP*s. In this section however we shall evaluate only the performances of the duplicate filtering VNF.

Traffic originated from clients is received at one or more *WTP*s where it is encapsulated (802.11 over Ethernet) and then forwarded to a *CPP* where the duplicates filtering VNF is deployed. This VNF is also responsible for decapsulating the 802.11 frame and converting it into an Ethernet frame before forwarding it to its intended destination.

In order to evaluate this VNF we exploited a network setup composed of a single client and three APs. Traffic is injected from the wireless client as a single UDP stream. Packet transmission rate and payload are kept fixed at, respectively, 100 packets/s and 1472 bytes. Impairments on the link between client and APs are simulated by randomly dropping received frames with probability *PL* at all receiving APs. For all measurements a total of 6000 frames were generated. Confidence intervals were very small for all the data points and have therefore be omitted in order to improve readability. Measurements have been taken using three packet dropping probabilities, namely: 0.05 (good channel conditions), 0.2 (medium channel conditions), and 0.8 (poor channel conditions).

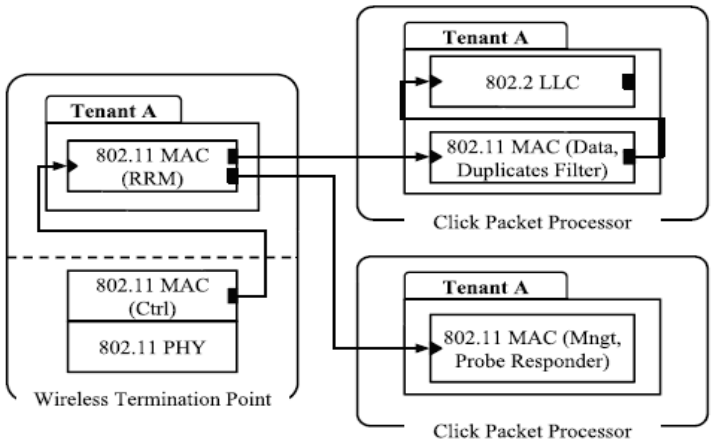
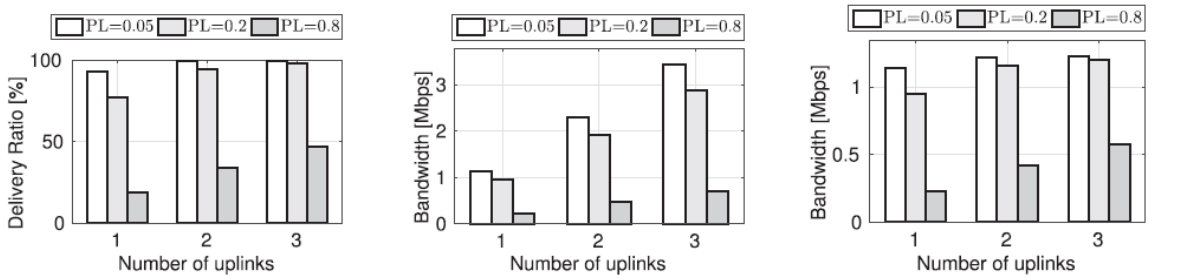


Fig. 3-13. Data/Management Plane Offloading application. Data–plane (duplicates filtering) and Management–plane (Probe Response generation) operations are offloaded to two different CPPs.



(a) Packet Loss. (b) Bandwidth utilization w/o Duplicate Filtering VNF. (c) Bandwidth utilization w/ Duplicate Filtering VNF.

Fig. 3-14. Packet loss (a) and bandwidth utilization (b and c) for a single client scheduled at multiple APs in the uplink direction. The duplicate filtering VNF can reduce the load on the backhaul by selectively dropping duplicate packets.

As it can be seen from Fig. 3-14a, the end–to–end packet delivery ratio increases with the number of available uplinks. The proposed uplink/downlink decoupling solution can provide a small performance improvement even when the channel conditions are good (PL = 0.05). On the other hand the performance improvements are significant when the channel conditions get worse. In particular this solution allows to turn an essentially broken channel (4 out of 5 dropped packets) into a usable channel (1 out of 2 dropped packets). Finally, in Fig. 3-14b and Fig. 3-14c, we can see the impact of the duplicate

filtering VNF on the bandwidth utilization. As expected without the VNF the bandwidth utilization increases with the number of uplinks, while using the VNF filtering the bandwidth utilization does not exceed the nominal goodput.

3.8 Conclusions and Future Work

In this section we presented our preliminary work on RAN sharing featuring a novel formulation of the VNF placement problem encompassing also radio access VNFs. We then introduced a ILP–based algorithm for small networks and a scalable heuristic, named WiNE, for larger deployments. Numerical simulations showed that the proposed heuristic can approximate the performance of the ILP–based placement algorithms for realistic substrate networks.

Finally, we reported on a preliminary proof–of–concept implementation of the proposed solution for Enterprise WLANs. As future work, we plan to investigate the resiliency properties of WiNE in case of node and link failures and to study how VNF placement can be optimized by taking into account wireless clients distribution and user mobility. We also plan to extend both the problem formulation and the implementation to LTE networks. All the code is released in conjunction with the COHERENT SDK in Deliverable D2.3.

4. RAN and Infrastructure Sharing

Mobile network operators need to meet the growing demand for data traffic by emphasizing on network provisioning, which increases capital expenditure (CAPEX) and operational expenditure (OPEX) correspondingly. In that context active RAN and Infrastructure sharing is likely to be the next significant evolutionary step, unlocking even greater CAPEX and OPEX efficiencies than traditional schemes. In more detail, RAN and Infrastructure sharing permits the virtualization of the LTE eNBs depending on the network state and enables an operator to lease on-demand the physical infrastructure and resources of additional eNBs (usually owned by different operators).

In principle, different kinds of active infrastructure sharing vary in terms of the degree of sharing. 3GPP has defined and ratified different kinds of architecture with varying degrees of sharing [**3GPP TS 23.251**]:

1. Multi-Operator RAN (MORAN) is the simplest scenario, in which only equipment is shared
2. Multi-Operator Core Network (MOCN), in which both spectrum and equipment are shared
3. Gateway Core Network (GWCN) is where both the RAN and some elements of the core network are shared.

In the following we present a generic framework developed for the Multi-operator/multial-service provider case operating over the virtualized MEC-enabled LTE network. Our framework is generic enough and with the necessary modifications can be applied in different kinds of RAN/infrastructure sharing.

4.1 Motivation

Although cloud computing can be used to meet traditional challenges, like scalability concerns and provide for fast resource provisioning times, a multifaceted analysis is required when it comes in multi-operator environments with time-critical applications and services. We claim that the service importance must be at the epicenter when it comes to the scheduling and placement decision of whether to deploy the service at the edge network or not. For example, Virtual machine (VM) scheduling decisions should avoid SLA violations for popular or time-critical services, and be fair between the service providers. In the analysis that follows a Lyapunov optimization framework is derived to solve this stochastic optimization problem that aims to maximize the revenue of the physical infrastructure owner in a multi-network operator-sharing environment with time-critical SLAs. A series of simulation experiments validate the high effectiveness of the proposed approach over benchmarking ones.

The ever-increasing popularity and requirements of mobile applications augment the need for efficient mobile computing architectures and efficient utilization of the wireless network physical resources. The core idea in the MEC design paradigm [Patel14], [NGMN] is to improve the end-user experience by utilizing cloud computing technologies and virtualized IT-based resources at the edge of the network. These are used in order to facilitate rapid service and function deployment, and the delivery of Over-The-Top (OTT) applications closer to the user. The concept considers multiple Radio Access Network (RAN) nodes (e.g., eNodeB, Wi-Fi etc.) with aggregated computing/memory/storage power in local nano-data centers, hierarchically located above the. The idea is that a request for service or function must be handled at the edge cloud; the request is forwarded to some external cloud by the gateway systems only if necessary. Note that the interfaces of a detailed MEC architecture are not yet standardized [ETSI].

Given the resource constraints in the edge network and the highly-volatile service demand, providing differentiated performance Service Level Agreements (SLAs) to various tenants and service providers that compete for the edge network resources is not trivial. There is some previous work on SLA-driven VM placement in the generalized cloud (like [Shen11], [Jin12] [Garg14]), mostly based on demand forecasting for resource scheduling. However, in contrast to general cloud computing environments, in MEC, it is the limitation on the number of physical server resources that requires proper modeling for extreme efficiency. Having this in mind, while also considering that a) multiple operators and stakeholders come into play concurrently to massively offer services, and b) there is lack of

distinction between time-critical and non-time-critical services, the area of SLA-driven VM scheduling in the MEC context remains highly unexplored.

In the analysis that follows, we study a VM scheduling and placement problem, where the VMs are used to support services deployment at the edge network. In our approach, it is the SLA requirements set on a per provider basis, that drives the service deployment and the relevant VM instrumentation. Intuitively, a service with strict performance requirements must be preferred for deployment at the edge network, as compared to a service with loose performance requirements. In the case of multiple providers with a mixture of time-critical and non-time-critical services, a non-trivial scheduling problem arises for the Mobile Edge Computing infrastructure owner (MEC-IaaS) provider, i.e., which service(s) from which provider(s) and where to be deployed at the edge. In our devised approach, the goal of the decision process is to maximize the MEC-IaaS provider revenue, while maximizing QoE for the customers of the time-critical services.

In more detail, the MEC-IaaS provider orchestrates rapid installation of services deployed on VMs for various service providers or mobile operators. A service provider can be any stakeholder that a client is associated with, in order to receive services (e.g., its mobile telecom operator). Henceforth, we will use the term mobile operators interchangeably with the service providers. A connected customer may request for services that are deployed in VMs that reside at the edge network or at some external cloud. In the former case, there is no need to create egress traffic beyond the edge network for receiving the services; hence, the delay is minimum and the QoE can reach to its extreme. Note that, potentially, the service to be deployed can be a web service or application or even a Virtual Network Function (VNF) (e.g., S-GW of the EPC network in LTE). This type of generality gives us the flexibility to apply the scheduling principles in various RAN/network resources sharing concepts. A VM of a specific type at the edge cloud is assumed to be able to handle efficiently a certain workload for a specific service. Excess workload is routed to a VM of the service at an external cloud and results to a SLA violation, which is penalized with a certain fee.

We model the service/VM requests made by the mobile operators as a queuing system. Our scheduling approach selects for every mobile operator a number of VMs of specific types to place at the edge network taking into account a) the SLAs of the services that need to be scheduled, b) the physical limitations of the edge-cloud nodes and c) fairness among service providers. The number of requested VMs per service provider is dynamically adapted to the workload of the services of the provider over time. At each slot, if there are no available resources at the edge cloud, the excess VMs requested are deployed to some external cloud system.

