UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Facolta' di Ingegneria

Corso di Dottorato di Ricerca in Ingegneria Informatica ed Automatica
XXIV Ciclo
Dipartimento di Informatica e Sistemistica

DESIGN, ANALYSIS AND EVALUATION OF ROUTING AND CHANNEL ASSIGNMENT ALGORITHMS FOR WIRELESS MESH NETWORKS

GIOVANNI DI STASI

Ph.D. Thesis

TUTOR
Prof. Roberto Canonico

COTUTOR
Prof. Stefano Avallone

COORDINATOR
Prof. Francesco Garofalo

November 2011

In this thesis we describe the efforts we made for designing, developing and analyzing new routing and channel assignment algorithms for wireless mesh networks. In particular we discuss about a new channel re-assignment scheme, able to minimize the number of changes required to accomodate new traffic demands, a new MPLS-based forwarding paradigm, resistant to failures and able to cope with variations of the traffic demands and, finally, a new forwarding paradigm able to exploit aggregation opportunities at the MAC layer.

Before going into the details of these new algorithms, we introduce some of the networking testbeds, platforms and tools we have leveraged for their evaluation. We also describe a contribution we have made in the field of heterougeneous networking testbeds, in particular regarding the integration of local wireless mesh testbeds in PlanetLab.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wireless Mesh Networks (WMNs) are becoming increasingly popular for providing Internet connectivity in wide or difficult to reach areas, thanks to their low deployment and maintenance costs [1].

A WMN is comprised of a set of mesh routers capable of wirelessly delivering, through multiple hops, packets either destined or originated from wireless clients. Wireless clients can connect to the WMN through special mesh routers, denoted as *mesh aggregation devices*, which have, in addition to routing capabilities, also access point capabilities. WMNs are usually capable to connect to other networks (e.g. a wired backbone) through mesh routers with bridge capabilities, denoted as *mesh gateways*. Mesh routers are usually fixed and therefore do not have strong requirements in terms of energy consumption. The architecture described is commonly known as *infrastructure/bachbone mesh* (see Fig. 1.1). Other architectures exist and are described in chapter 4.

Mesh routers are usually equipped with multiple radios that allow them to transmit simultaneously on different channels, i.e. range of frequencies, therefore reducing interference.

The availability of multiple radios per node leads to the channel assignment problem, i.e., the problem how to select a channel for each radio in the network. Such a channel can be changed at the packet level or can be used for a relatively long time, e.g. tens of minutes. The change of channel at the packet level, or with a high frequency, requires support from the network hardware, which is not available in current off-the-shelf IEEE 802.11 tecnology.

The problem of routing is related to the way packets are delivered between any two nodes of the network. As bandwidth is a limited resource due to interference, routing should be able to distribute traffic in order not to exceed the available bandwidth on each link.

In this thesis we describe the efforts which have led to the definition of a new channel re-assignment algorithm and two novel forwarding paradigms for wireless mesh networks able to solve some issues of previous state-of-the-art algorithms.

The new channel re-assignment algorithm performs, in case of variations of the traffic demands, the change of the channel of a subset of radios taking into ac-

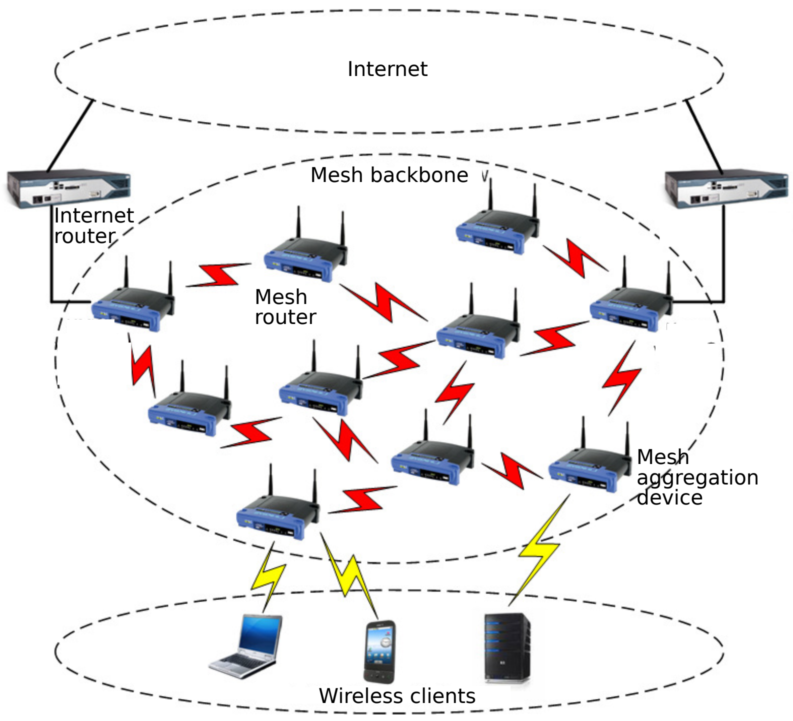


Figure 1.1: A wireless mesh network.

count the current channel allocation. We show in this thesis that changing channel on a high number of radios can lead to a reduced network throughput for a relatively long time, i.e. tens of seconds. The proposed algorithm is able to keep the number of required channel changes low, while still achieving good performances and allowing to reduce the time required to re-establish connectivity after the channel re-assignment process.

The first forwarding paradigm, based on MPLS [2], allows a fast recovery from node failures and has low dependency on variations of the traffic demands.

The second forwarding paradigm combines routing and packet aggregation trying to exploit aggregation opportunities at the MAC layer. It leverages most of the assumptions and ideas of the MPLS-based routing paradigm but in addition puts packet aggregation into the picture.

Before introducing in details the aforementioned algorithms, we introduce some of the networking testbeds, platforms and tools we have leveraged to evaluate them. We also describe a contribution we made in the field of the heterogeneous networking testbeds, which allows to make experiments with wireless mesh networks in heterogeneous environments. Heterogeneous networking testbeds are considered more and more important, as they allow to test new protocols and networking systems in an environment which is more representative of real networks. The contribution consists in adding heterogeneity to the PlanetLab [3] platform, in particular the ability to use a wireless mesh testbed as access network for PlanetLab.

The thesis is organized as follows. In chapter 2 we describe the main networking testbeds, platforms and methodologies we have leveraged for the evaluation of the proposed algorithms. In chapter 3 we goes into details of one of the used platforms and describe the contribution related to the integration of wireless mesh networks in PlanetLab.

In chapter 4 we describe the model of wireless network we assume and give some basic definitions that will be used later for the description of the channel re-assignment and forwarding paradigms we propose.

In chapter 5 we introduce the new channel re-assignment algorithm.

In chapters 6 and 7 we introduce the new forwarding paradigms.

Finally in chapter 8 we draw the conclusions.

Chapter 2

Methodologies and tools for performance assessment of wireless mesh networks

2.1 Introduction

It is of utmost importance to properly design and carry on the evaluation process of new designed algorithms. For this reason the reasearcher must master the most important methodologies and tools in order to properly apply them in the different phases of the design and evaluation processes. In this chapter we introduce some of the methodologies and tools we have relied on for the evaluation of the routing and channel assignment algorithms proposed in this thesis. In particular we follow a common taxonomy which leads to the separation of the tools and related methodologies in three different cathegories: simulation, emulation and testbed based. We discuss the pros and cons of each category and give some details on some specific tools (e.g. ns-2). In addition to the tools belonging to the aforementioned categories, we also describe an hybrid approach which allows to perform the initial tests in a simulated scenario and them move to testbed based experimentation with very little efforts.

2.2 Network simulation

Network simulation is a common tecnique used to test routing and channel assignment algorithms for wireless mesh networks. The main advantage of this tecnique is related to the high controllability it guarantees during experiments and to the possibility of rapidly testing new design alternatives.

A drawback of this tecnique is the scarse reliability of results, due to the use of approximated models.

In the following subsections we briefly describe the two tools we used for part of the evaluation of the proposed algorithms.

2.2.1 Network Simulator 2

The Network Simulator 2 [4] (Ns2) is a discrete-event network simulator targeted at networking researches. It provides support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. Ns-2 started as a variant of the REAL network simulator in 1989 and has evolved substantially from there, thanks to the contribution of different research institutes (e.g. DARPA, Xerox PARC, UCB). Nowadays, ns-2 is not receiving substantial new contributions, since the development efforts are more concentrating on ns-3, a new network simulator written from scratch to replace ns-2 (known to have some limitations, especially regarding its extendability).

We used ns-2 to test the new aggregation-aware routing algorithm, for which we already had an implementation of the packet aggregation module.

2.2.2 Network Simulator 3

Ns-3 is a new network simulator written from scratch by the networking community to overcome some of the weaknesses of the old ns-2 simulator. In particular, it tries to be more modular so as to be more easily extendable and more easily interconnected with real systems[1]. A lot of design choices for ns-3 have been made in the direction of having components more similar to the real counterparts (e.g. the routing stack offers to applications a socket-like API and talks to the net devices through a Linux-like packet socket interface[5]).

Ns-3 is particularly interesting for new research projects, when all the code is written from scratch. The reason is related to the greater facility in developing and integrating new elements, thanks to ns-3 modularity; in contrast to that, ns-2 for the same goal usually requires changes to its core parts.

Ns-3 is written purely in C++, with binding available in Python. It is possible, therefore, to write pure C++ applications which use ns-3 components, while in ns-2 it was always necessary to use the Otcl[6] interpreter.

Ns-3, on the other hand, has not yet availability of all the models and components available in ns-2.

An area where ns-3 is strong is simulation of IEEE 802.11 wireless networks, thanks to the advanced models it employes. Similar models are available in ns-2, but often in the form of external patches that need to be applied and sometimes generate conflicts with other patches.

[1]Working in such a case as emulator.

We used ns-3 to test the new MPLS-based routing algorithm and the new channel re-assignment algorithm.

2.3 Network emulation

With network emulation real components become part of the system under test. This allows to perform experiments on a system which encompasses also real components, therefore giving more reliable results.

Emulation can be considered as a step forward the evaluation of the real system. Among the plethora of emulation systems which are available, we evaluated and used *VirtualMesh*, described in the following section.

2.3.1 VirtualMesh

VirtualMesh is an emulation system specifically tailored for wireless mesh networks. It allows to create an experimental scenario where the *datalink* and *physical* layers are emulated and the remaining layers are real (real applications and a real IP network stack can be used).

In Fig. 2.1 the architecture of the system is reported. The emulated system comprises real nodes and an emulator engine based on OMNeT++ [7]. Each node is univocally associated to a counterpart in a model of network run by the simulator (working as emulator). The model specifies the position of nodes, the settings of the WiFI interfaces [2], the characteristics of the wireless channel, and so on.

Nodes are connected to the simulator through an UDP connection which is used to send data and control packets. Data packets are generated by the applications or by the network stack of nodes (e.g. IP packets, ARP packets, ...). Such packets are first sent to the OMNeT++ simulator that decides to which nodes delivering them, depending on the model of network run in the simulator. Control packets are used to communicate changes to the model triggered by the nodes (e.g. a new channel used by a WiFI interface).