The main contributions of this paper are as follows:

- We formulate a joint optimization problem where the MEC-IaaS provider aims to maximize its expected lifetime revenue when deploying VMs that host services and minimize SLA violations for the deployed services, while being fair among the service providers.
- The problem formulation takes into account not only the service requests by the service providers, but also the expected workload in the short run. The idea is to deploy VMs that host popular services.
- Due to stochastic nature of the problem, we employ Lyapunov optimization [Neely10], [Geo2006], in order to derive fast myopic VM placement that is close to optimal in the long run.
- Because of the power of Lyapunov optimization, we make no assumptions on the arrival rate or the service lifetime distributions, which are considered unknown. Thus, our methodology can be used to derive optimal bounds for any arbitrary arrival and lifetime distributions, as long as the relevant random variables that constitute the stochastic nature of the problem are bounded.
- We evaluate our approach by extensive simulations of the proposed solution against variations of well-known benchmark techniques (i.e., First Fit).

We believe that our work can be used in multiple application scenarios and various 5G use cases, like the ones identified by NGMN. Also, note that, although we target the MEC environment, our scheduling and placement approach is quite generic and it can be employed in different cloud settings.

4.2 System Model

We assume a system with a set of N = {1, 2, …, N} of servers at the edge network. Let P = {1, 2, …, *P}* denote the set of Mobile Service Operators (e.g., MSO A, MSO B, etc.). Every wireless client in the area can be associated with such a provider, in order to receive services.

Controller operation: Our system operates in time slots. During each time slot, every provider sends requests to the cloud infrastructure owner (MEC-IaaS provider) to deploy a specific service. This service must be deployed over a VM at some server. The lifetime of deployment is also part of the request, as opposed to the VM type. Every request for some service with a certain SLA for performance is actually mapped to a VM request of a certain type by the controller (e.g., an advertisement service can be deployed in a "small" VM, while a time-critical service may need more powerful VMs). We explain the mapping process in detail in the following. For ease of notation, we assume a predefined set of available VM types (i.e., "small", "medium", "large") and let V denote the set with all the VM types. Also, let R be the set of capabilities of any physical or virtual machine, i.e., $R = \{cpu, memory, storage, bandwidth\}$.

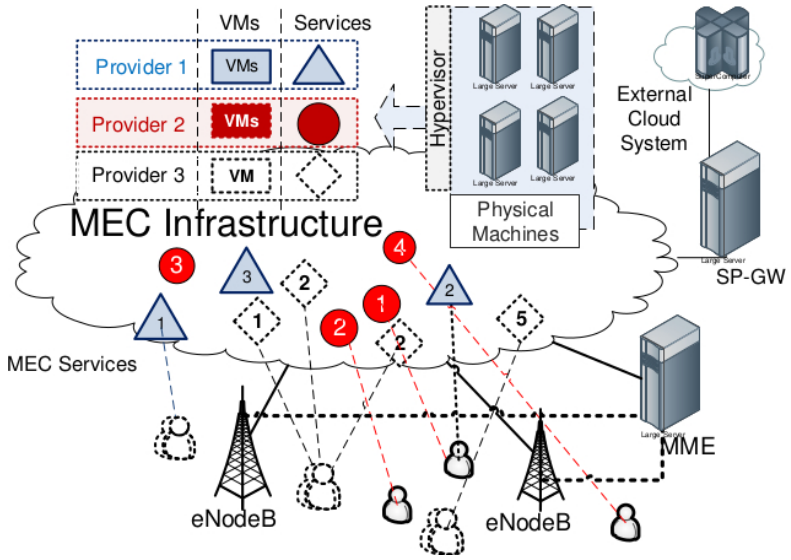


Figure 4-1: The MEC concept in LTE networks and services

The control actions are taken at the beginning of each time-slot. At each control instant, the controller of the MEC-IaaS provider is responsible to decide: a) which new services are to be deployed, b) which VM type the service request is mapped to and c) in which server to deploy the VM at. If there is no available capacity, VMs are deployed at some remote cloud system. In our model, at the beginning of each time slot, the controller flushes all the enqueued requests for all the providers queues and no service backlog is transferred to the next slot. For simplicity and clarity of presentation, we refrain from considering VM migrations in this work. **Live VM migration** refers to the process of moving a running virtual machine or application between different physical machines without disconnecting the client or application. We plan to enhance our model and take into the account the possibility of VM migrations in the future.

The controller operation is based on the incentive to maximize the revenue of the MEC-IaaS provider by deploying the most popular (by means of users requests) and performance- demanding services, while utilizing edge-cloud resources as much as possible. The deployment of more performance-demanding services (or applications or even virtual functions) at the edge servers increases the QoE of end users and network utilization efficiency at the same time. This is a good motivation for the service provider to pay the MEC-IaaS provider to host its services. We explain this process in detail below.

Description of the Service Request process: From the business perspective, every provider requires the satisfaction of a specific SLA for each service. With our approach even very strict SLAs can be met by deploying the service at VMs at the actual edge, following an optimal strategy. Note that each VM

instance that runs a specific service for a provider can serve up to a maximum number of client requests, while respecting the service SLA. Client service requests that cannot be handled by VMs at the edge incur a certain penalty payable to the service provider (e.g., similar to the penalty fee described in the Amazon EC2 SLA for service unavailability).

Let $S_i = (S_{i,j})$ denote the set of all the services j the provider $i \in P$ wants to deploy. Let S denote the set of all services $S = \cup S_i$. Every service can be deployed in some VMs of type $v \in V$. The service model can be quite extensive, meaning that the services deployed may be related to interactive services, batch requests, or even to VNFs. In this work, we do not get into this level of detail. We only consider the request pattern associated with each service.

Requests for Specific Service Instances: For every service $s \in S$, there is a) the lifetime period of deployment and b) an associated request rate from mobile end users and/or other applications that are using it. The client request rate of a service can be either estimated by the IaaS controller based on prior measurements of client requests for this service deployed at the edge or some exterior cloud from mobile users and/or other applications, or set by the Mobile Service Operator (MSO) based on private estimates (e.g., MSO 1 makes an initial estimate of 1000k requests per sec for some service he wants to deploy). From a practical perspective, by using packet steering techniques or SDN methodologies, we can obtain per flow/service statistics. In either case, in the next slot, just before the control decision, the estimate of the client request rate per service is adjusted according to the actual load using various moving average techniques, such as Simple or Exponential Moving Average.

We denote as $r_j^s(t)$ the aggregated rate for the expected number of client requests made by all mobile users or applications/functions associated with service provider j for service s. Moreover, we denote as $\tilde{r}_j^s(t)$ the actual request rate by all mobile users or applications associated with service provider j for service s when s is hosted at the edge network and as $\hat{r}^s(t)$ when s is hosted in some remote cloud (this is due to the fact that the service may be hosted partially in the cloud and partially locally). Then,

$$r^s{}_j(t) = \tilde{r}_j(t) + \hat{r}_j(t). \qquad (1)$$

All these rates are calculated based on prior statistics at the end of each slot and are used by the IaaS controller for the control decision in the beginning of the next slot.

Mapping Service Requests to VM Requests: In order to satisfy the client request rate for some service, a number of VMs is required. This relation is provided by some function

$f(\cdot)$, which is defined as follows:

$$f : RQ \times S \times P \longrightarrow N^{|V|} \qquad (2)$$

$f(\cdot)$ gives the vector of the numbers of VMs per VM type for handling requests $r^s(t) \in RQ$ for service s according to the SLA requirements of customer j. For example, assume that provider j wants to deploy service s and it is expecting a request rate of $r^s(t) = 10000$ requests per slot, with some specific SLA requirements (e.g., 10ms average respond time). If function f returns the vector $(2, 1, 0)$, this means that this SLA can be satisfied with 2 small VMs or with 1 one medium VM. In this context, we also define as $\xi^{v,s}$ to be the maximum request rate for service s that can be served by a VM of type v within the delay bound per request satisfied by the SLA to customer j for service s.

For example, as depicted in Fig. 4-2, we benchmark an Apache web server deployed on a VM, where we bring the CPU of the host machine in overload, using the Linux *stress* tool. In practice, this CPU overload of the host machine can be due to multi-VM operations or other functionality. As we can observe in Fig. 2, the maximum rate of requests that can satisfied under host CPU overload, within a 95-percentile delay bound of 73ms per HTTP request, is ξ=462.578 requests per second. That is, a decrease of almost 180 requests per sec from the case where the host is not stressed. For lower delay bounds or VMs with limited resources, ξ values for this service are expected to be lower. Employing similar stress tests per service s and per VM type v, one can find $\xi^{v,s}$ values for the delay bound per service s specified at the SLA of service provider j.

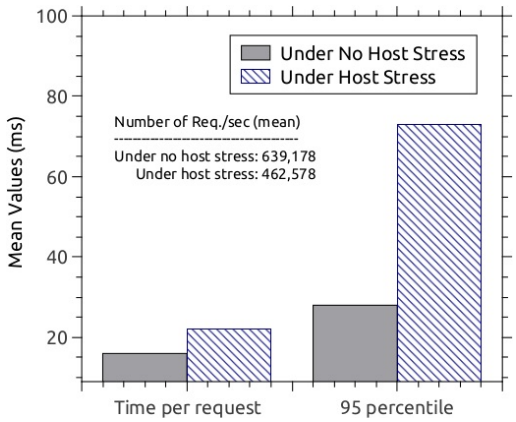


Figure 4-2: Stressing Apache service at a host machine

The problem of finally mapping service requests to VM requests resembles the *Knapsack problem* as follows: Each VM of type v has a rent of w_v and serves up to $\xi^{v,s}$ client requests per slot. We need to serve W_S requests for service s. We employ a greedy approximation algorithm, which is a variation of [10], as follows: Sort the VM types in decreasing order of requests per unit of rent, $\xi_{v,s}/w_v$.

Add the VM types to the sack starting with as many copies as possible from the first VM type, so that adding one more VM of that VM type keeps the total number of requests satisfied lower than or equal to W_s. Then, continue adding VMs of the second VM type in the same manner and so on, until all VM types are examined. Finally, add one VM of the cheapest type to the sack, if the total number of satisfied requests is lower than W_s. Note that this last VM is guaranteed to increase the total number of satisfied requests to a value more than Ws., if m is the maximum value of items that fit into the sack, then the greedy algorithm is guaranteed to achieve at least a value of m / 2, while satisfying W_s client requests (This bound comes from Dantzig' greedy approximation algorithm).

Capacity Constraints: A VM can be scheduled to some physical machine if its resource specification does not violate the physical capacity of the server in any of its resource pools. Let the vector $P_i = [p_l^j], j = 1,..,R$ denote the physical capabilities *R of physical or virtual machine i*. Any VM type $v \in V$ defines a vector $M_v = [m_v^j], j = 1,..,R$ with the physical capabilities R required. Moreover, we define $n_{i,j}^{v,S}(t)$ as the number of VMs of type v that are hosted in physical machine i at time slot t for provider j and service s at the edge. Let $n(t) = (n_{i,j}^{v,S}(t))$ be a vector with all the allocations of VMs in the edge cloud at time slot t. This is a state vector for our system.

See Fig. 4-3, for the outcome of the control scheduling and placement process. This is a cloud Allocation Matrix with the information of where to actually deploy and instantiate services on per provider and service basis.

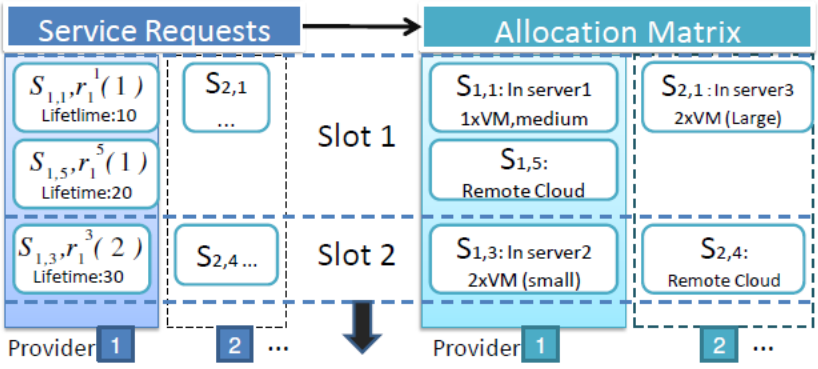


Figure 4-3: An example allocation matrix.

Justification of the System Model: With the above analysis, we stress the fact that some services are more important than others and have different impact on the revenue that the service provider enjoys. By more important, we mean that more client requests are heading for these services or the service has timecritical requirements reflected in higher penalties for SLA violations. At the end of the time slot, if some service is not deployed at the edge, the MSO has the ability to just resend the request at the next time slot. Meanwhile, there is no service idle-time, since the service would be deployed in some external cloud.

4.3 Problem Statements

We define as $\omega(t) = (A(t), D(t), r(t))$ a random event at time slot t. A random event comprises the service/VMs request arrivals, the VM departures (deletions) and the client request rates for the various services. We denote as $A(t) = (A_j^{v,s}(t))$ the number of VMs of type v for service s from provider $\boldsymbol{j}$ at slot t. Based on the SLAs and service profiling on the different VM types, the function $\boldsymbol{f}(\cdot)$ determines the number of VMs per VM type required for handling client requests for each service within its SLA requirements. Also, $D(t) = (D_j^{v,s}(t))$ is the number of VMs of type v for service s of provider $\boldsymbol{j}$ that terminate from server $\boldsymbol{i}$ at slot t.

At the beginning of each time slot t, the controller takes as input the instantiation of the random event $\omega(t)$ and the edge cloud system state in terms of VMs deployed for the various services of the different MSOs, and determines the new VM deployments at the different nodes of the edge cloud. Let $a(t)$ denote the four dimensional vector of control actions at slot t decided by the scheduling policy in effect. $a_{i,j}^{v,s}(t) \in \{0, 1, \ldots, A_j^{v,s}(t)\}$, $\boldsymbol{i} \in N$, $\boldsymbol{j} \in P$, $v \in V$, $s \in S$ represents the number of VMs of type v for service s of MSO $\boldsymbol{j}$ to be deployed at time slot t at edge-cloud node i.

The objectives of the IaaS controller for selecting the control action $a(t)$ are the following:

Objective 1: For each VM of type v for service s that is created at the edge cloud, there is a rent price $w_{v,s}$ (i.e., an hourly rent price amortized per time epoch) that is paid to the edge cloud provider. The edge cloud provider would seek to allocate VMs to its physical machines, so as maximize its profit $W(t)$ per time slot t, given by:

$$W(t) = \sum_{i \in N} \sum_{p \in P} \sum_{v \in V} \sum_{s \in S} a_{i,j}^{v,s}(t) w_{v,s} \quad (3)$$

Over the time horizon, the cloud provider would seek to maximize the *expected cumulative profit* or equivalently the average expected profit over time, i.e.

$$max \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} E[W(t)] \quad (4)$$

which can be written as a minimization problem as

$$min \lim_{t \to \infty} -\frac{1}{t} \sum_{\tau=0}^{t-1} E[W(t)] \quad (5)$$

Objective 2: For each request that is not served by a VM residing at the edge cloud, there is a penalty arising from the potential SLA violation for this service. We aim to minimize the *overall penalty* for the requests not handled at the edge cloud. We define the penalty function $P(t)$ as:

$$P(t) = \sum_{j \in P} \sum_{s \in S} [r_j^s(t) - \sum_{v \in V} (\sum_{i \in N} n_{i,j}^{v,s}(t) + a_{i,j}^{v,s}(t) - D_{i,j}^{v,s}(t)) \xi_j^{v,s}] \pi_j^s \quad (6)$$

where π_j^s is the monetary penalty for not serving at the edge cloud a client request for service s of provider j. Then, our objective for minimizing the *average expected penalty over time* is given by

$$min \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} E[P(t)] \quad (7)$$

Objective 3: As a third objective, we seek to guarantee at least a minimum service at the edge cloud for all providers. This can be achieved by a *proportionally-fair* VM allocation among providers according to their relative SLA significance. To this end, we define a logarithmic utility function $u(\cdot)$ of the allocated resources x to each provider j as follows:

$$U_j=\varphi_j \log(x) \quad (8)$$

φ_j is the penalty for being unfair to provider j. We define the following function as an *unfairness metric*:

$$K(t) = -\sum_{j\in P}\varphi_j \ -\log(\sum_{i\in N}\sum_{v\in V}\sum_{i\in N} n_{i,j}^{u,s}(t) + a_{i,j}^{u,s}(t) - D_{i,j}^{u,s}(t)) \quad (9)$$

Thus, regarding fairness our long run objective is

$$min \lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1} E[K(t)] \quad (10)$$

Note that the logarithm in $K(t)$ can be approximated by a piecewise linear function, so that the objective in (10) is linear. Henceforth, such an approximation is assumed to be employed.

Constraints: The control actions a(t) to be selected by the IaaS controller at time slot t are constrained as follows. The VMs that will operate over some physical machine i at time slot t should not violate the physical capabilities p_i of the system for each resource. Let the vector $y_i(t)$ represent the resource residuals at host i at time slot t, as an effect of the control action a(t) and the random event $\omega(t)$ at time slot t (captured by the function $Y_i(a(t),\omega(t))$), given by:

$$\begin{aligned}\mathbf{y}_i(t) &= Y_i(\mathbf{a}(t),\boldsymbol{\omega}(t)) \\ &= \sum_{j}^{P}\sum_{v}^{V}\sum_{s}^{S}(n_{i,j}^{v,s}(t) + a_i^{v,s}(t) - D_{i,j}^{v,s}(t))\mathbf{m}_v \\ &- \mathbf{p}_i \ .\end{aligned}$$

$n_{i,j}^{u}(t)$ is the number of VMs of type v that operate in physical machine i at any time slot t for provider j. Then, the feasibility constraints are given by:

$$y_i(t) \le 0, \quad \forall i \in N. \quad (11)$$

Another constraint is related with the control actions: the VMs to be deployed at time slot t should be lower or equal to those requested, i.e.

$$\sum_{i\in\mathcal{N}} a_{i,j}^{s,v}(t) \le A_j^{s,v}(t), \forall s\in\mathcal{S}, v\in\mathcal{V}, j\in\mathcal{P} \ . \quad (12)$$

The overall optimization problem becomes:

$$\textbf{Minimize: } \lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\Big(-E[W(\tau)] + E[P(\tau)] + E[K(\tau)]\Big)$$

$$\textbf{s.t. } (11), (12) \quad (13)$$

Therefore, the controller tries to select the VMs to be deployed for the various service providers over time, so as to maximize revenue (and edge-cloud utilization), deploy the most popular services, minimize the SLA violations and be fair among service providers at the same time.

4.4 Proposed Approach: Lyapunov Optimization-based VM Scheduling (LBVS)

The controller at each time slot selects the VMs to be deployed for the various service requests by the different MSOs. The edge cloud carries the effect of control decisions to the future, i.e., it is a *dynamic system*, while for the state there are some stability conditions. We exploit, Lyapunov optimization, and specifically the drift-plus-penalty technique [8], [9], [11], in order to achieve the long-run objective in problem (13). With this approach the minimization of the objective function depends only on the control actions and the random event (i.e., service arrivals and departures, client requests) at each slot and not on the whole system state, history or knowledge of distribution mean values. The alternative of calculating the optimal control actions over all time slots using dynamic programming would suffer from the curse of dimensionality and it would be impractical, as it would require knowledge of all random events over time in advance. We assume that this problem is *feasible*, i.e., a control action that can satisfy all of the desired constraints exists. The virtual queues for the constraints are defined as follows:

$$\mathbf{Q}_i(t+1) = \max[\mathbf{Q}_i(t) + \mathbf{y}_i(t), \mathbf{0}], \ \forall i \in N \tag{14}$$

The Lyapunov function for the virtual queues is

$$L(t) = \frac{1}{2} \sum_{i \in \mathcal{N}} \mathbf{Q}_i^2(t) \, .$$

Using the drift-plus-penalty approach, we have

$$\begin{aligned} \Delta(t) + V(-W(t) + P(t) + K(t)) \leq B \\ + V(-W(t) + P(t) + K(t)) + \sum_{i \in \mathcal{N}} \mathbf{Q}_i(t) \mathbf{y}_i(t) \, , \end{aligned} \tag{15}$$

where Δ(t) = L(t + 1) − L(t) and B is a positive constant. The V parameter can be chosen adequately large, so as to ensure the time average of the objective is arbitrarily close to optimal, with a corresponding side-effect in the average virtual queue size. Thus, we can find almost optimal VM placement actions by greedily minimizing at each slot t the following problem:

$$\begin{aligned} &Minimize: V\big(-W(t) + P(t) + K(t)\big) + \sum\nolimits_{i \in N} \boldsymbol{Q}_i(t) \boldsymbol{y}_i(t) \\ &\text{s.t.(11) ,(12)} \end{aligned} \tag{16}$$

Due to Jensen inequality, it can be shown that an optimal solution to (13) can be achieved by solutions of the type a(t) =a*, where a* is a vector that solves the linear problem (16). Further, any time-averaged vector lim a(t) corresponding to a solution of the time-averaged problem (13) must solve the linear problem (16). Therefore, the original optimization problem (13) can be solved by taking the time average of the decisions made when the drift-plus-penalty algorithm (16) is applied. The linear program (16) admits a worst-case polynomial-time algorithm with complexity O((|V| · |S| · |N| · |P|)3.5L), where |V| · |S| · |N| · |P| is the number of problem variables that can be encoded in L input bits [Kar1984]. At each slot, a solver can be used to solve (16).

4.5 Evaluation

We evaluate the proposed scheduling approach by means of simulation experiments. The goals of the evaluation process were to verify the theoretical results, while also to demonstrate how the various edge-cloud and statistical parameters affect the performance of our approach. In addition, in order to demonstrate the compensation of optimality, we contrast the proposed scheduling policy with the well-known First Fit algorithm [Xia2010], i.e., place each VM into the first host where it fits, which is used as benchmark. First Fit algorithm is supposed to consider VMs according to two different

orderings: i) Round-Robin (FF-RR), and ii) randomly (FF-Random). In FF- RR, small VMs are considered prior to medium ones, which are considered prior to large ones in a Round-Robin fashion among providers. In both cases of *edge-cloud parameters* and *statistical parameters*, a number of variables can be tuned, which could affect the effectiveness of the proposed scheduling policy in terms of convergence speed. For example, such edge- cloud parameters are the number of providers, the number of services or the number of host machines, and such statistical parameters, are the service request arrival rate distribution, the service lifetime distribution, the SLA-violation penalty and other weight factors. For brevity, we present here only indicative results out of extensive simulations.