[2]WiFI interfaces are also generically referred to as radios in this thesis.

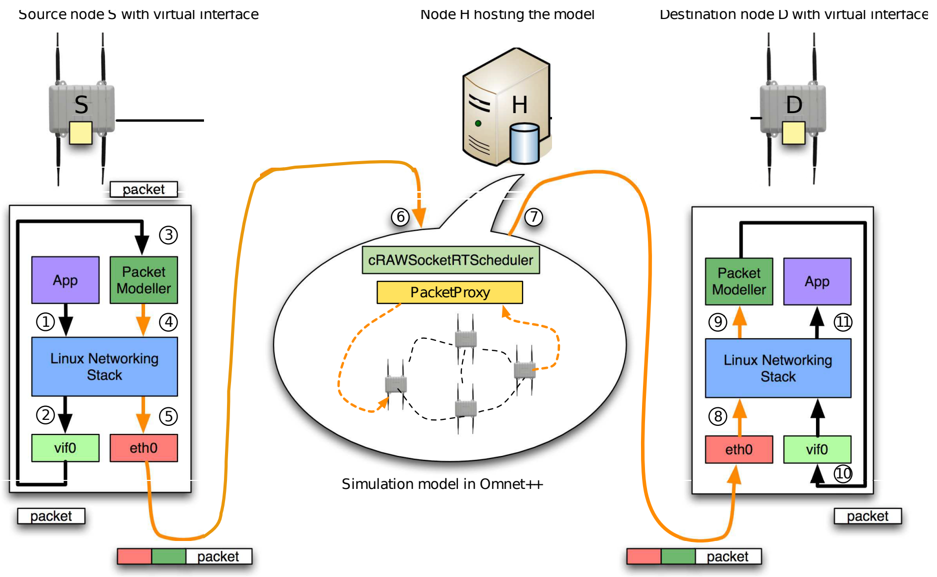


Figure 2.1: Packet flow between two nodes interconnected by the OMNeT++ simulation model.

2.4 Experimental facilities

Experimental facilities, also known as testbeds, are being more and more used to test new routing and channel assignment algorithms for wireless mesh network because of the high reliability of the results they can provide. Several testbeds are available nowadays to the community of experimenters. Among them, we will describe Orbit, an heterogeneous testbed that allows to perform both wired and wireless based experiments, and PlanetLab, a geographically distributed large-scale testbed used for deploying and evaluating planetary-scale network applications in a highly realistic context.

2.4.1 Orbit radio grid testbed

The Orbit [8],[9] radio grid testbed is a large-scale mixed wireless/wired testbed operated by the University of Rutgers (USA). It consists of four hundred nodes, organized in a 20x20 grid, each of which is equipped with one or two ethernet interfaces (connected to a common bridge) and two WiFI interfaces (see Fig. 2.2). We leveraged the Orbit testbed to test the new channel re-assignment algorithm for wireless mesh networks. Orbit resources are managed by OMF (cOntrol and Management Framework), a software platform that allows the automatic execution of experiments and the automatic collection of results.

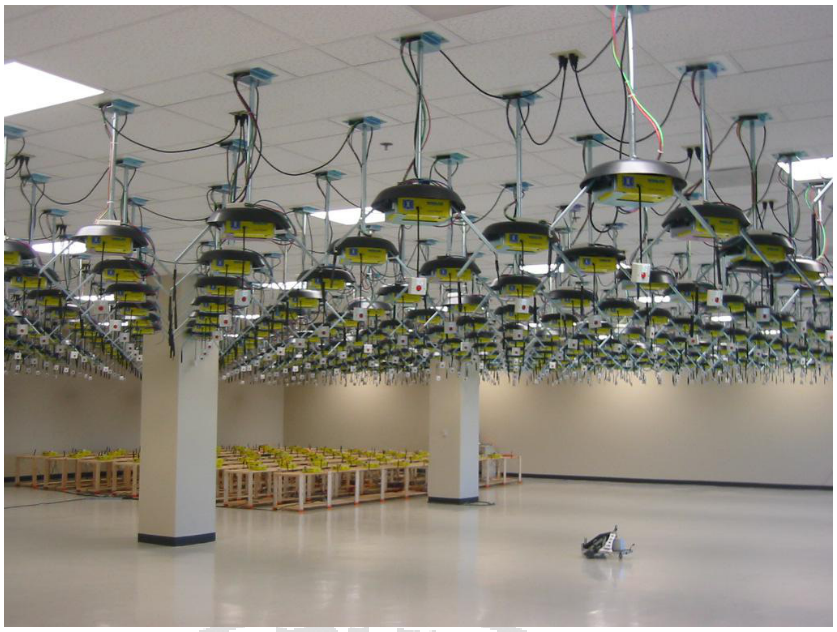


Figure 2.2: ORBIT radio grid testbed.

OMF: cOntrol and Management Framework

OMF (*cOntrol and Management Framework*) is a Testbed Control, Measurement and Management Framework. It was originally developed at the University of Rutgers, New Jersey, as the management tool for ORBIT, while now is mainly being developed by NICTA [3]. An important companion library of OMF is OML (*OMF Measurement Library*), which is used to automatically filter and collect experiment data on one or more measurement servers. OMF has been very useful to automatize and allow the repeatability of the experiments we performed to test the behavior of the channel assignment algorithm proposed in this thesis (described in Chapter 5).

[3]http://www.nicta.com.au/

OMF Architecture

The components of OMF (Fig. 2.3) work together to automatically perform all the phases needed to execute the experiment, from the provisioning of resources to the collection of experimental data. The most important component is the *Experiment Controller* (EC), which is also the interface to the user. It accepts as input an experiment description and takes care of orchestrating the testbed resources in order to accomplish the required experiment steps. It interacts with the *AggregateManager*, the entity responsible of the resources of the testbed as a whole, and provides some basic services to the EC, such as checking the status of a node, rebooting a node, etc.

The EC also interacts with the *Resource Controllers* (RCs) installed on the testbed nodes. These latter entities are responsible of performing local configuration steps, e.g. configuring the channels on the Wi-FI interfaces, and of controlling the applications, e.g. the traffic generator. The communication between the EC and the RCs is based on a publish/subscribe paradigm, where the EC publishes the messages on a XMPP server [10] and the RCs pick the messages addressed to them.

An important companion library of OMF is OML (*OMF Measurement Library*), which is used to automatically filter and collect experiment data on one or more measurement servers. OMF is able to instrument the OML library, in order to configure and guide the collection of experiment data.

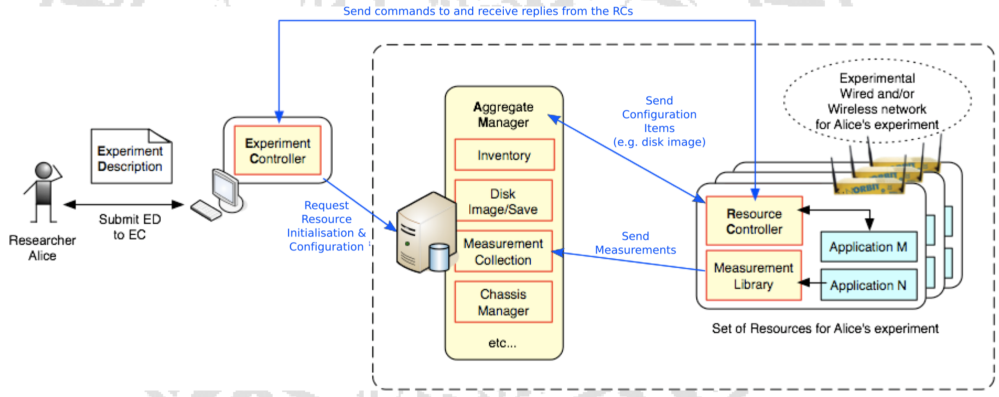

Send commands to and receive replies from the RCs
Researcher Alice
Experiment Description
Submit ED to EC
Experiment Controller
Request Resource Initialisation & Configuration
Aggregate Manager
Inventory
Disk Image/Save
Measurement Collection
Chassis Manager
etc...
Send Configuration Items (e.g. disk image)
Send Measurements
Experimental Wired and/or Wireless network for Alice's experiment
Resource Controller
Measurement Library
Application M
Application N
Set of Resources for Alice's experiment

Figure 2.3: OMF architecture overview.

OMF Usage model

In order to perform an experiment, users have to log into the *testbed console*, i.e. the host running the Experiment Controller (EC). The execution of an experiment can be requested to the EC by submitting an experiment description in the domain-specific OEDL language, which is derived from Ruby. The experiment description usually consists of two parts: i) a first declarative part, comprising a list of required resources and applications, with their configuration; ii) a second part, describing the set of actions to be performed in order to realize the experiment. The execution of specific actions may depend on events which are defined by the platform, e.g. all the nodes are up and running.

OMF Resource management

OMF, in its basic form, assigns resources to users following a FCFS strategy: the user supplies an experiment description and the system tries to assign the resources requested by the experiment if they are available.

OMF can be customized, though, to support some kind of reservation of resources. In ORBIT a Scheduler interface is provided to support the reservation of the entire testbed. The user books the testbed in advance and during the reserved time slot he/she is the only one allowed to log into the testbed console and run his/her own experiments.

2.4.2 PlanetLab

PlanetLab is considered by the research community as one of the most relevant large scale distributed testbed for networking research [3]. PlanetLab is a geographically distributed testbed for deploying and evaluating planetary-scale network applications in a highly realistic context. Nowadays the testbed is composed of more than 1000 computers, hosted by about 500 academic institutions and industrial research laboratories. One of the main limitations of PlanetLab, however, is its lack of heterogeneity. Nearly all PlanetLab nodes are server-class computers connected to the Internet through high-speed wired research or corporate networks. As a consequence, it has also been noted that the behavior of some applications on PlanetLab can be considerably different from that on the Internet [11], [12]. Several efforts have been done in the last few years to add different kinds of networking technologies to PlanetLab (e.g. UMTS integration in PlanetLab is described in [13]) or to integrate new kind of terminals (e.g. the integration of non-dedicated devices made available by residential users is described in [14]). However, it is now clear that PlanetLab can be usefully complemented by a variety of other testbeds, in particular when experimentation with wireless technologies is required. We describe in chapter 3 how we integrate local OMF-based wireless testbeds in PlanetLab, which allows to evaluate new algorithms for WMNs in a large and heterogeneous scenario.

Architecture

Figure 2.4 shows a conceptual view of the current architecture of the PlanetLab testbed, whose node set is the union of disjoint subsets, each of which is managed by a separate authority. As of today, two such authorities exist: one is located at Princeton University (PLC) and the other is located at Université Pierre et Marie Curie UPMC in Paris, France (PLE). An experiment in PlanetLab is associated to a so-called *slice*, i.e. a collection of virtual machines (VMs) instantiated on a defined subset of all the testbed nodes. Each testbed authority hosts an entity called *Slice Authority* (SA), which maintains state for the set of system-wide slices for which it is responsible. The slice authority includes a database that records the persistent state of each registered slice, including information about every user that has access to the slice [15].