4.5.1 Simulation Setup

For the evaluation of the proposed policies, we used a custom JAVA simulator and the IBM CPLEX optimization kit, which solves linear program (16) and determines VM deployments per time slot. For clarity of presentation of the system dynamics, we present the performance of our scheduling approach in a simple scenario with 2 providers. However, note that larger-scale experiments were performed with similar findings.

Each provider sends requests for a single web service deployment over the edge cloud comprising a single physical machine. The same SLA contract is provided to both providers. This will be the basic setup for our simulation experiments unless stated otherwise. For this basic setup (see Table 4-1), the service requests follow a Poisson distribution for both providers, whereas provider 2 has a higher service deployment request rate.

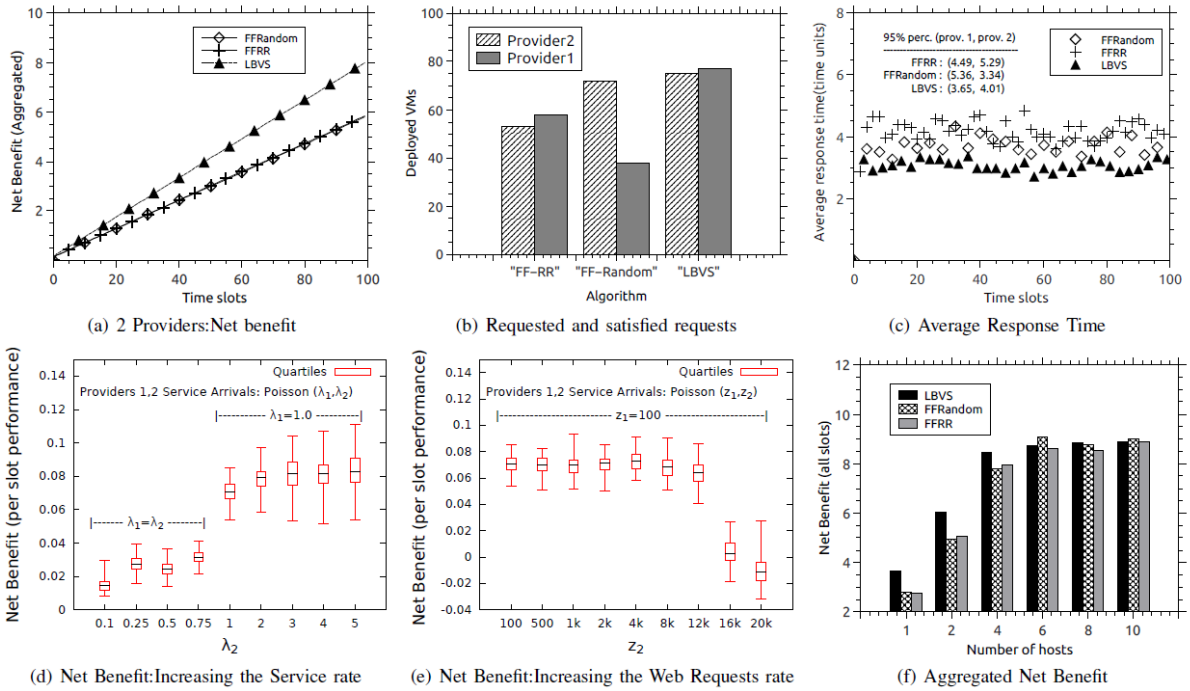
Furthermore, in order to estimate the expected client request load per service in runtime, a simple moving average with window size 3 is employed based on prior load measurements. Each simulation run lasts 100 time slots and we average our results over 100 runs.

Table 4-1: Basic Setup - Simulation Parameters

Resources	Host	VM (small,mdm,lrg)
CPU	8	(1,2,4)
Memory	100000	(1024,2024,4024)
Storage	100	(10,20,40)
Bandwidth	10	(1,1,1)
Provider Config.	**Provider 1**	**Provider 2**
Service arrival	Poisson, λ=1	Poisson, λ=2
Web Request arrival	Poisson, z=100	Poisson, z=100
Service lifetime	Exponential, μ=2	Exponential, μ=2
Local Cloud Resp. time	Exponential, μ=1	Exponential, μ=1
Rem. Cloud Resp. time	Exponential, μ=10	Exponential, μ=10
Client Requests within SLA	ξ^{small}=5000, ξ^{mdm}=10000, ξ^{lrg} =20000	ξ^{small}=5000, ξ^{mdm}=10000, ξ^{lrg} =20000
VM Rent (small, mdm, lrg)	0.026, 0.052, 0.104	0.026, 0.052, 0.104
SLA-violation Penalty	1	1
Unfairness Weight	1	1

4.5.2 Simulation Results

In Fig. 4(a), we present a performance comparison of the three scheduling policies considered in terms of cumulative net benefit in the basic scenario. Evidently, even in this simple case where a single service is requested by each provider, the LBVS approach outperforms the other two policies, since it is more capable to adapt on service-deployment load differences (provider 2 has increased service deployment rate). Therefore, LBVS results to better utilization of the physical resources, as compared to both variants of the First Fit scheduling policy.



(a) 2 Providers:Net benefit (b) Requested and satisfied requests (c) Average Response Time

(d) Net Benefit:Increasing the Service rate (e) Net Benefit:Increasing the Web Requests rate (f) Aggregated Net Benefit

Figure 4-4: Evaluation Results

This result is also explained by Fig. 4-4(b) presenting the number of deployed VMs per scheduling policy. Although the total number of VM deployments can reach a saturation point due to capacity constraints, LBVS approaches closer to the theoretically optimal bound of VM deployments without any knowledge of the statistical properties of the random variables (i.e., service-deployment requests, service lifetime, load of client requests) of the problem. Moreover, note that the LBVS approach is fairer for the providers, as compared to the benchmarking scheduling policies. The better utilization of the Edge-Cloud by the LBVS scheduling policy as compared to the benchmarking ones is also reflected to the response time of client requests in Fig. 4(c). As depicted therein, LBVS achieves lower response time than rival scheduling policies.

Then, we investigate the effects of increasing the service load (and consequently the VM load), while keeping fixed the client-request load for the service. As shown in Fig. 4(d), keeping the VM-deployment rate of provider 1 fixed (λ_1=1 VM request/slot) and increasing the VM-deployment rate of provider 2, a saturation point is reached at point $\lambda_1 = \lambda_2 = 1$, after which the net benefit of the Edge-IaaS cannot be in- creased. This saturation point means that resource capacity limits of the Edge-Could have been reached. Note that, in this case, after the capacity limit of the Edge-Cloud has been reached, the net benefit of the Edge-IaaS that equals (rent from VM deployments) - (penalty for SLA violation) - (penalty for being unfair) (i.e., eq. (3) – eq. (6) - eq. (9)) remains constant. This is because, no additional rents can be obtained, while penalty factors remain unaltered, since web-service client requests remain fixed. This is not the case when the client-request load for the web service increases, as depicted in Fig. 4(e). Specifically, we examine the case where the VM/service load is fixed, while we increase the web load for both providers. As depicted in Fig. 4-4(e), after the total number of client requests that can be served within the SLA by the VMs deployed at the edge cloud is exceeded, the SLA-violation penalty increases, while the rent from deployed VMs and the unfairness penalty remain stable. The concluding remark for the cases presented in Figs. 4-4(d) and 4 - 4(e) is that a very careful SLA definition is necessary when strict performance guarantees are to be provided to services. As experimentally shown, even with a close-to optimal scheduling policy, such as LBVS, penalties arising from SLA violations may be critical and may even nullify the profit from deploying VMs in the edge cloud.

Finally, in Fig. 4 - 4(f), we present the total cumulative net benefit for both providers over time as the number of nodes of the edge cloud increases. As expected, when the edge cloud has abundance of resources, every scheduling policy achieves a high net benefit for the Edge-IaaS, as all service/VM deployment requests are satisfied in this case. However, the more limited the edge-cloud resources, the higher the performance gap achieved by the LBVS scheduling policy over the benchmark ones.

4.6 Conclusions and Future Work

We investigated the problem of service/VM scheduling in environments with limited physical resources, like the edge cloud, where multiple service providers and service operators need to deploy time-critical services over a shared infrastructure. We proposed an SLA-driven VM scheduling approach that maximizes the revenue of the Edge-IaaS and minimizes SLA violations, while being fair to the various service providers. The problem was solved by a Lyapunov optimization framework and the performance our proposed scheduling approach was experimentally validated by means of simulation experiments. As a future work, we plan to enhance our model to consider VM migrations and more advanced load-forecasting methods. We also intend to implement our LBVS policy in a real testbed and evaluate it in RAN-sharing concepts.

5. Network Graph-based Energy Consumption Management

Small cell networks are key components in 5G networks to boost the network capacity, improve spectrum and energy efficiency, and enable flexible and new services. Due to both the flexible spectrum access and the flexible deployment of small cells, the inter-cell coordination becomes critical for the performance of the network. In this work, based on the key concept in software defined networking (SDN), we first introduce the network graph approach as a tool for the control and coordination among small cells. The network graph is constructed from the abstracted network state information extracted from underlying base stations. It shields the logical centralized control unit from implementation details of the underlying physical layer and thus reduces the control overhead in a centralized solution. We use the network graph for network energy saving in small cell networks, in which network graphs are used to decide the optimal set of small cells in the network. The cells outside of this set can then be switched off for energy saving. We propose three types of network graphs with different network state details. Based on these graphs, we formulate the energy saving problem as an integer linear programming (ILP) problem, and propose practical algorithms to solve the problem. The performance of the algorithms is studied by simulation. It shows the potential of the proposed network graph approach for the inter-cell resource coordination in small cell networks.

5.1 Motivation

Mobile networks are evolving to the next generation, in which small cells are expected to play an important role [Andrews2012]. Since 2012 the number of small cell base stations (BS) around the world has surpassed that of traditional cellular BSs. The prevalence of small cells brings higher network capacity, lower energy consumption, more efficient use of spectrum, and lower operational expenditure (OPEX). Meanwhile, the wide and dense deployment of small cells introduces new research challenges. New solutions are demanded for spectrum management, interference coordination, traffic steering, mobility, backhaul technologies and energy efficiency.

Technical challenges regarding small cells are well summarized in [Andrews2012]. Due to the spectrum reuse among macrocells and small cells, the interference coordination and joint resource allocation among the network become necessary [Singh2014]. This requires efficient coordination among cells. In multitier cellular networks with small cells, traffic offloading is an important means to balance the resource utilization in the network [Chen2015]. However, the load coupling in small cells makes the traffic offloading decision a complex optimization problem [Ho2014]. In addition to conventional optimization approaches, both centralized and distributed learning algorithms are proposed [Chen2015]. Traffic offloading is utilized to improve energy efficiency in multi-tier cellular networks [Budzisz2014].