Testbed authorities also include a so called *Management Authority* (MA), which is responsible of installing and managing the updates of software running on the nodes it manages. It also monitors these nodes for correct behavior, and takes appropriate action when anomalies and failures are detected. The MA maintains a database of registered nodes at each site. Each node is affiliated with an organization (owner) and is located at a site belonging to the organization.

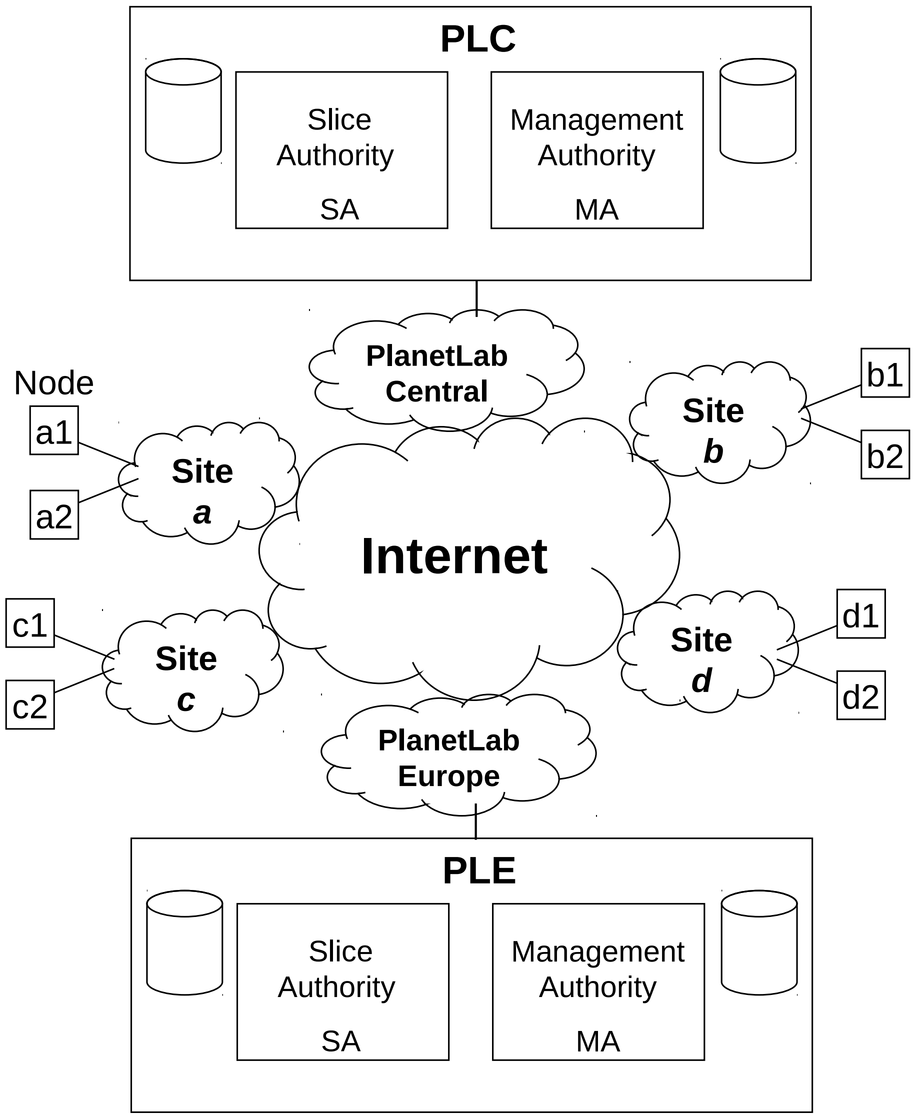


Figure 2.4: Conceptual PlanetLab architecture.

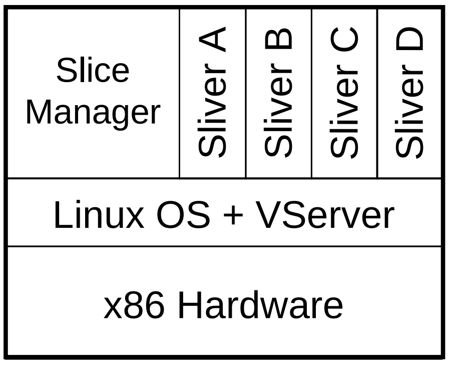


Figure 2.5: Internal view of a PlanetLab node.

Usage model

To run a distributed experiment over PlanetLab, users need to be associated with a slice. Slices run concurrently on PlanetLab, acting as network-wide containers that isolate services from each other. An instantiation of a slice in a particular node is called a *sliver*. Slivers are Virtual Machines created in a Linux-based environment by means of the VServer virtualization technology. By means of so-called *contexts*, VServer hides all processes outside of a given scope, and prohibits any unwanted interaction between a process inside a context and all the processes belonging to other contexts. VServer is able to isolate services with respect to the filesystem, memory, CPU and bandwidth. However, it does not provide complete virtualization of the networking stack since all slivers in a node share the same IP address and port space. The adoption of VServer in PlanetLab is mainly motivated by the need of scalability, since up to hundreds of slivers may need to be instantiated on the same physical server [16]. Figure 2.5 shows the internal view of a PlanetLab node.

PlanetLab resource management

In PlanetLab, slice creation and resource allocation are decoupled. When a slice is first created, a best effort service is associated with it and resources are acquired and released by the slice during its entire lifetime. Therefore, by default, slices are not bound to sets of guaranteed resources. Such an approach has been deliberately chosen in the original PlanetLab design. PlanetLab, in fact, has not been designed for controlled experiments, but to test services in real world conditions [17], [18]. After its initial development, PlanetLab has been extended with a calendar service, called SIRIUS, whose purpose is to allow users to obtain a "better service" from all the nodes participating to a given slice. In practical terms, this means that, during a reserved time slot, a slice may be granted 25% of each processor's CPU capacity, and 2 Mbps of link bandwidth. The actual usage of SIRIUS by PlanetLab users is quite modest, since it does not allow precise control over the reservable resources.

2.5 An hybrid approacch: Ns-3 plus Click!

An hybrid approach based on simulation we would like to mention is related to the integration of Click Modular Router [19] and Ns-3. Click Modular Router (abbreviated in Click!) is a framework that allows to construct software routers by exploiting elements of a vast library (that implements common operations, like decrementing the TTL of an IP packet) with custom made elements (to implement a particular behavior). The behaviour of a Click! router can be defined by a graph whose vertices are the elements and whose links represents the flow of packets.

When combined with ns-3, Click allows to run the software router in a simulated environment, therefore exploiting the advantages of simulation-based experiments (high controllability, easy test of design alternatives, etc.), and at the same time, on commodity hardware, thus allowing the making of experiments on real networks.

Chapter 3

Heterogeneous networking testbeds

3.1 Introduction

It is important for the wireless mesh network paradigm to have success, to be able to test it in real world scenarios. Due to the inherent difficulty of capturing all the relevant aspects of the real behavior of these systems in analytical or simulation models, research on wireless mesh networks (WMNs) has always heavily relied on experimental testbeds. In fact, the creation of such experimental testbeds has been an active area of research in wireless mesh networking over the last ten years [20]. To allow for a realistic evaluation of new applications, services and algorithms specifically designed for wireless mesh networks, we analyzed the existing projects that enable to share and manage testbeds and resources over a large geographic area. On the one hand, PlanetLab is universally known to be an open platform to conduct realistic experiments on a planetary scale [3]. On the other hand, OMF (*cOntrol and Management Framework*) is a well-established software platform that supports the management and automatic execution of experiments on a networking testbed [21].

In this chapter we present a contribution made in collaboration with some researchers of the COMICS [22] group at the University of Naples and some researchers of the NITLAB group at the University of Thessaly (Greece) [23] towards the interconnection of geographically distributed OMF-based wireless testbeds through PlanetLab [III.1],[III.5]. We have allowed the making of experiments involving the use of resources provided by local wireless testbeds in combination with other resources provided by other remote sites connected to the PlanetLab planetary-scale testbed. This allows running experiments on wide-area infrastructures, involving several kinds of technologies, both in the core of the network, where they cannot be controlled by experimenters, and at the edges, where they can be selected to compare several kinds of access networking technologies, such as Wi-Fi, WiMAX, UMTS, Wireless Mesh Networks.

The contribution presented is in line with current ongoing efforts towards the so called "federation" of experimental infrastructures. A testbed federation has been recently defined as *the interconnection of two or more independent testbeds for the creation of a richer environment for experimentation and testing, and*

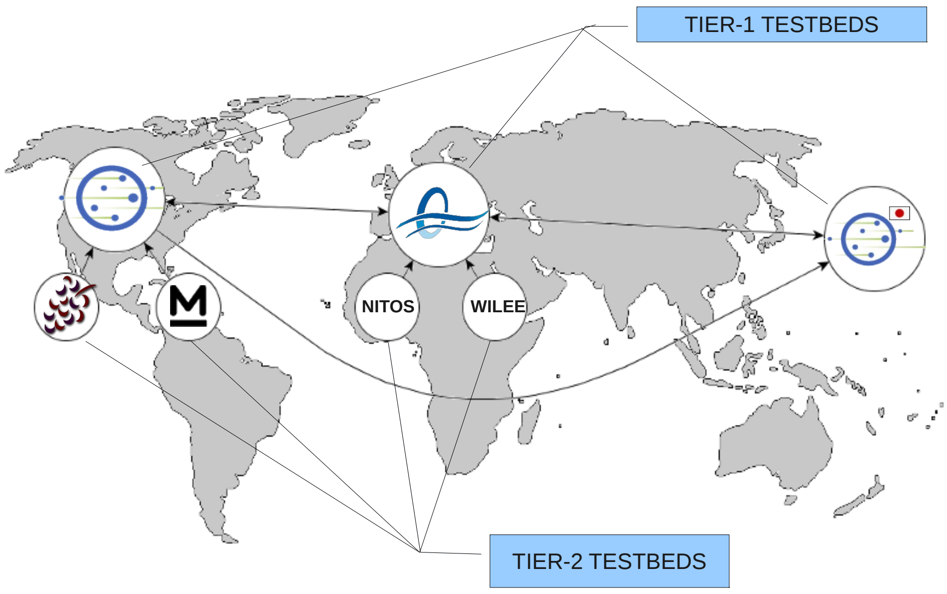


Figure 3.1: Hierarchical federation of heterogeneous testbeds.

for the increased multilateral benefit of the users of the individual independent testbeds [24] and it currently appears as the most reasonable way to build large-scale heterogeneous testbeds. Roadmaps envisioned by the most significative research initiatives focusing on future research infrastructures, such as GENI [25], [26] and FIRE [27], assign a key role to federation of existing testbeds. Actually, we envision a hierarchical federation model, as depicted in Figure 3.1, in which global scale Tier-1 testbeds, federated among them in a peer-to-peer way, act as "aggregators" of local Tier-2 testbeds. In this view, we assume PlanetLab and PlanetLab Europe as existing Tier-1 testbeds, whose federation is already in place and operational since 2008.

Federation of heterogeneous testbeds involves a number of both technical and organizational issues. With regards to the technical challenges, they comprise the problem of sharing user credentials, as well as armonising usage models and resource management policies among testbeds. Our contribution accounts for such problems and we will describe hereinafter how we dealt with them. Thus, our contribution can be viewed as a preliminary effort in the direction of the *federation* of two different kinds of testbeds that we feel are of extreme importance for researchers working on wireless mesh networks.

The rest of the chapter is organized as follows. In section 3.1 we describe a component devoted at the reservation of resources of the OMF testbed. We modified this component to make it handle also PlanetLab bookable resources.

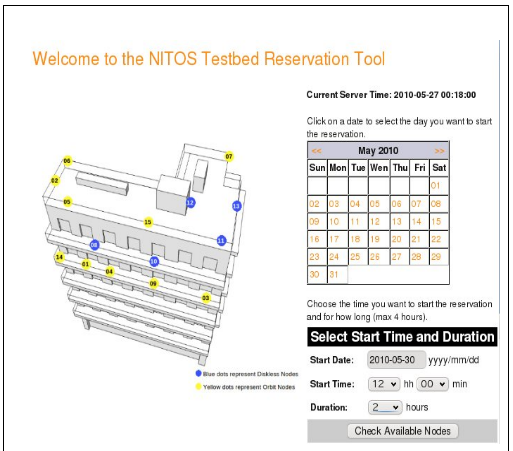
In section 3.2 we describe the integration steps that we developed to allow for distributed experiments involving OMF-based wireless mesh testbeds and PlanetLab.

In section 3.3 we illustrate how we used the integrated testbed setup to conduct an experiment aimed at evaluating a peer-to-peer traffic optimization technique. This is a typical distributed experiment in the PlanetLab wired environment, but in our case it involves the usage of a wireless mesh as an access network, which would not be possible in the plain PlanetLab environment.

Finally, in section 3.4 we draw the conclusions on the relevance of this contribution and its potential for future developments.

The NITOS scheduler

The OMF standard distribution does not include any scheduling algorithm to synchronize the execution of experiments. For this reasons, some developers of the University of Thessaly in Greece, with whom we collaborated, developed the NITOS Scheduler [28], a system which adds scheduling and reservation capabilities to the OMF platform. The NITOS Scheduler works by dividing the testbed resources into two categories: nodes and spectrum. In assigning such resources to experimenters, the NITOS Schedulers applies the concept of *slicing*, which consists in assigning to each user a subset of nodes and available transmission frequencies. Different users can therefore run concurrently their experiments, as each experiment will make use of different nodes and different transmission frequencies (supposed to be orthogonal). The NITOS Scheduler consists of a web inter-



Welcome to the NITOS Testbed Reservation Tool

Current Server Time: 2010-05-27 00:18:00

Click on a date to select the day you want to start the reservation.

<<			**May 2010**			>>
Sun	**Mon**	**Tue**	**Wen**	**Thu**	**Fri**	**Sat**
						01
02	03	04	05	06	07	08
09	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Choose the time you want to start the reservation and for how long (max 4 hours).

Select Start Time and Duration

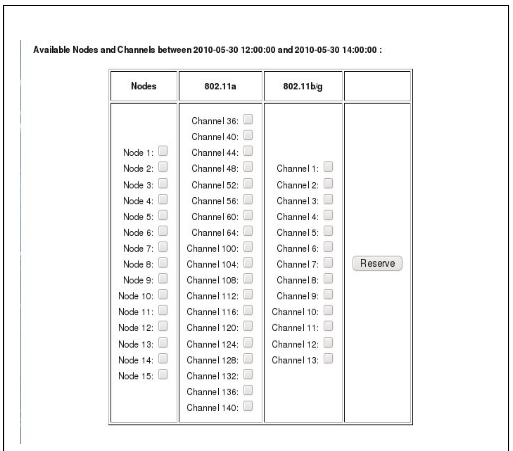
Start Date: 2010-05-30 yyyy/mm/dd

Start Time: 12 hh 00 min

Duration: 2 hours

Blue dots represent Diskless Nodes
Yellow dots represent Orbit Nodes
Check Available Nodes

(a) Fill in date and time



Available Nodes and Channels between 2010-05-30 12:00:00 and 2010-05-30 14:00:00 :

Nodes	**802.11a**	**802.11b/g**	
	Channel 36: ☐		
	Channel 40: ☐		
Node 1: ☐	Channel 44: ☐		
Node 2: ☐	Channel 48: ☐	Channel 1: ☐	
Node 3: ☐	Channel 52: ☐	Channel 2: ☐	
Node 4: ☐	Channel 56: ☐	Channel 3: ☐	
Node 5: ☐	Channel 60: ☐	Channel 4: ☐	
Node 6: ☐	Channel 64: ☐	Channel 5: ☐	
Node 7: ☐	Channel 100: ☐	Channel 6: ☐	
Node 8: ☐	Channel 104: ☐	Channel 7: ☐	Reserve
Node 9: ☐	Channel 108: ☐	Channel 8: ☐	
Node 10: ☐	Channel 112: ☐	Channel 9: ☐	
Node 11: ☐	Channel 116: ☐	Channel 10: ☐	
Node 12: ☐	Channel 120: ☐	Channel 11: ☐	
Node 13: ☐	Channel 124: ☐	Channel 12: ☐	
Node 14: ☐	Channel 128: ☐	Channel 13: ☐	
Node 15: ☐	Channel 132: ☐		
	Channel 136: ☐		
	Channel 140: ☐		

(b) Resources Availability

Figure 3.2: NITOS Scheduler Interface: example of reservation

face through which users make reservations and some software components that enforce the reservation at experiment time in collaboration with the OMF components. As our efforts aimed at allowing the execution of experiments involving both PlanetLab and OMF testbeds, we had to modify the NITOS Scheduler (creating the extended NITOS Scheduler) to make it support also the reservation of PlanetLab resources.

3.2 Integration in PlanetLab of wireless mesh access networks

Our main goal was to integrate a global scale PlanetLab infrastructure with a local OMF-based wireless testbed. In particular, we aim at using the OMF-based testbed as an access wireless mesh network for a set of PlanetLab nodes co-located (i.e. in range of wireless transmission) with it.

As described in the introduction, we recognize a value in this integration, as a first necessary step for the federation of these two kinds of infrastructures, and because it adds new capabilities to the PlanetLab environment. To this purpose, we engineered some basic mechanisms for accessing the resources provided by a OMF-based wireless testbed from a PlanetLab node co-located with it. Our sys-

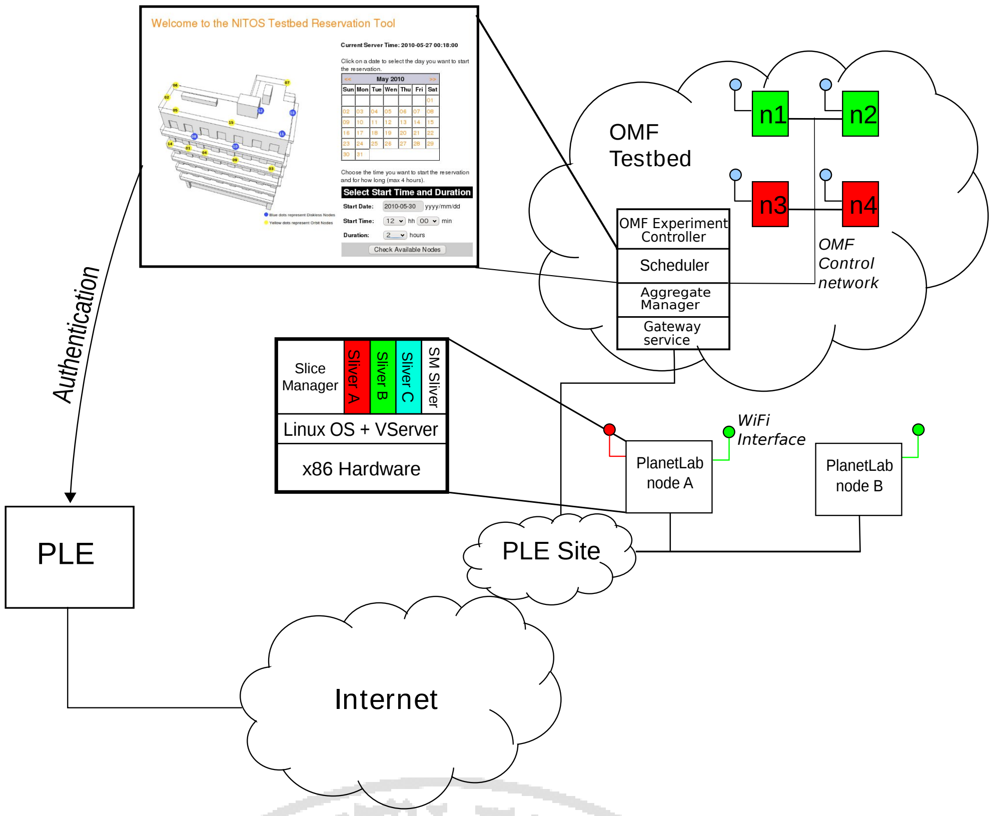


Figure 3.3: OMF-PlanetLab integrated architecture.

tem allows the seamless integration of the OMF-resources into the global scale PlanetLab infrastructure, creating a synergic interaction between the two environments.

Access to the OMF testbed is allowed through one or more *edge PlanetLab nodes*, whose configuration grants access to the OMF testbed resources only to specific slices. Such configuration is controlled by the NITOS scheduler of OMF resources, which assigns OMF resources to specific slices instantiated on the PlanetLab edge node.

3.2.1 Architecture

The architecture realized is depicted in Fig. 3.3. It consists of the following elements:

- A PlanetLab site S whose nodes are equipped with one ore more Wi-Fi interfaces that allow them to be connected to a local wireless OMF testbed. In the following these nodes are called *PlanetLab Edge Nodes* (PL-Edge Nodes).

- The PlanetLab Europe Central server (PLE), which hosts the information on the PlanetLab Europe testbed, e.g. user accounts, slices.

- The OMF testbed and its components: the Aggregate Manager, the Experiment Controller and the Gateway Service.

- The extended NITOS Scheduler, used to manage the reservation of resources shared through booking.

The Gateway Service is implemented in a Linux box and acts as a *Network Address Translator* (NAT). It is needed for enabling Internet access to the OMF testbed's nodes, whose NICs are assigned private IP addresses.

The PL-Edge nodes are multi-homed PlanetLab nodes which can act as clients for the OMF wireless testbed. The lack of proper support for multihoming in PlanetLab led us to the developement of *sliceip*, a tool for allowing the definition of slice-specific routing tables that will be presented later.