While many research problems in small cell networks have been intensively studied in recent years, they are based on the current 3G and 4G cellular network architecture. The coupling in small cells with regard to spectrum access, interference and traffic load calls for effective and efficient inter-cell coordination. Similar to software defined networking (SDN) for Internet, the logical centralized control will be a powerful means to solve the cooperation and coordination problem in small cell networks.

The centralized control solution can be tracked back to 2G systems. However, the flexibility and scalability are missing therein for large scale small cell networks. New control architecture for next generation mobile networks is needed. We proposed in [Chen2014] the software defined radio access networks (SD-RAN) control architecture to satisfy this need. The key idea is to introduce a logical centralized control and coordination framework for heterogeneous mobile networks, in which network graphs reflecting low layer states of networks are used at central control units for inter-cell resource coordination.

This work develops coordination algorithms based on the proposed control framework. We focus on the network energy saving problem in small cell networks. Since a user in the network may be covered by multiple cells, by re-associating users in the network, some cells in the network can switch to the energy saving mode. With network awareness by the proposed control framework, we can turn the working cell selection problem to a graph problem and solve it by rich optimization tools. The main contribution of

this work is to propose different network graphs, develop working cell selection algorithms, and study the energy saving performance of different network graphs.

We first introduce the abstract network graph concept. The network graph captures the abstracted network states of the underlying RANs and offer to tune selected parameters for network re-configurations. Based on the proposed network graphs, we develop algorithms to find the optimal cell set, which satisfies the quality of service (QoS) needs of users while minimizing the number of the working cells in the network in order to reduce the energy consumption. Finally, we evaluate the performance of the proposed cell selection algorithms.

5.2 Network Graph-based Approach

The network graph approach introduced in this work is based on the SD-RAN control architecture proposed in [Chen2014]. Similar to the idea in the SDN concept for Internet, the proposed approach aims to enable a logical centralized control solution for large scale heterogeneous mobile networks. It can be used to improve spectrum management, network energy saving, mobility management, and other middle to long term network re-configurations in mobile networks.

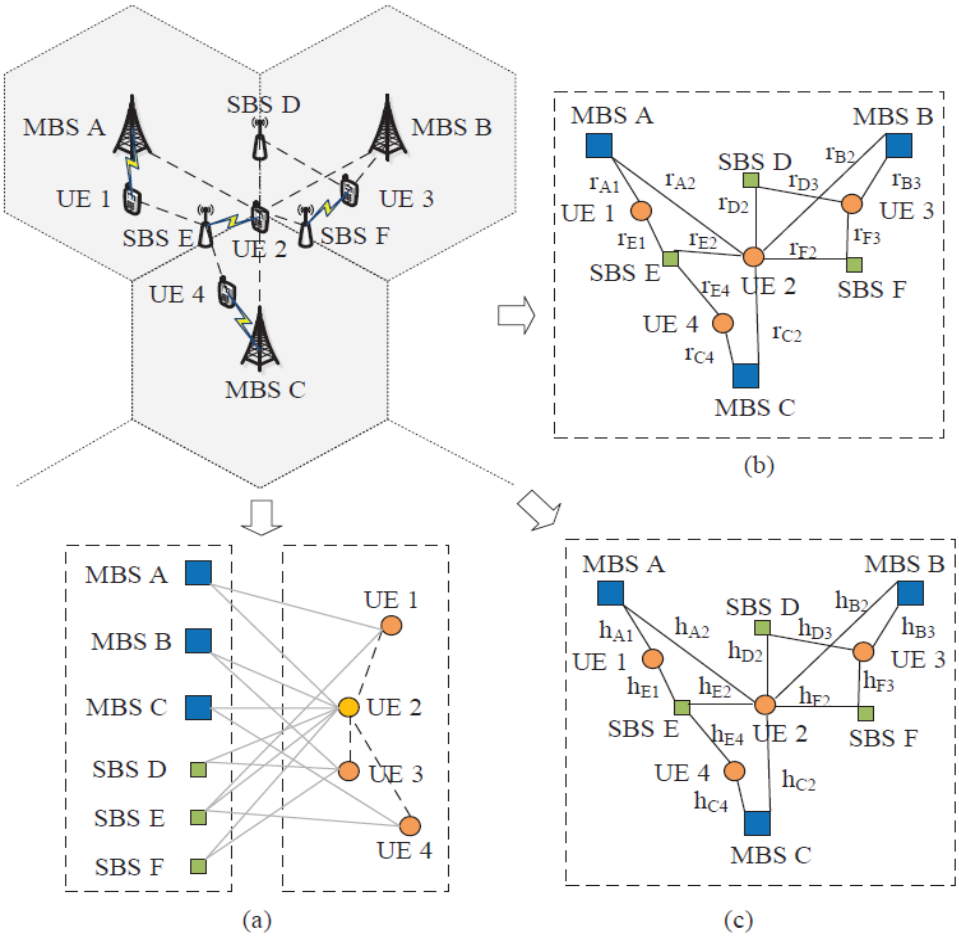


Fig. 5-1. An example of heterogeneous mobile network and three derived network graphs.

The network graph approach could be realized by additional control interfaces, protocols and control units within current cellular and wireless local area network (WLAN) network architecture. With necessary extensions, BSs and APs are functioned to report abstracted network states to the Centralized Controller and Coordinator (C3) Entity so that network graphs reflecting the underlying radio network reality can be used to tune necessary network parameters according to network performance targets. Fig. 5-1 provides an example of network graphs used in this work. In a network graph, an edge presents a

feasible connection between a BS/AP and use equipment (UE). While we focus on long term evolution (LTE) networks in this study, instead of using the terms eNodeB and home eNodeB for macrocell and small cell BS, we use macrocell BS (MBS) and small cell BS (SBS) through the study. The general term BS stands for both MBS and SBS. Moreover, we mix the use of the terms cell, BS and AP.

According to the details of reported state information, different types of network graphs may be generated. For instance, Fig. 5-1(a) only shows feasible connections between BS and UEs; Fig. 5-1(b) includes the path loss between BS and any potential UE, and therefore the interference-free link rate to a reachable BS; Fig. 5-1(c) contains channel states among BSs and UEs, which allow estimating inter-cell interference in the network graph. Incorporating additional information like power, traffic load, spectrum and QoS requirements in the network graph, different algorithms can be developed to get the optimal set of working BSs for network energy saving.

Centralized solutions for large scale networks normally suffer from the scalability and overhead problems. One key idea in the proposed SD-RAN control framework [Chen2014] is to extract abstracted and only necessary network state information from the physical and MAC layer of RANs to the logical central control unit. The abstraction here means that the central control unit needs not to know physical layer implementation details of underlying radio access technologies (RAT) but just necessary information to construct the abstracted view of underlying networks for inter-cell resource coordination and control. Therefore, it is critical to understand the principles to abstract low layer states for this control purpose.

5.3 System Model

We study in this work the network energy saving performance of different types of network graphs depicted in Fig. 5-1. The objective is to determine a minimal set of BSs in the network with coverage and capacity satisfying the QoS needs of UEs. We assume the above-mentioned SD-RAN architecture is available. We will develop BS selection algorithms based on available network graphs to obtain the minimal BS set.

More specifically, assume in a given region we have K regularly deployed MBSs, N ad-hoc deployed SBSs, and M' UEs which have low mobility. SBSs provide the open access to UEs. A UE only connects to one BS at a time. The MBSs and SBSs share the spectrum of the total bandwidth W. The spectrum access among MBSs and SBSs is flexible. They can use the spectrum partition method to avoid inter-cell interference when the network traffic load is low, or otherwise the spectrum sharing to improve the network capacity. We assume the central control unit will decide the proper spectrum access method for individual cells according to the network traffic conditions.

We study the downlink traffic in this work. But the work can be easily extended to the uplink and mixed cases. We assume the perfect neighbor discovery of a UE. A UE u_i is able to accurately estimate channel gain h_{ij} to the neighboring cell B_j. Therefore, the average channel gain h_{ij} is available in the network graph. For simplicity, we assume same type of BSs use the same transmission power. The transmission power for the MBS is P_M, and for the SBS is P_H. With the transmission power and channel gain available, a UE can find with which BSs it can associate by evaluating the signal to noise ratio (SNR).

Each UE has its minimal data rate requirement. For simplicity, we assume all UEs have the same rate requirement R. In our study, a BS tries to satisfy that rate to UEs. With the SNR and assigned bandwidth, the interference-free link rate between a BS and UE is estimated in the network graph. Assuming the UE takes all bandwidth W, the link rate between the UE u_i and the BS B_j is r_{ij}. The share of resource allocated by B_j to u_i is R/r_{ij}. The load of B_j, denoted by L_j, is the sum of R/r_{ij} from all associated UEs. We assume a BS will serve the load L_j and then turn to the idle mode. That means no extra data rate is provided for UEs. Note that those are assumptions used in the network graph to decide the working BS set. The actual data rate of a UE depends on the scheduling algorithm by the BS and inter-cell interference.

The inter-cell interference in a spectrum sharing network is a complex problem, mainly due to the coupling of inter-cell interference. In this work we use a simple interference model in the network graph. We will use more realistic models as in [Singh2014] [Soh2013] in our future work. We define two cells are neighbors when at least one UE can connect both. If the sum load of two neighboring cells is less than one, we assume a perfect intercell scheduling among them and thus no inter-cell interference. Otherwise, the UEs covered by both cells may suffer inter-cell interference and the amount is determined by the total load of both cells. In reality the inter-cell interference is worse and unpredictable. Therefore, the actual data rate of an UE may be less than the required R.

Following the power consumption model of LTE home eNodeB in [Imran2011] we assume it will consume 10 watts if switched on and 0 watt otherwise. Since the MBS needs to be always on in order to ensure the coverage, we assume it consumes the constant power and then leave it out in the computation of the minimal BS set. In the future study, we will take into account the load dependent power consumption model of the MBS and jointly consider MBSs and SBSs in the algorithms. Since we assume the SBSs consume the same amount of power, the problem turns to finding the minimal dominating set of small cells with the QoS constraints of UEs.

Three network graph models are used in the algorithm development. All network graphs have the incident matrix $A_{[N\times M]}$, indicating $A_{ij} = 1$ if u_i can associate with B_j, and otherwise $A_{ij} = 0$. Note that we only consider the SBSs in the network graph. UEs have the priority to connect to SBSs for high data rates. In the case of the first network graph shown in Fig. 5-1(a), the QoS constraint of the UE is not considered so the algorithm will obtain the minimal BS set. The case of the second network graph (Fig. 5-1(b)) considers the data rate requirement of the UE and the capacity limit of the SBS, but inter-cell interference is not considered. The third case (Fig. 5-1(c)) considers inter-cell interference on top of the second case. The afore-mentioned simplified inter-cell interference model is applied.