In the OMF-PlanetLab integrated scenario, two kinds of resources are made available to experimenters:

- *bookable resources*, i.e. resources that can be exclusively assigned to an experiment over a given time interval;

- *non-bookable resources*, i.e. resources that cannot be exclusively assigned to an experiment over a given time interval, as they are shared among concurrently running experiments;

The purpose of the extended NITOS scheduler is to allow the reservation of bookable resources in the integrated scenario. These resources comprises both OMF wireless nodes and channels, and PlanetLab non-virtualized resources, i.e. the Wi-FI interfaces. To do that, the extended NITOS scheduler interacts with the

OMF Console, in order to enable or disable access to slices to the Experiment Controller, and with the PlanetLab nodes, in order to enable or disable the access to specific slices to the wireless interfaces. The communication with the PlanetLab nodes is performed by means of a management sliver, called *SM Sliver* (Scheduler Management Sliver), which accepts requests by the Scheduler through a secure ssh connection and performs the association between the slices and the wireless interfaces. We remember that we allow only one slice at a time to have access to a wireless interface, in order to limit interferences among experiments.

The Scheduler performs authentication of the user on the PLE, thus allowing access to the Tier-2 OMF wireless testbed to PlanetLab Europe users. Local users, i.e. users of the wireless testbed, are supported and their credential are stored on the Scheduler. These class of users however, i.e. users of the Tier-2 testbed, have not access to the global infrastructure, i.e. the Tier 1 testbed.

In the OMF wireless testbed private IP addressing is used. Therefore, in order to allow experiments involving nodes located elsewhere on the public Internet, a node acting as a NAT router is needed. This function is performed by the Gateway Service. In the case of experiments involving OMF nodes located at different PL-OMF sites, site-to-site IP tunnels might be established between PL-OMF Edge Nodes. This process would be easy to be managed if these nodes were VINI nodes.

After user authentication the OMF Scheduler, by means of cron scripts, enables/disables access to OMF testbed nodes from the user's slice.

3.2.2 Usage model

In the following we list the sequence of steps needed to execute an experiment using an OMF testbed at site S as access network for PlanetLab. The experiment is going to be executed over a specific time interval T= $[T\_START, T\_END]$.

1. PlanetLab user U adds one or more PL-OMF Edge Nodes (OP) to his/her slice;

2. U logs into the Scheduler at site S and books the resources (nodes, channels,

Wi-Fi interfaces of OP nodes) he needs for his/her experiment over time interval T, providing the slice identifier. According to PlanetLab's resource management scheme, booked resources are actually associated with such slice rather than with the user that performed the reservation;

3. While time is in T, each slice's user is allowed to access the OMF EC (Experiment Controller) to perform his/her experiment involving the OMF resources assigned to him/her.

3.2.3 Adding multihoming support in PlanetLab

While trying to support the proposed usage model, we run across a serious limitation of the PlanetLab management software. Such a limitation is about the correct managing of multi-homed nodes, i.e. nodes connected to more than one access network. This has not been a problem for a long time, as PlanetLab mainly consisted of just a set of hosts connected to Internet through a single, high speed corporate connection. In such a scenario, there is no need for users to be able to modify the routing table, as the route for the Internet is only one. In recent times, though, some attempts to enhance the heterogeneity of PlanetLab have been made. In the context of the OneLab European research project, different kinds of wireless access technologies (such as UMTS, WiMAX and Wi-Fi) have been made available to a subset of nodes connected to PlanetLab Europe, in addition to the default wired connection to the Internet. In [III.4], the software tools that have been developed to manage a UMTS connection in that context are described. In the following section, we describe a generalization of that software, that allows experimenters to work with any kind of network interface.

The sliceip tool

In order to fully exploit the possibility of multi-homed PlanetLab nodes we developed a tool called *sliceip*. The purpose of this tool is to enable slice-specific routing tables in PlanetLab. Using this tool, the user is able to define routing rules which apply only to traffic belonging to his/her slice. This is required for users to be able to choose which interface to use for their experiments. For instance, a user can specify that he or she wants to reach a certain destination on the Internet, e.g. another PlanetLab node, through the Wi-Fi interface. For achieving this result, he or she would add a routing rule in his/her own routing table by means of our tool, in the same way he or she would do with conventional tools like *ip* of *route*. This is not possible in PlanetLab, because PlanetLab users do not have the superuser privileges required to modify the routing table of the node. Even if they had such privileges, any modification they performed on the routing table would interfere with all the experiments running on that node, thus breaking the isolation among experiments. With sliceip, instead, we give to the user the ability to define his/her own routing table, with no effects on experiments performed by other users.

sliceip enables slice-specific routing tables by leveraging a feature of the Linux kernel and a feature of the VNET+ subsystem of PlanetLab [29]. The Linux kernel has the ability to define up to 255 routing tables. To have some traffic routed with a particular routing table, it is necessary to associate that traffic to it by means of rules applied with *iproute2*. The rules can specify packets in terms of the destination address, the netfilter mark, etc. In our case, we set the netfilter mark of packets belonging to the user's slice (i.e. the packets that are generated or are going to be received by an application running on that slice) by exploiting a feature of the VNET+ subsystem of PlanetLab. By means of an *iptables* rule, we instruct VNET+ to set the netfilter mark equal to the slice id to which they belong. We then add an *iproute2* rule to associate packets belonging to the slice to the slice-specific routing table. We also set an *iptables* SNAT rule (*Source Network Address Translation*) in order to set the source IP addresses of packets that are going out through a non-primary interface (the primary interface is the one the default routing rule points to). This rule is required because the source ip addresses of packets are set after the *first routing process* happens. In fact, in case more than a routing table is used, the routing process follows these steps: 1) the interface for sending the packets is decided following the rules of the main routing table and the source ip addresses are set accordingly (this is the first routing process); 2) if the user changes the mark of the packets in the *mangle chain* of *ipt-*

ables and a rule is defined for routing those packets with a different routing table, a *rerouting process* is triggered. This rerouting process follows the rules of the selected (i.e. the slice-specific) routing table and the interface to be used is set accordingly; 3) the packet is sent out using the selected interface. During the step 2, the source ip addresses of packets are left unchanged, so we need to change them explicitely before the packets are sent during the step 3.

The user interacts with *sliceip* by means of a front-end that resides in the slice. This front-end extends the syntax of the *ip* command of the *iproute2* suite with the following two commands:

- *enable <interface>*: initialise the routing table for the user's slice, set the rule to mark packets belonging to the user's slice, add a rule to associate those packets with the routing table of the slice and add the SNAT rule for <interface>;
- *disable <interface>*: remove the SNAT rule for <interface>, remove the rule to associate the packets to the routing table of the slice and remove the rule that marks the packets of the user's slice.

The tool sliceip has been integrated in the vsys library in collaboration with some researchers of the Princeton University [III.3].

3.2.4 Extension of the NITOS scheduler to manage PlanetLab resources

In order to support the reservation of bookable Planetlab resources, i.e. the Wi-Fi interfaces of the PL-edge nodes, we extended the NITOS Scheduler and made some additions to the management software of the PL-edge nodes.

The Scheduler has been extended to show among the available resources also the Wi-Fi interfaces of the PL-Edge Nodes and to allow the user to reserve them. Reservation records are kept in the Scheduler database and it is Scheduler responsibility to make sure that reservations made by two users do not overlap.

In order to enforce the assignment of the interface to the slice, when the reservation time starts, the Scheduler interacts with the *Scheduler Management Sliver*

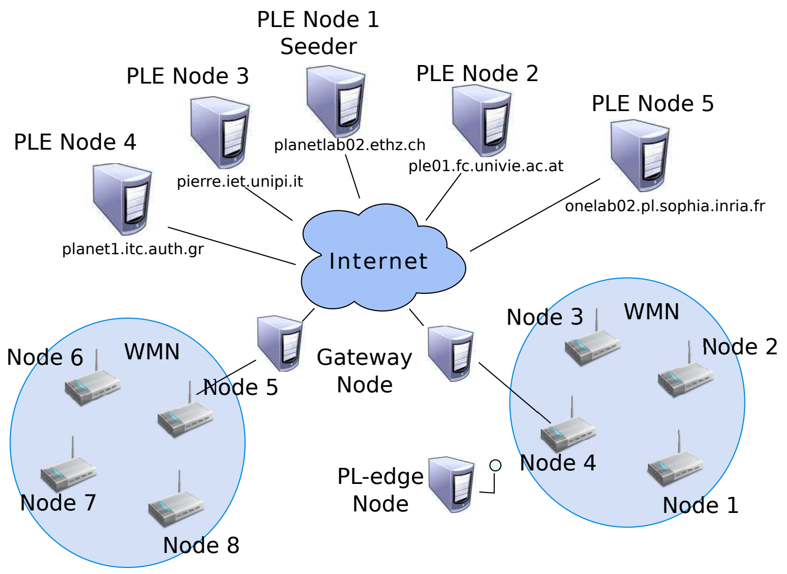


Figure 3.4: Experiments setup.

allocated on the PL-edge node. Such interaction is performed through a secure ssh connection. By means of *vsys* [30], the *Scheduler Management Sliver* is able to execute a script in the root context. This script makes the actual assignment of the Wi-Fi interface to the slice by setting some *iptables* rules which block all packets that are about to go out through the Wi-Fi interface and do not belong to the slice for which the Wi-Fi interface has been reserved.

The Scheduler checks the user's credentials by means of the PLC API and enables/disables access to OMF testbed nodes from the user's slice for the specific time and duration. In particular, the Scheduler interface is extended to support authentication of users by means of PLC managed usernames and passwords, while access to the OMF EC is performed by means of users' public keys linked to the slice, retrieved using the PLC API.

3.3 Experimental evaluation

In the following sections an experiment aimed at investigating a problem that is frequently studied on top of PlanetLab, i.e. peer-to-peer traffic optimization is described. The peculiarity, in our case, is that we create a distributed setup for our experiment involving the use of our wireless mesh testbeds as access networks to the Internet. In fact, we intend to investigate this problem, and compare its solutions, in the specific context of WMNs, where specific cross-layer approaches can be part of the solution. In the following we present how we conducted the experiments and the reasons that make our integrated infrastructure useful for evaluating wireless meshes in realistic conditions. The experiments were carried on on the NITOS and WILEE testbeds that are described in the following.

The NITOS testbed

NITOS is a wireless testbed located in the University of Thessaly campus. NITOS aims to provide all the software and hardware facilities that can gather multiple wireless communication technologies under a common structure.

NITOS testbed features 3 different types of computer main boards, 2 types of wireless media as well as 2 other types of peripherals. More specifically the NITOS testbed features 10 Alix embedded PoE nodes with 500Mhz i386 CPUs, which are primarily used for development of networking systems, 10 Orbit AC powered nodes (1 Ghz i386 CPUs and 1 Gb ram) and 20 Commel AC powered nodes that feature 2.4 GHz core duo CPUs (x86_64). Wireless media includes 50 Atheros 5212 interfaces and 10 Atheros 5001 interfaces.

The WILEE testbed

The WILEE (WIreLEss Experimental) Wi-Fi Mesh Testbed is a testbed located in the Computing Department of University of Napoli Federico II. It consists of three Soekris net4826-48 Single Board Computers and eigth Netgear WG302Uv1 access points. It also features a node belonging to a private PlanetLab deployment which acts as the PlanetLab Edge node and a Linux machine acting as gateway towards the Internet.

It consists of:

- 3 Soekris net4826-48 Single Board Computers;
- 8 Netgear WG302Uv1 access points;

- 1 Linux machine acting as gateway towards the Internet;
- 1 node belonging to a private PlanetLab deployment (the *PlanetLab edge node*).

The testbed ia managed by OMF and can be accessed by researchers through a PlanetLab edge node.

The Soekris net4826-50 SBC is based on the AMD Geode SC1100 CPU (at 266Mhz), has 128 Mbyte DRAM memory, a 128 Mbyte Flash disk, a FastEthernet interface and two 802.11a/g Atheros wireless cards. The Netgear WG302Uv1 access point features on an Intel XScale IXP422B network processor (at 266Mhz), has 32 Mbyte DRAM memory, a 16 Mbyte flash disk, a FastEthernet interface and two 802.11a/g Atheros wireless cards.

3.3.1 Testing overlay routing strategies in WMN-based access networks

An increasing number of popular Internet applications, such as Bittorrent, Skype, GoogleTalk, and P2P-TV relies on the peer-to-peer paradigm. These applications produce more than 50 percent of the overall Internet traffic. One of the inherent characteristics of peer-to-peer systems is that they build *network overlays* among their peers, and route traffic among them along the virtual links of such an overlay. Peer-to-peer routing decisions are made at the application layer, independently of Internet routing and ISP topologies. Hence, overlay routing decisions collide with those made by underlay routing, i.e. ISP routing decisions [31]. As a consequence of such a dichotomy, several inefficiencies may result. For instance, it is not uncommon that adjacent nodes of an overlay network are in different ASes. Such a topology arrangement leads to traffic crossing network boundaries multiple times, thus overloading links which are frequently subject to congestion, while an equivalent overlay topology with nodes located inside the same AS could have had same performance. Such a behavior is undesirable for ISPs, also because their mutual economic agreements take into account the volume of traffic crossing the ISP boundaries.

From what is described above, it emerges that overlay routing, and peer-to-peer applications, may benefit from some form of underlay information recovery, or in general from cross-layer information exchange. Aggarval et al. in [32] suggest that such a cooperation would be beneficial for both ISPs and users. When creating an overlay network, the choice of the nodes to be connected, i.e. the network topology, can be done by taking advantage of information from the underlay network. Different strategies have been proposed recently in the literature that attempt to introduce some cooperation between the two routing layers [32][33]. Given the role of access networks played by wireless mesh networks, it is interesting to experiment with such techniques when peers are attached to different WMNs connected to the Internet. Our contribution makes such experiments possible. In the next subsection, we describe experiments carried out to show that our approach makes it very simple to perform realistic experiments to test overlay routing strategies.

3.3.2 Experimenting with P2P applications on the integrated environment.

In the following we describe an experiment aimed at evaluating a traffic optimization solution for a BitTorrent file-sharing peer-to-peer system. BitTorrent is used to efficiently distribute files of large size from one or more initial *seeds* to a population of large numbers of downloaders, forming what is referred to as a *swarm*. Files are exchanged in smaller *chunks* that can be individually retrieved. One of the peculiarities of BitTorrent is that downloaders, a.k.a. *leechers* in BitTorrent terminology, also contribute to spread the content to other peers. As soon as a peer obtain all the chuncks of the desired file, it becomes a seed on its own. We have designed and implemented a solution that aims at incentivating traffic exchange in a BitTorrent system between peers that are located within the same Autonomous System. Our solution does not require any modification to the BitTorrent protocols, nor to the application used by end users. The only modified component of a typical BitTorrent system is the *Tracker*, i.e. the system that is contacted by peers to obtain a list of other peers to contact, in order to retrieve chunks of the file to download. In our system, the tracker returns to peers a sorted list of peers to be contacted, where the sorting criterion is by-increasing-AS-distance. In other terms, as soon as a peer contacts the tracker, the tracker determines the AS-number associated with the IP address of that peer, and returns a list of peers whose first items are the closest peers in the swarm (in terms of AS distance), while the last items are the furthest peers. Our experiment is aimed at evaluating our tracker-based solution when a significant fraction of peers are connected to the Internet through the same wireless mesh network. Our objective is to show that in this case, by adopting our strategy, a substantial amount of traffic is reduced through the wireless mesh gateway, i.e. the node connecting the wireless mesh to the wired Internet. To this purpose we created a slice involving ten PlanetLab Europe nodes and the PlanetLab edge node situated at the edge of the WILEE testbed. To this slice, some bookable resources, i.e. four wireless nodes from the WILEE testbed and the Wi-Fi interface of the PL-edge node, were added to the slice by using the extended NITOS Scheduler at the WILEE site. In the same way, other four nodes belonging to the NITOS testbed were added by using the exented NITOS Scheduler at the NITOS site.

The wireless nodes were configured by using the facility offered by OMF to form two single-channel WMNs and, in case of WILEE nodes, also to provide Internet access to the PL-edge node. A Bittorrent client (*TransmissionBT*) was installed on the PlanetLab Europe nodes, on the PL-edge node and on the wireless nodes. One of the PlanetLab Europe nodes was chosen as the seeder of the Bittorrent swarm, which consisted of a file of approximately 50 megabytes. The scenario of the experiments is illustrated in Fig. 3.4.

We performed a set of experiments by employing alternatively a standard Bittorrent tracker (*Quash*) and the same tracker modified by us in order to take into account the distance between peers in terms of ASes.

At the end of each experiment we measured the traffic belonging to connections which were either originated or destined to nodes located behind the OMF gateways, i.e. the NITOS and WILEE wireless nodes and the PL-Edge node. Our objective was to demonstrate that the traffic crossing the WMNs boundaries

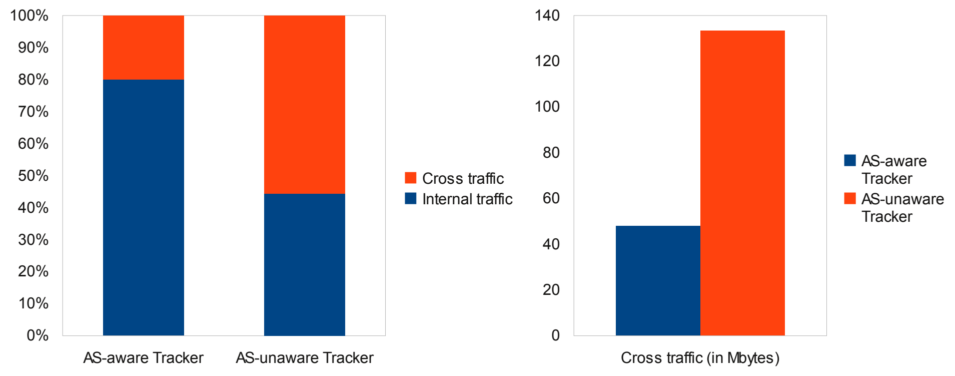


Figure 3.5: Experiments: internal vs. cross traffic (percentage of total traffic) on the left; cross traffic volume on the rigth.

Table 3.1: Traffic matrix for an experiment with the modified Tracker.

	N1	**N2**	**N3**	**N4**	**PL-Edge**	**N5**	**N6**	**N7**	**N8**	**PlanetLab**
N1	0	2.34	1	0	0.44	0	0	0.81	0	41.03
N2	0	39.77	0.06	0.06	1.39	0	0	0	0	5.69
N3	13.99	3.9	0	1.89	27.19	0	0	0	0	0
N4	13.36	3.61	5.27	0	26.45	0	0	0	0	0
PL-Edge	40.7	4.23	0.64	0.09	0	0	0	0	0	0
N5	0	0	0	0	0	0	0.13	0.09	0	45.03
N6	0	0	0	13.2	0	29.79	0	0	3.55	0
N7	0	0	0	0	0	20.12	23.91	0	2.29	0
N8	0	0	0	0	0	8.17	1.95	0.5	0	37.05

was minimized by using our modified tracker. In Fig. 3.5 we report the results averaged on 10 repetitions. The figure shows that the amount of traffic flowing through the OMF Gateways was significantly lower in case the modified tracker was used. If we compare the overall amount of bytes exchanged by peers, the results show that, in case the modified tracker was used, the file was downloaded in average from the outside slightly more than once for each WMN, and then disseminated in the WMNs among nearby nodes. In case the unmodified tracker was employed, instead, it is as though the file was retrieved almost three times by each WMN (about 280 Mbytes downloaded from the outside by the two WMNs), thus indicating a non-optimum peer selection strategy. Tables 3.1 and 3.2 report the traffic matrices for two experiments. On the rows are the receiving nodes, while on the columns are the sending nodes. N1, N2, etc. stand for Node1, Node2, etc.,

Table 3.2: Traffic matrix for an experiment with the standard Quash Tracker.

	N1	**N2**	**N3**	**N4**	**PL-Edge**	**N5**	**N6**	**N7**	**N8**	**PlanetLab**
N1	0	0	0	2.88	0	0	0	0	0	43.37
N2	0	0	0	5.5	62.3	0	0	0	0	4.38
N3	0	0	0	0	4.73	0	0	0	0	48.84
N4	44.43	7.52	0	0	0	0	0	0	0	0
PL-Edge	0	0	7.88	0	0	0	0	0	0	46.88
N5	0	0	0	0	0	0	0	0	22.97	24.29
N6	0	0	0	0	0	0	5.31	0	0	40.82
N7	0	0	0	0	13.65	0	0	0	0	37.53
N8	0	0	0	0	10.82	19.88	0	0	0	16.14

while PlanetLab is a meta node which comprises all the PlanetLab nodes. All the values are in Mbytes. It can be seen that, in case the modified tracker is used (Table 3.1), traffic is exchanged mainly between nodes located inside the same WMN, while in case the standard tracker is used (Table 3.2), wireless nodes often download from nodes which are outside their WMN.

While conducting the experiment, some real world issues arised and made evident the usefulness of having such an heterogeneous network scenario.

The first problem was about the private addressing of the WMN and the need to NAT the traffic generated from the wireless nodes and destined to the Internet. This was, however, not sufficient, as the Bittorrent protocol requires that the clients be reachable from the outside on public IP-port pairs. For this reason, we had to setup a NAT-PMP service on the gateway node [34]. Through this protocol, clients are able to request a port to be forwarded from the gateway node, so that they can accept incoming connections from other peers on the gateway IP and the assigned port.

Clients, therefore, announce themselves to the Tracker with their public IP-port pair. This requires, in turn, that the connections between two wireless nodes go through the gateway machine and be source NATted, at the gateway node, even if they do not involve a node on the Internet. Solutions to this problem require modification to the Bittorrent client, e.g. in order to implement a local peer discovery process.

The IETF (Internet Engineering Task Force) [1] is currently defining an architecture called ALTO (Application-Layer Traffic Optimization) [35] which can be exploited to optimize application layer traffic by means of information on the network topology made available by entities called *ALTO Servers*. We implemented an ALTO-compliant system for p2p traffic optimization specifically tailored for WMNs and tested it on the PlanetLab-OMF integrated infrastructure [III.2].