5.4 Problem Formulation

Given the set of UEs $\{ u_i : i = 1 \cdots \mathrm{M}' \}$ in which M of them are covered by SBSs, the set of BSs $\{ B_j : i = 1 \cdots \mathrm{K} + \mathrm{N} \}$ in which N of them are SBSs, the average power consumption P_j^e for the MBS B_j and P_{hj} for the SBS B_j, the incident matrix A from the network graph, the data rate requirements of UEs as R, the channel gain between BS and UE in network graph, and the transmission power of BSs as P_M, and P_H for MBS and SBS respectively, the original problem is to minimize the total average power consumption of the network as:

$$min \sum_{j=1}^{K} P_j^e + \sum_{j=1}^{N} y_j P_j^h \qquad (1)$$

where y_j is the indicator variable, while $y_j = 1$ if the SBS B_j is in the working BS set or otherwise 0. As we discussed in Section 5.3, the problem in Eqn. (1) can be reduced to:

$$min \sum_{j=1}^{N} y_j \qquad (2)$$

subject to:

$$y_j \geq x_{ij} \qquad (3)$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad (4)$$

$$A_{ij} \geq x_{ij} \qquad (5)$$

and the QoS constraints:

$$\sum_{i=1}^{M} \alpha_{ij} \leq 1 \qquad (6)$$

$$\alpha_{ij} = R / r_{ij}^{'} \qquad (7)$$

where $x_{ij} = 1$ if $u_i \in B_j$ or otherwise $x_{ij} = 0$; α_{ij} is the portion of resource taken by u_i from B_j, and $L_j = \sum_{i=1}^{M} \alpha_{ij}$; $r_{ij}^{'}$ is the link rate between u_i and B_j. In this study, we get α_{ij} from Eqn. (7) where $r_{ij}^{'}$ is estimated by assuming ui takes all bandwidth W for its transmission. Depending on the algorithm, $r_{ij}^{'}$ may or may not take into account the intercell interference. If no inter-cell interference is considered, $r_{ij}^{'} = r_{ij}$.

In the above formulated problem, the constraint in (3) only allows a UE to associate with one BS in the selected working BS set; the constraint in (4) limits a UE to connect only with one BS; the constraint in (5) associates a UE only with reachable BSs; the constraint in (6) adds the capacity limit of a BS to the problem.

In the following we refine the problem in three network graphs as described in Section 5.3.

Network graph case I: We have no QoS constraints in this type of network graph. The problem takes the simplest form, which is formulated by Eqn. $(2-5)$. It is an integer linear programming (ILP) problem, which can be effectively solved.

The solution provides the lower bound of the problem in Eqn. (2). However, the solution may end up with no realistic result as the cell capacity and interference are not considered. We will use the case I as the reference model to the other algorithms.

Network graph case II: In this type of network graph, we do not consider the interference from other cells. Therefore, we have $r_{ij}^{'} = r_{ij}$, which is the interference-free link rate for $u_i \in B_j$ in Eqn. (7). The problem is formulated by Eqn. $(2-7)$, which remains an ILP problem. Due to the QoS constraint, the problem may have no solution. In this case we will use other association method, for instance, a UE associates with a BS with the best SNR. Therefore, the outcome may result in more cells than that of the case I.

Network graph case III: The inter-cell interference is considered in this network graph. However, as far as the load coupling is concerned, the change of load and association in one cell will affect the interference to other cells and thus the achievable rate of UEs. The problem is not any more a linear programming problem. It involves the complex interference calculation and even demands the power control at BSs to find the minimal working BS set.

To simplify the problem, we use a simple interference model to calculate r'_{ij} in Eqn. (7). For $u_i \in B_j$, it has a number of neighbor cells, defined in $S = \{ B_q : q = 1 \cdots Q \}$. Then the probability of collision with a neighboring cell in S is

$$p_{iq}^{c} = \begin{cases} 0, & if\ L_j + L_q \leq 1 \\ L_j + L_q - 1, & otherwise \end{cases}$$

The probability that ui will have a successful transmission is then: $p_{ij}^{s} = \prod_{q \in S} (1 - p_{iq}^{c})$.
We can get the $r_{ij}^{'}$ as $r_{ij}^{'} = p_{ij}^{s} r_{ij}$.

It turns the problem to an ILP problem. However, the interference model will underestimate the real interference in the network, making the actual data rate of u_i less than R. Moreover, the ILP problem

may have no solution due to the QoS constraint. We use the same approach as in the case II to find the working BSs.

5.5 Proposed Approach

This section describes the proposed algorithm. Since the computation overhead of the ILP problem is proportional to the size of the network and the number of UEs, we first explore the structure of the network graph to reduce the computation complexity.

For all BSs in the graph, two BSs are connected if they are neighboring BSs. In a connected BS set, any two BSs are connected directly or through other BSs. A connected BS set and the UEs covered by them form a sub-graph. While it depends on the network topology, normally an ad-hoc deployed network can be decomposed to several small size subnetworks. By applying well established graph decomposition methods, e.g. Dulmage-Mendelsohn decomposition [Dulmage58], we can easily obtain the set of sub-graphs. We apply the algorithm on the sub-graphs. The combined results from all sub-graphs give the solution for the whole network.

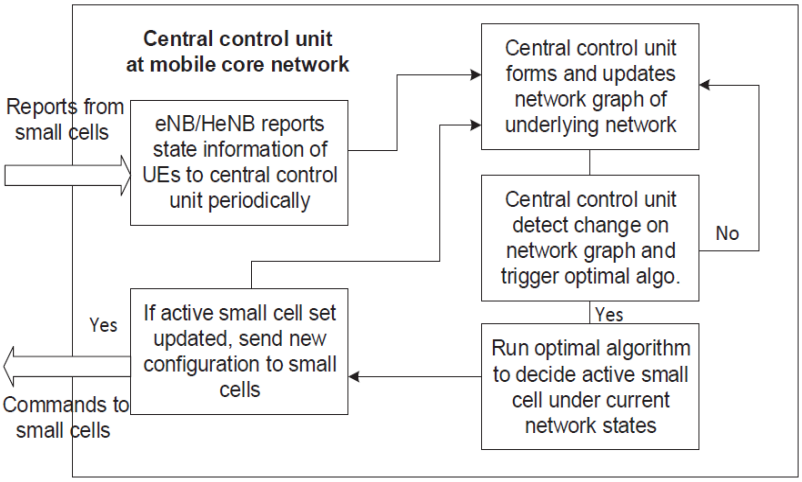


Fig. 5-2. Flow chart of algorithm.

The flow chart of the algorithm is shown in Fig. 5-2. Note that all three network graph cases use the same flow chart. First, the network graph is created by using the reported states from low layers. The graph decomposition will then break down the network graph into several sub-graphs. For each sub-graph, the ILP problem is set up and solved. We get the new working BS set by the results from all sub-graphs. The new result is compared with the previous result, and the new configuration will be sent to the underlying BSs if two results are different.

5.6 Evaluation

5.6.1 Simulation Environment

The performance of the algorithms is studied by simulation based on MATLAB. We deploy one MBS in the middle, and a number of SBSs and UEs uniformly distributed in a playground of size 500m × 500m. The SBS has the transmission power of 15dBm, and the coverage of 40m. As not all UEs are covered by the SBSs, the uncovered UEs will be served by the MBS. We assume the spectrum partition between MBS and SBS and thus no interference between the MBS and SBSs. The simplified spectrum access model allows us to focus on the network graph approach for SBSs. In the future work we will study more realistic spectrum sharing models.

The bandwidth used by the SBSs is 1MHz. The path loss model from [Cichon98] is used in the small cell path loss calculation, i.e. $L(dB) = 37 + 32log_{10}(d)$, where. $L(dB)$ is the path loss in dB, d is the distance between the SBS and UE.

To test the performance of algorithms under different QoS requirements, three data rate requirements are set for UEs, standing for very low, moderate, and very high data rate requirements compared to the capacity of the SBS. We assume the normal number of UEs in a SBS is 6. Under the system parameters listed above, 6 UEs at the edge of an SBS can each achieve 2.7Mbit/s. This is the moderate rate set in our simulation. The other two rates are then set as 2.7kbit/s and 6Mbit/s.

We develop the bestRate algorithm as a reference to compare the performance. In this algorithm, UEs simply associate with the SBSs with highest SNR. The data rate requirement of UEs is not considered in this reference algorithm.

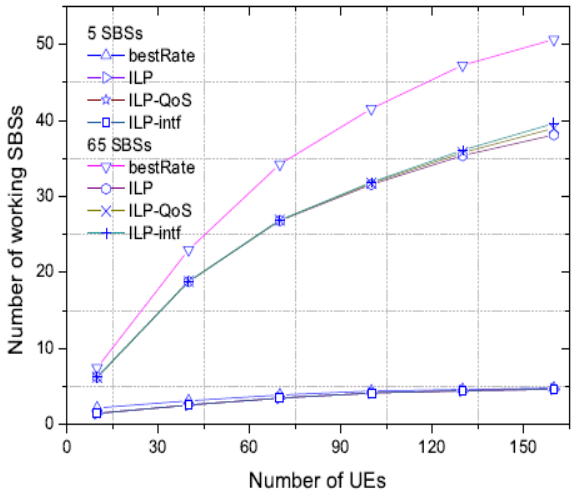


Figure 5-3. The number of working SBSs under different number of UEs in the network. Network setting: 5 and 65 SBSs. Data rate requirement of UEs 2.7Mbit/s

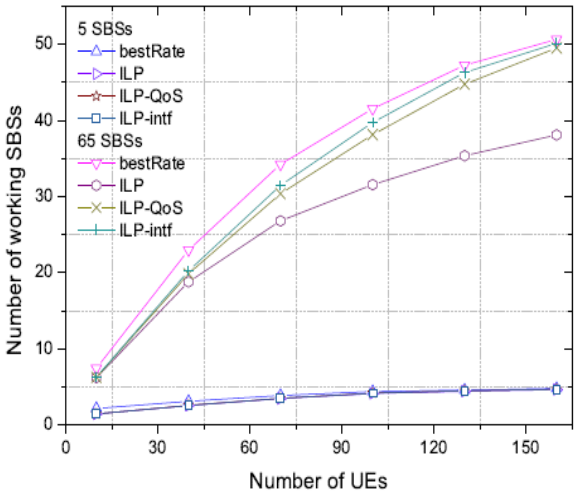


Figure 5-4. The number of working SBSs under different number of UEs in the network. Network setting: 5 and 65 SBSs. Data rate requirement of UEs: 6Mbit/s.

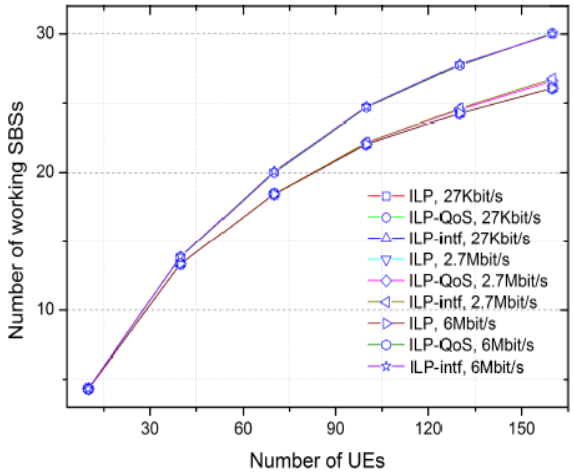


Fig. 5-5. The number of working SBSs under different number of UEs and different data rate requirements in the network. Network setting: 35 SBSs.