3.4 Conclusions

The availability of large scale testbeds integrating several local wireless mesh testbed in a realistic global-scale environment is necessary to test WMNs in the wild. In this chapter we presented an integration architecture that allows to combine local OMF-based wireless testbeds with the planetary-scale PlanetLab infrastructure. In particular, we described how we solved the problem of harmonizing the resource management schemes of the two testbeds, that comprise both bookable and non-bookable resources. We also present some test case experiments we run on our initial implementation of the integrated architecture. In particular, we describe an experiment aimed at evaluating a BitTorrent traffic optimization system. Our experiment includes two OMF-based wireless testbeds (namely, NITOS and WILEE) as well as a number of PlanetLab nodes located across Europe. We believe that the integration achieved between PlanetLab and OMF-based testbeds can greatly help the research related to the design of new routing and channel assignment algorithms for wireless mesh networks, because of the high heterogeneity of the resulting experimental scenario.

Relevant publications

[III.1] G. Di Stasi, R. Bifulco, S. Avallone, R. Canonico, A. Apostolaras, N. Giallelis, T. Korakis, L. Tassiulas. “Interconnection of geographically distributed wireless mesh testbeds: resource sharing on a large scale.” . AdHoc Networks, Elsevier, 2011

[1] http://www.ietf.org

[III.2] F. P. D’Elia, G. Di Stasi, S. Avallone, R. Canonico. “Bittorrent traffic optimization in Wireless Mesh Networks with ALTO service”. HotMesh, June 20-24, 2011, Lucca, Italy

[III.3] S. Bhatia, G. Di Stasi, T. Haddow, A. Bavier, S. Muir and L. Peterson. “vsys: A programmable sudo”. USENIX ’ATC 2011

[III.4] A. Botta, R. Canonico, G. Di Stasi, A. Pescape’, G. Ventre, S. Fdida, ”Integration of 3G connectivity into a PlanetLab based testbed - A step of an evolutionary path towards heterogeneous large scale network testbeds”, ACM/Springer Mobile Networks and Applications, Volume 15, Issue 3, June 2010, Pages 344-355

[III.5] G.Di Stasi, S. Avallone, and R. Canonico. “Integration of OMF-based testbeds in a global scale networking facility“, volume 22 of Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, pages 545-555. Springer, Heidelberg, 2009

[III.6] G. Di Stasi, R. Bifulco, S. Avallone, R. Canonico, A. Apostolaras, N. Giallelis, T. Korakis, L. Tassiulas., ”Experimenting with P2P traffic optimization for Wireless Mesh Networks in a federated OMF-PlanetLab environment“, WCNC (Wireless Communications and Networking Conference), Cancun, Messico, March 28-31, 2011

Chapter 4

Wireless mesh networks

4.1 Introduction

In this chapter we describe the general architecture and the model of wireless mesh network we assume and discuss in general the problems of channel assignment and routing. We give, in particular, the definition of *total utilization* of a collision domain and describe how it can be related to the schedulability of a set of flow rates. Such a result will be used in the subsequent chapters, where we describe our channel assignment and routing algorithms.

4.2 Wireless mesh network architecture

A wireless mesh network (WMN) is comprised of a set of mesh routers capable of wirelessly delivering, through multiple hops, packets either destined or originated from wireless clients. Wireless clients can connect to the WMN through special mesh routers which have, in addition to routing capabilities, also access point capabilities (the so called *mesh aggregation devices*). WMNs are usually capable to connect to other networks (e.g. a wired backbone) through mesh routers with bridge capabilities (the *mesh gateways*). Mesh routers are usually fixed and therefore do not have strong requirements in terms of energy consumption. The architecture described is commonly known as *infrastructure/bachbone mesh* (see Fig. 1.1).

There exist variants of such architecture: one of such variants, called *hybrid mesh*, consists in having also wireless clients delivering packets on behalf of other clients. This allows to extend the coverage of the network through the collaboration of clients. This architecture is less popular, as it requires modifications at the client side.

Another architecture, called *client mesh*, consists in having only wireless clients that deliver packets among them through multiple hops. This architecture, which is even less widespread, can be considered a degeneration of the hybrid architecture, where there are only clients and no mesh routers.

Mesh routers are usually equipped with multiple radios that allow them to transmit simultaneously on different channels, i.e. range of frequencies, therefore reducing the interference.

The availability of multiple radios per node leads to the channel assignment problem, i.e., the problem how to select a channel for each radio in the network. Such a channel can be changed at the packet level or can be used for a relatively long time, e.g. tens of minutes. The change of channel at the packet level, or with a high frequency, requires support from the network hardware, which is not avaible in current off-the-shelf IEEE 802.11 tecnology.

The problem of routing is related to the way packets are delivered between any two nodes of the network. As bandwidth is a limited resource due to interference, routing should be able to distribute traffic in order not to exceed the available bandwidth on each link.

Forwarding is the part of the routing process that is in charge of delivering the packet to the next-hop. It chooses the next-hop among a set of candidate next-hops which belong to the paths that are configured by other procedures of the routing process.

Routing and channel assignment in wireless mesh networks are not independent problems [36]. This is because the channel assignment algorithm needs to be aware of the traffic to be routed on each link in order to assign to it the required bandwidth. The traffic to be routed on each link is known, however, only after the routing problem has been solved. On the other hand, the routing problem needs to be aware of the bandwidth available on each link, which is not known until the channel assignment problem has been solved. Routing and channel assignment need therefore to be solved jointly, which leads to a problem which is NP-complete [37]. For this reason, the proposals that recently appeared in the literature addressing such joint problem solve the channel assignment problem and the routing problem separately. A common approach is to first solve the routing problem, i.e., how to determine the amount of flow (referred to as the *flow rate*) to be routed on each link, and then to solve the channel assignment problem, i.e., how to assign channels in such a way that the resulting bandwidth available on each link exceeds the link flow rate.

Even if addressed in this way, the channel assignment algorithm is NP-complete

[37]. For this reason only heuristics are provided, such as the one we propose in chapter 5.

4.3 Wireless mesh network model

We assume that each mesh router u is equipped with $k(u) \geqslant 1$ radio interfaces and there are $|\mathcal{C}|$ available channels. For every radio, we assume a fixed transmission power, while the transmission rate can be selected in the (increasingly) ordered set $\{r_m\}_{m=1}^{M}$. Given that a radio may serve multiple links and the ability of commodity hardware to set the transmission power and rate on a per-packet basis, we will assign rate and power to links rather than radios, meaning that specific rate and power values will be assigned to a radio when it is sending packets for the corresponding link. We adopt the *physical* model of interference, which considers a transmission successful if the Signal-to-Interference and Noise Ratio (SINR) at the receiver is sufficiently high to decode the signal. The SINR at receiver v when a signal is transmitted by u is defined as

$$SINR_{uv} = \frac{G_{uv}P_u}{\sum_{x\to y\neq u\to v} G_{xv}P_x + n_v}$$

where P_u is the power emitted by u to transmit to v, G_{uv} is the gain of the radio channel between u and v, and n_v is the thermal noise at receiver v. If u transmits at rate r_m, the receiver v can correctly decode the signal if $SINR_{uv} \geqslant \gamma_{r_m}$, where γ_{r_m} denotes the minimum SINR required to correctly decode a signal modulated at the rate r_m. It is a known result that the higher the transmission rate, the higher is the SINR threshold.

We model the WMN as a directed graph $G_I = (V, E_I)$, where V is a set of nodes each representing a mesh router. Given two nodes $u, v \in V$, the directed edge $u \to v \in E_I$ iff, in the absence of transmissions on other links, the signal to noise ratio at v is larger than the SINR threshold for one of the available transmission rates r_m, i.e., $\frac{G_{uv}P_u}{n_v} \geqslant \gamma_{r_m}$. The capacity of the link $c(u \to v)$ is then set to the highest transmission rate for which the signal to noise ratio is larger than the corresponding threshold. An edge $u \to v \in E_I$ indicates that u can transmit

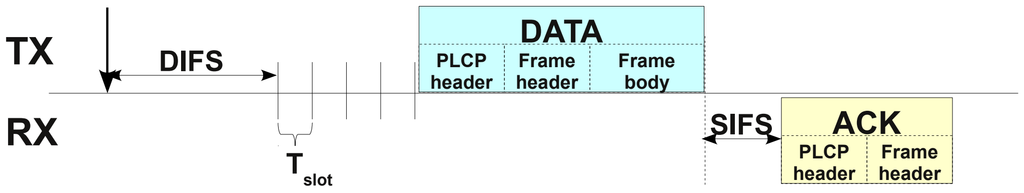


Figure 4.1: Wireless medium access according to 802.11 DCF

to v provided that they are assigned a common channel. A channel assignment $\mathcal{A}$ assigns a set $\mathcal{A}(u)$ of channels ($|\mathcal{A}(u)| \leqslant k(u)$) to each node $u \in V$. Thus, $\mathcal{A}$ induces a new graph model $G = (V, E)$ where two nodes u and v are connected if $u \rightarrow v \in E_I$ *and* they share at least one common channel. In case u and v share multiple channels, the set E may include as many links between the two nodes as the number of common channels. To differentiate among those links and stress that a link has been assigned channel c, we use the notation $u \xrightarrow{c} v$. Finally, we say that a link $x \xrightarrow{c} y \in E$ interferes with $u \xrightarrow{c} v \in E$ if a simultaneous transmission on $x \xrightarrow{c} y$ prevents v from correctly decoding the signal from u.

4.4 Channel assignment

Several proposals have appeared recently in literature that cope with the problem of channel assignment. In sec. 5.2 we give an overview of the channel assignment algorithms that relate most with our proposal. In the following, instead, we give the definition of *total utilization* of a collision domain and motivate why it can be used as optimization criteria for channel assignment algorithms.

4.4.1 A condition on the flow rates of interfering links

The effect of the interference in WMNs is to prevent simultaneous transmissions over neighboring links using the same channel. Hence, the throughput that can be achieved across a wireless link (denoted as *flow rate* in the following) is affected by the amount of traffic transmitted on the neighboring links. Given a set of links L such that no two links can be transmitting simultaneously, our goal is to determine a condition establishing whether the associated flow rates can be

Parameter	Value
$SIFS$	16μs
T_{slot}	9 μs
$DIFS$	$SIFS$ + 2$\cdot T_{slot}$
CW_{min}	15
T_{PLCP}	23 μs
$HLEN$	28B
ACK	14B
R_{ctrl}	6 $Mbps$

Figure 4.2: 802.11a parameters

actually achieved or not. In a given time interval of duration T, each link $e \in L$ with a flow rate $f(e)$ has to carry an amount of data equal to $f(e)T$. If we denote by p the average size of the frame body, such amount of data is transmitted by means of $\frac{f(e)}{p}T$ packets. The time required to transmit a packet is given by the time actually needed to transmit the frame body ($\frac{p}{c(e)}$) plus the overhead introduced by the medium access function (denoted as $\Omega(e)$). Since no two links of the set L can be transmitting simultaneously, a necessary condition for the associated set of flow rates to be achievable is that the sum of the amount of time required by every link to transmit the necessary packets to guarantee the corresponding flow rate be less than T, i.e., $\sum_{e \in L} \left(\frac{p}{c(e)} + \Omega(e) \right) \cdot \frac{f(e)}{p} T \leqslant T$,which yields:

$$\sum_{e \in L} \frac{f(e)}{c(e)} \leqslant 1 - \sum_{e \in L} \frac{f(e)}{p} \Omega(e) \tag{4.1}$$

where the summation in the right hand side (RHS) represents the sum over all the links in L of the overhead related to the transmission of a packet times the number of packets sent per second. The RHS of (4.1) thus represents an upper bound to the sum of the flow to capacity ratios that can be actually achieved. In order to derive a tighter upper bound, transmission failures might be taken into account by multiplying the number of packets sent in the interval of duration T by the average number of transmission attempts. However, we are interested in determining the highest value possible for the sum of the flow to capacity ratios and thus we consider the ideal case of absence of collisions.