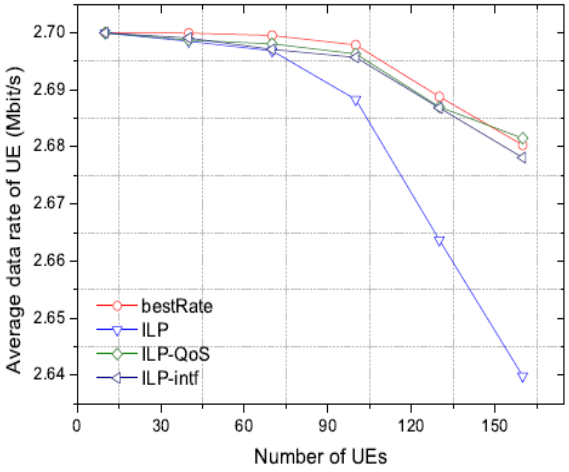


Fig. 5-6. Actual data rate by UEs under different number of UEs in the network. Network setting: 65 SBSs. Data rate requirement of UEs: 2.7 Mbit/s.

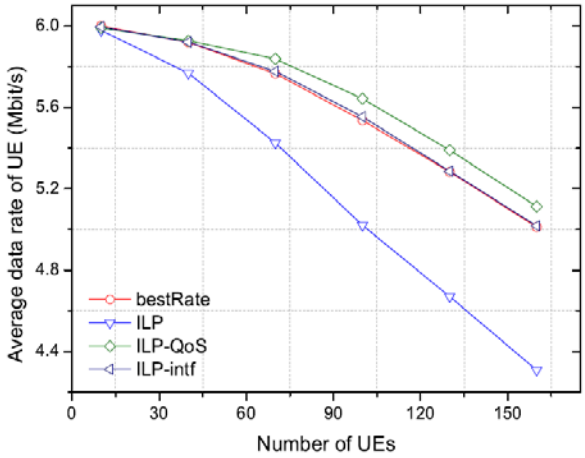


Fig. 5-7. Actual data rate by UEs under different number of UEs in the network. Network setting: 65 SBSs. Data rate requirement of UEs: 6Mbit/s.

Firstly, we compare the number of working BS sets obtained from each algorithm. Note that since each working SBS has the fix power consumption of 10W, the number of working BSs reflects the total power consumption of SBSs. Therefore, we only show the SBS number in the simulation results.

5.6.2 Simulation Results

Fig. 5-3 and 5-4 show the results of bestRate and three developed algorithms under different QoS requirements and different SBS settings. ILP, ILP-QoS and ILP-intf in the figures are the algorithms from the network graph case I-III, respectively. We can see from the 5 SBSs case, the performance of all algorithms in both figures is similar. It is because the UEs have fewer cells in the network to associate. When the SBSs in the network increase to 65, the best result is achieved by the ILP algorithm in both Fig. 5-3 and 5-4. If the QoS requirement is taken into account, since the SBS in Fig. 5-3 has the sufficient capacity to support nearby UEs, ILP, ILP-QoS and ILP-intf have the similar performance. However, when the data rate requirement increases to 6Mbit/s, ILP-QoS and ILP-intf will need more SBSs being switched on as the number of UEs in the network is increasing. But they still out-perform the bestRate algorithm when the data rate requirement of UEs is 6Mbit/s.

The performance of all proposed algorithms under different data rate requirements is compared in Fig. 5-5. When the data rate requirements are small, as in the case of 27kbit/s, all algorithms produced the same results under all network settings. When the data rate requirement is moderate, i.e. 2.7Mbit/s, the differences of all algorithms are very small at the low UE numbers, but increase with the UE numbers. As expected, ILP results in the smallest number of SBSs. ILP QoS outperforms ILP-intf because the interference considered in ILP-intf leads to more active SBSs. When the data rate requirement is as high as 6 Mbit/s both ILP-QoS and ILP-intf need more active SBSs than ILP, and the difference increases quickly as the number of UEs in the network grows. ILP-QoS and ILP-intf have the similar results because, even though no interference is considered, ILP-QoS needs more working SBSs to support the QoS of UEs.

Actual received data rate per UE is shown in Fig. 5-6 and 5-7. That is the data rate taken into account the simple interference model described in Section 5.4. When the number of UEs increases, the actual data rate decreases. In both figures, the ILP algorithm produces the minimal number of working SBSs. But the UEs suffer more inter-cell interference and the actual data rate is compromised. When the data rate requirement is moderate as in Fig. 5-6, bestRate, ILP-QoS and ILP-intf have the similar actual data rate. Under the high data rate requirement as in Fig. 5-7, ILP-intf has the same data rate results as bestRate, because in most cases no optimal solutions will be found for the ILP-intf problem due to the QoS constraints. The UEs have to use the bestRate approach to find their associations. ILP-QoS provides the best actual data rate.

5.7 Conclusions and Future Work

In this section we presented a network graph approach to offer efficient centralized network coordination for large scale heterogeneous mobile networks. Based on the proposed network graph, we study the BS switch on/off problem for network energy saving in small cell networks. We propose a simple inter-cell interference model to simplify the working cell selection problem and formulate the problem as an ILP problem. The simulation study shows the impact of the QoS requirement on the optimal number of cells to serve the network. In the future work, we will consider more realistic interference models and study the joint spectrum allocation and traffic steering for the tradeoff between network capacity and energy efficiency. Finally, we shall also evaluate the feasibility of implementing a selected subset of the results presented in this section using the COHERENT SDK.

6. Conclusions and Impact

6.1 Technical Impact

In spite of their preliminary nature, the results presented in this deliverable have already illustrated several benefits of the COHERENT architecture.

For example, the preliminary results on the CAP trade-offs in software-defined RANs show that the COHERENT architecture can be adapted to scenarios requiring high network resiliency in spite of severe failures and network partitions. Similarly results on RAN sharing and in particular on performance isolation in multi-tenant RANs show the usefulness of the network graph abstraction as input for the performance isolation enabling algorithms. Similar consideration can also be made for the preliminary results on traffic steering and load balancing aimed at optimizing the energy consumption of the RAN while maintaining the expected quality of experience.

The COHERENT architecture allows each network slice to use the resource allocation and resiliency mechanisms that are best suited to their operational environment. For example, there is not a single network resiliency mechanism that will be effective or required in all networking environments, therefore being able to trade-off network resiliency for network complexity and signaling as showed in Sec. 2 using the same architectural framework is a great advantage over the state of the art.

6.2 Feedback Toward Development

The development of the COHERENT C3 Platform and SDK was in its early stage during this reporting period. Nevertheless, some preliminary results especially for WiFi based Enterprise WLANs have been produced providing precious guidelines for future development efforts. The following bullet points summarize the feedback towards the COHERENT SDK development:

- In order to support the proposed solutions on network resiliency, the COHERENT platform and SDK, as well as the reference WiFi datapath implementation must be extended with support for the Consensus Controller and leader election mechanism. Development efforts have already been started in this direction.
- The validation of the proposed solutions regarding network resiliency will require a suitable network emulation/simulation environment in order to conduct tests on a network with a significant number of nodes. The extension of mininet and mininet-WiFi to this scenario is currently being investigated.
- In both the traffic steering and in the RAN sharing scenario evidence has emerged toward the need for some interaction between radio domain and backhaul. This is because radio resource allocation operation performed in the RAN affects the state of the backhaul and vice versa. The use of an intent-based interface between the C3 Entity and a standard backhaul controller is currently under investigation with a preliminary prototype already tested in a small lab setup.

6.3 Expected Business Impact

Next generation mobile networks will have increasing demands placed on them for greater throughput, lower latency, improved coverage and capacity with reduced operational cost compared to current networks. Two main technologies presented in this document, and analyzed in WP5, are: Network Function Virtualization and RAN Sharing, which promise clear economic advantages when deployed in heterogeneous networks compared to today's mobile networks. Each technology addresses specific challenges or objectives brought by 5G, as summarized in Table 6-1.

	Scalability	Operational Cost	Latency	Coverage/ Capacity	Business Model Flexibility
Network Function Virtualization	+++	+	+	o	+++
RAN Sharing	+	+++	o	++	+

Table 6-1: Advantages compared to current network architectures

For both of these technologies, the economic impact can be roughly analysed at each phase: development, deployment and operation.

6.3.1 Network Function Virtualization

As stated in Section 3, "Network Function Virtualization (NFV) promises to reduce the cost to deploy and operate large networks by migrating network functions from dedicated hardware appliances to software instances running on general purpose virtualized networking and computing infrastructures." These VNFs can be placed in any location of the network from the data center to the network node to the equipment which is on the customer side. In this way the service offered to the end user can have better performance and the CAPEX and OPEX costs for the operator to be decreased. The intention is not only to distribute existing network functions but to enable new services that were either not possible or too expensive to develop/deploy before. New verticals can be created to generate new revenue sources.

Development: As NFV is intended to be deployed on general computing infrastructures, it eliminates the need for customized hardware development, which reduces not only development efforts but also Bill of Materials during deployment. Development efforts shift to software activities, which are intensified due to increased complexity. However, equipment vendors can utilize software development teams in lower cost regions to minimize development costs.

Deployment: New features and services can be deployed with significantly lower cost and effort. Practically an infinite number of verticals can be addressed by operators to generate new services and the corresponding revenue streams.

Operation: Operational costs could be slightly increased due to a larger number of services and verticals being supported.

An additional advantage is increased rate of innovation and competition. With a focus on software development based on off-the-shelf platforms, the barrier to creating new features and services is lowered. This not only encourages new entrants as competitors but also spurs innovation and new ideas.

6.3.2 RAN Sharing

The CAPEX for deployment of necessary network upgrades to meet 5G requirements will be enormous. In addition, roughly 40-50% of cell sites achieve low or negative return on an operator's investment. There are very few network operators in the world with the resources and access technology experience to ultimately deploy extensive 5G networks and later remain profitable with additionally increased OPEX. The next evolutionary step, RAN sharing, proposes to offer cost efficiencies over traditional schemes. The main benefit from RAN sharing is driven by the need to maximize the operator value and at the same time it will be the reduction in CAPEX and OPEX. The RAN sharing enables dynamic allocation and sharing of an access network, avoiding operators deploying multiple physical instances of RAN in the same area.

The RAN sharing framework proposed by COHERENT addresses a multi-network operator-sharing environment with time-critical SLAs with the aim of ensuring maximum revenue for the infrastructure owner and QoE for the customers of the time-critical services.

Development: Little efficiency is gained during development. Use of off-the-shelf platforms minimize hardware development costs, or completely eliminate them. However, software complexity is increased as well as validation efforts. Equipment vendors will face the challenge of supporting multiple methods of infrastructure sharing.

Deployment: As a general concept, RAN sharing has a huge impact on the cost of network deployment and is essential to the financial viability of 5G. Operators will face choices of 1) which partners, 2) what type of sharing, 3) where to share and 4) how to share.

Operation: in the operational phase, SLA violations will be minimized/eliminated and OPEX savings up to 30% can be gained. Valuable spectrum can also be more efficiently utilized. The concepts explored in COHERENT will potentially allow new and more real time critical services to be financially viable.

6.4 Next Steps

The work towards the final version of the algorithms for programmable RAN will focus on two main axes. On one hand, the research will continue to devise the COHERENT concept and further validate its usefulness. On the other hand, we intend to combine several research areas, designing and executing more complex experiments that help validate the SDK and the COHERENT concept at a larger scale. The objective here is to move from a situation where single partners carried out (high-impact) research activities to a situation where cross-research area solutions are jointly developed and tested. These experiments will provide opportunities for cross-fertilization between academic and industrial research as well as precious feedback towards the development.

As far as economic aspects are concerned, the preliminary analysis will be extended to match the larger scope introduced by the more complex experiments combining different research areas and providing a more holistic approach from the perspective of the presented industry use cases. The business impact exercise will provide a feasibility analysis for initial COHERENT adoption in a larger scale, as well as a potential roadmap for larger penetration of the technology towards existing and new business stakeholders, such as datacenter operators, network service providers, cloud infrastructure/software providers, large enterprise, etc. This will take into account not only the potential benefits and relative advantage, but also compatibility and complexity considerations with existing (e.g. previous CAPEX investments) and developing technology (e.g. adoption paths for SDN, NFV, etc.)

Bibliography

[Andrews2012] J. G. Andrews, H. Claussen, M. Dohler, S. Rangan, and M. C. Reed, "Femtocells: Past, Present, and Future," Selected Areas in Communications, IEEE Journal on, vol. 30, no. 3, pp. 497-508, 2012

[Blenk2016] A. Blenk, A. Basta, M. Reisslein and W. Kellerer, "Survey on Network Virtualization Hypervisors for Software Defined Networking," in IEEE Communications Surveys & Tutorials, vol. 18, no. 1, pp. 655-685, Firstquarter 2016.

[Bote2013] Botelho, Fábio Andrade, Fernando Manuel Valente Ramos, Diego Kreutz, and Alysson Neves Bessani. "On the feasibility of a consistent and fault-tolerant data store for SDNs." In 2013 Second European Workshop on Software Defined Networks, pp. 38-43. IEEE, 2013.

[Bote2014] Botelho, Fábio, Alysson Bessani, Fernando MV Ramos, and Paulo Ferreira. "On the design of practical fault-tolerant SDN controllers." In 2014 Third European Workshop on Software Defined Networks, pp. 73-78. IEEE, 2014

[Chandra1996] T. D. Chandra and S. Toueg, "Unreliable failure detectors forreliable distributed systems," J. ACM, vol. 43, no. 2, pp. 225-267, Mar. 1996.

[Chen2014] T. Chen, H. Zhang, X. Chen, and O. Tirkkonen, "SoftMobile: Control Evolution for Next Generation Heterogeneous Mobile Networks," Wireless Communications, IEEE, vol. 21, no. 6, pp. 70-78, 2014.

[Chen2016] Tao Chen,Xianfu Chen, Roberto Riggio, "A Network Graph Approach for Network Energy Saving in Small Cell Networks", in Proc. of IEEE VTC Spring 2016

[Chowdhury2009] M. Chowdhury, M. R. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," IEEE/ACM Trans. Netw., vol. 20, no. 1, pp. 206-219, Feb. 2012.

[Cichon98] D. Cichon and T. Kurner, "EURO-COST 231 Final Report," Tech. Rep, Tech. Rep., 1998.

[Desai] Desai, A. and Zheng, W., Building Reliable and Performant Software Defined Networks. https://people.eecs.berkeley.edu/~kubitron/courses/cs262a-F14/projects/reports/project16_report.pdf

[Dulmage58] A. L. Dulmage and N. S. Mendelsohn, "Coverings of Bipartite Graphs," Canadian Journal of Mathematics, vol. 10, no. 4, pp. 516-534, 1958.

[ETSI] ETSI, "Mobile Edge Computing (MEC): Framework and Reference Architecture," GS MEC 003 V1.1.1, 2016.

[ETSINFV] European Telecommunications Standards Institute (ETSI), "ETSI GS NFV 002 network functions virtualisation (NFV); Architectural framework," Dec. 2014.

[Fischer13] A. Fischer, J. Botero, M. Till Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," IEEE Commun. Surveys Tuts., vol. 15, no. 4, pp. 1888-1906, Apr. 2013.

[Fontes2015] R. Fontes, S. Afzal, S. Brito, M. Santos, and C. Rothenberg, "Mininetwifi: Emulating software-defined wireless networks," in 2015 11th International Conference on Network and Service Management (CNSM).

[Fontes2015] R. Fontes, S. Afzal, S. Brito, M. Santos, and C. Rothenberg, "Mininetwifi: Emulating software-defined wireless networks," in Network and Service Management (CNSM), 2015 11th International Conference on, Nov 2015.

[FV2009] FlowVisor: A Network Virtualization Layer, Open Networking Foundation Technical Report, OPENFLOW-TR-2009-1

[Garg14] S. K. Garg, A. N. Toosi, S. K. Gopalaiyengar, and R. Buyya, "Sla-based virtual machine management for heterogeneous workloads in a cloud datacenter," Journal of Network and Computer Applications, vol. 45, pp. 108-120, 2014.

[Geo2006] L. Georgiadis, M. J. Neely, and L. Tassiulas, Resource allocation and cross-layer control in wireless networks. Now Publishers Inc, 2006.

[Ghaznavi15] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic virtual network function placement," in Proc. IEEE CloudNet, pp. 255-260, 2014.

[Gilbert2002] S. Gilbert and N. Lynch, "Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web," MIT, Berkeley, Tech. Rep., Jun. 2002.

[Gudipati2013] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "Softran: Software defined radio access network," in Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN), ACM, 2013

[Gudipati2014] A. Gudipati, L. E. Li, and S. Katti, "Radiovisor: A slicing plane for radio access networks," in Proc. ACM Workshop on Hot Topics in Software Defined Networking, 2014, pp. 237-238.

[Heusse2003] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11b," in Proc. IEEE INFOCOM, San Francisco, CA, USA, 2003, pp. 836-843.

[Ho2014] C. K. Ho, D. Yuan, and S. Sun, "Data Of?oading in Load Coupled Networks: A Utility Maximization Framework," IEEE Transactions on Wireless Communications, vol. 13, no. 4, pp. 1921-1931, Apr. 2014.

[Imran2011] M. Imran and E. Katranaras, "ICT-EARTH Project, Deliverable D2. 3, EC-IST Of?ce, Brussels, Belgium (January 2011)."

[Jennings15] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," J. Netw. Syst. Manage., vol. 23, no. 3, pp. 567-619, Jul. 2015.

[Jin12] H. Jin, D. Pan, J. Xu, and N. Pissinou, "Efficient VM placement with multiple deterministic and stochastic resources in data centers," in IEEE GLOBECOM, 2012.

[Kar1984]]N. Karmarkar, "A new polynomial time algorithm for linear programming," Combinatorica, vol. 4, no. 4, pp. 373-395, 1984.

[Katsalis2016] Kostas Katsalis, "SLA-driven VM Scheduling in Mobile Edge Computing", in Proc. of IEEE INFOCOM 2016

[Katta2015] Katta, N., Zhang, H., Freedman, M. and Rexford, J., 2015, June. Ravana: Controller fault-tolerance in software-defined networking. In Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research (p. 4). ACM.

[Katti2014] S. Katti and L. Li, "RadioVisor: A slicing plane for radio access networks," in Proc. USENIX Open Netw. Summit (ONS), 2014, pp. 237-238.

[Kohler2000] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. 2000. The click modular router. ACM Trans. Comput. Syst. 18, 3 (August 2000), 263-297. DOI=http://dx.doi.org/10.1145/354871.354874

[Kreutz2013] Kreutz, D., Ramos, F. and Verissimo, P., 2013, August. Towards secure and dependable software-defined networks. In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (pp. 55-60). ACM.

[Kshemk2011] A. D. Kshemkalyani and M. Singhal, "Failure detectors," in Distributed Computing Principles, Algorithms, and Systems. Cambridge, Mar. 2011, vol. March 2011, pp. 567-577.

[Logcabin] D. Ongaro, "Logcabin." [Online]. Available: https://github.com/logcabin/logcabin

[Mininet] "Mininet network emulator." [Online]. Available: http://mininet.org/

[Moens14] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in Proc. IEEE 10th Int. Conf. Netw. Serv. Manage. (CNSM), 2014.

[Neely10] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," Synthesis Lectures on Commu- nication Networks, vol. 3, no. 1, pp. 1-211, 2010.

[NGMN] NGMN Alliance, "5G white paper," Next Generation Mobile Networks, White paper, 2015.

[Ongaro2014] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference (ATC). Berkeley, CA, USA: USENIX Association, 2014 pp 10-2.

[OpenBaton] Fraunhofer FOKUS. OpenBATON [Online]. Available: http://openbaton.github.io/

[Panda2013] A. Panda, C. Scott, A. Ghodsi, T. Koponen, and S. Shenker, "Cap for networks," in Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN), ACM, 2013

[Patel14] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal et al., "Mobile-Edge Computing Introductory Technical White Paper," White Paper, Mobile-edge Computing (MEC) industry initiative, 2014.

[Riggio2015A] Roberto Riggio, Abbas Bradai, Tinku Rasheed, Julius Schulz-Zander, Slawomir Kuklinski, Toufik Ahmed, "Virtual Network Functions Orchestration in Wireless Networks", in Proc. of IEEE CNSM 2015, Barcelona, Spain.

[Riggio2015B] R. Riggio, M. K. Marina, J. Schulz-Zander, S. Kuklinski and T. Rasheed, "Programming Abstractions for Software-Defined Wireless Networks," in IEEE Transactions on Network and Service Management, vol. 12, no. 2, pp. 146-162, June 2015.

[Riggio2016] Roberto Riggio, Abbas Bradai, Davit Harutyunyan, Tinku Rasheed, Toufik Ahmed, "Scheduling Wireless Virtual Network Functions", in IEEE Transactions on Network and Service Management May 2016 DOI: 10.1109/TNSM.2016.2549563

[Ryu] Ryu SDN Framework [Online]. Available: http://osrg.github.io/ryu/

[Shen11] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: Elastic resource scaling for multi-tenant cloud systems," in ACM SOCC, 2011.

[Singh2014] S. Singh and J. G. Andrews, "Joint Resource Partitioning and Of?oading in Heterogeneous Cellular Networks," Wireless Communications, IEEE Transactions on, vol. 13, no. 2, pp. 888-901, 2014.

[Urgaonkar10] R. Urgaonkar, U. C. Kozat, K. Igarashi, and M. J. Neely, "Dynamic resource allocation and power management in virtualized data centers," in IEEE NOMS, 2010.

[Xia2010] B. Xia and Z. Tan, "Tighter bounds of the first fit algorithm for the bin-packing problem," Discrete Applied Mathematics, vol. 158, no. 15, pp. 1668-1675, 2010.

[Yu2008] Y. Minlan, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 17-29, Mar. 2008.

[Zhu2006] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in Proc. IEEE INFOCOM, Barcelona, Spain, Apr. 2006, pp. 1-12.