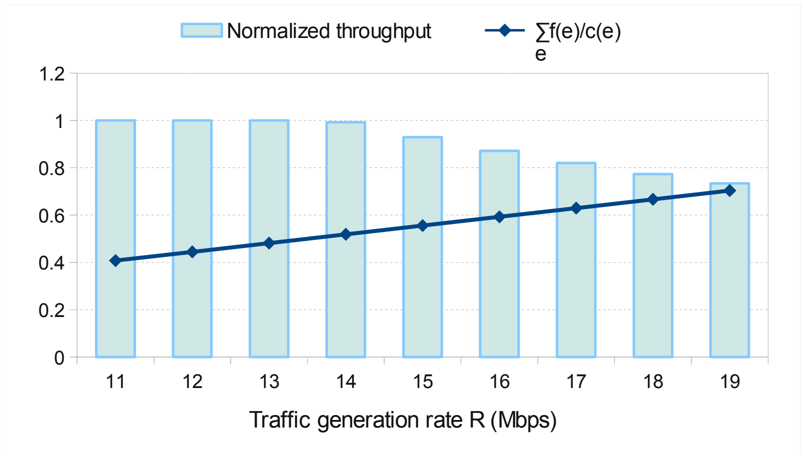


Figure 4.3: ns-3 simulation (UDP traffic)

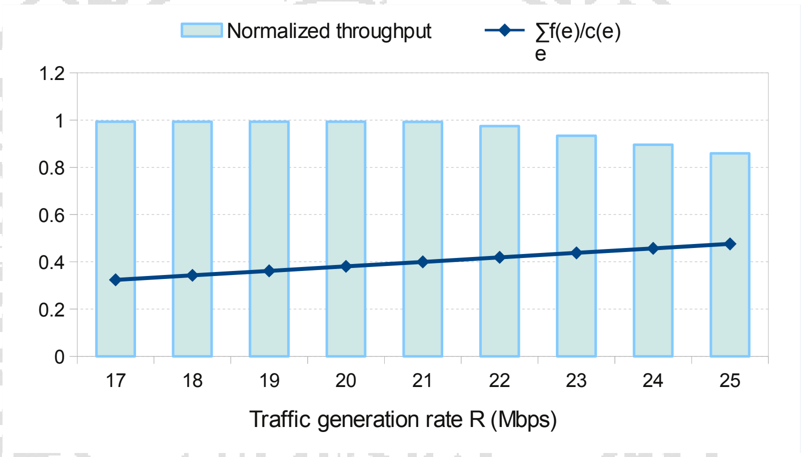


Figure 4.4: ns-3 simulation (TCP traffic)

We now show how to evaluate $\Omega(e)$ in case the basic (i.e., without advanced features such as block ack, transmission opportunity, frame aggregation) 802.11 DCF (Distributed Coordination Function) is used to access the wireless medium. As shown in Fig. 4.1, all the time intervals but the time required to send the frame body are to be considered as overhead:

$$\Omega(e) = DIFS + T_{slot} \cdot N_{slot} + 2 \cdot T_{PLCP} + \frac{HLEN}{c(e)} + SIFS + \frac{ACK}{R_{ctrl}}$$

where $HLEN$ is the size of the 802.11 header, T_{PLCP} is the time to transmit the PLCP (Physical Layer Convergence Procedure) preamble, ACK is the size of the ack frame and R_{ctrl} is the rate used to transmit control frames. Given the assumption that no transmission failure occurs, the contention window of all the stations stays at its minimum value (CW_{min}) and the number of slots (N_{slot}) a station waits in the backoff stage is, on the average, half the value of the contention window.

An upper bound to the sum of the flow to capacity ratios can be easily determined in case all the links in the set L use the same transmission rate (denoted as c). In such a case, $\Omega(e) = \Omega$ and (4.1) yields:

$$\frac{\sum_{e \in L} f(e)}{c} \leqslant \frac{p}{p + \Omega c} \tag{4.2}$$

If we consider the physical layer specified in 802.11a (whose specific values are reported in Fig. 4.2), $c = 54Mbps$ and an average packet payload size of $1400B$ (which leads to a frame body size of $1428B$ if we consider UDP and IP headers), then the RHS of (4.2) is approximately equal to 0.53. Thus, in the considered case, the maximum value for the sum of the flow rates on all the links is about half the physical transmission rate. We performed a simple experiment using the network simulator ns-3 to support this result. Two nodes having a single radio are placed at a distance of 30m (so that they transmit at $54Mbps$) and each of them generates UDP traffic at rate R destined to the other node. According to our analysis, the generated traffic rates are achievable if the sum of the flow to capacity ratios is below 0.53, i.e., $2R/54 \leqslant 0.53$, which implies $R \leqslant 14.3Mbps$. We vary the rate R from 11 to 19 $Mbps$ and measure the average throughput over

30 seconds. The bars in Fig. 4.3 show the measured throughput normalized to the rate of the generated traffic ($2R$), while the symbols indicate the sum of the flow to capacity ratios ($2R/54$). The figure shows that the measured throughput equals the generated traffic rate until $R = 14Mbps$. If we increase the generated traffic rate R beyond the maximum value determined based on our analysis ($14.3Mbps$), the throughput reaches a saturation value ($\simeq 28Mbps$) and then drops more and more below the generated traffic. Such experiment thus confirm that a set of flow rates are actually achievable if the sum of the flow to capacity ratios is below a certain threshold. Beyond such threshold, the achieved throughput is a decreasing fraction of the offered load. Also, the experiment shows that the value of such threshold has been correctly predicted by our analysis.

In case of TCP traffic, we need to also consider the TCP acknowledgments. Assuming that TCP acks traverse the same nodes as the TCP segments (in the opposite direction) and are not piggy-backed by the TCP segments in the reverse direction, we need to consider on each link e of the set L an additional flow rate of $\frac{f(e)}{p}p_{ack}$ (i.e., the number of TCP acks per second is the same as the number of TCP segments sent per second), where $p_{ack} = 40B$. Thus, an upper bound to the sum of the flow to capacity ratios, in case all the links use the same transmission rate and transmit TCP traffic, can be derived from (4.1) as well:

$$\frac{\sum_{e\in L} f(e)}{c} \leqslant \frac{p}{p + p_{ack} + 2\Omega c} \tag{4.3}$$

Considering again the 802.11a physical layer, $c = 54Mbps$ and an average packet payload size of $1400B$ (which leads to a frame body size of $1440B$ if we consider TCP and IP headers), we obtain from (4.3) that the maximum sum of the flow to capacity ratios is about 0.37. We perform a similar ns-3 experiment to validate such result. We still consider two nodes, but there is only one source of TCP traffic. According to our analysis, we expect that the maximum traffic generation rate for the TCP source is $0.37 \cdot 54 \simeq 20Mbps$. We vary the traffic generation rate R from 17 to 25 $Mbps$ and measure the average throughput over 30 seconds. The results (Fig. 4.4) show that the throughput is equal to the generated traffic rate until R is lower than or equal to $21Mbps$ (or, equivalently, the sum of the flow to capacity ratios is lower than 0.40). The fact that the bounds provided by our analysis are exceeded can be likely explained by considering that a TCP ack is not necessarily sent for every single TCP segment, but the transmission of a TCP ack can be delayed to acknowledge more than one segment. Hence, the rate of the TCP acks is lower and there is room for sending more TCP segments. Nevertheless, these simulations show that the maximum achievable sum of the flow to capacity ratios is bounded and the bound can be provided with a good approximation by (4.3).

4.4.2 The total utilization of a collision domain

In the previous subsection we derived (4.1) as a necessary condition for a given set of flow rates associated with links interfering with each other to be actually achieved. In order to apply such condition to a network topology, we need to determine all the sets of links such that no two links in a set can be transmitting simultaneously. For this purpose, we might build a *conflict graph* [38], i.e., a graph where vertices represent network links and edges connect vertices representing interfering links, and find all the maximal cliques in the conflict graph. However, finding all the maximal cliques in a graph is a known NP-complete problem. In order to lower the complexity, we consider the notion of *collision domain* of a link. We define the collision domain of a link $u \overset{c}{\rightarrow} v$ as the set of all the links that interfere with it. Formally, $\mathcal{D}(u \overset{c}{\rightarrow} v) = \left\{ x \overset{c}{\rightarrow} y \in E \mid \frac{G_{uv}P(u \rightarrow v)}{G_{xv}P(x \rightarrow y) + n_v} < \gamma_{c(u \overset{c}{\rightarrow} v)} \right\}$ $\cup \left\{ v \overset{c}{\rightarrow} u \right\}$. Thus, by definition, $u \overset{c}{\rightarrow} v \in \mathcal{D}(u \overset{c}{\rightarrow} v)$. Also, $v \overset{c}{\rightarrow} u$ belongs to $\mathcal{D}(u \overset{c}{\rightarrow} v)$ as a single radio cannot transmit and receive simultaneously. Thus, none of the links in the collision domain of link e can be active at the sime time as e. However, two links in the collision domain of e might be able to transmit simultaneously. It follows that, when applied to the links of a collision domain, (4.1) is no longer a necessary condition for achieving the set of flow rates. Hence, the sum of the flow to capacity ratios over the links of a collision domain can exceed the RHS of (4.1). Also, unlike the previous experiments, links usually have different capacities and hence it is not easy to derive from (4.1) an upper bound to the sum of the flow to capacity ratios. For conciseness, we define the sum of the flow to capacity ratios over the links of the collision domain of link e as the *to-*

tal utilization of that collision domain and denote it by $U_{tot}(e) = \sum_{e_0 \in \mathcal{D}(e)} \frac{f(e_0)}{c(e_0)}$. Nonetheless, as shown in Section 5.5.1, there is still a strong (negative) correlation between the maximum total utilization over all the collision domains and the ratio of the network throughput to the offered load. Indeed, as the maximum total utilization increases beyond a certain threshold, it is more likely that the achieved throughput is below a given fraction of the offered load. The results reported in Section 5.5.1 also show that the previous analysis is able to predict such threshold with a good approximation. Such results motivate us to consider the condition that the total utilization of all the collision domains be below a given threshold as the objective of our channel re-assignment algorithm.

4.4.3 The total utilization as optimization criteria

The analysis of the overhead introduced by the IEEE 802.11 medium access function has led to the identification of a parameter, the *total utilization* of a collision domain, that is shown to have a strong (negative) correlation with the network throughput. Hence, we consider the minimization of the maximum total utilization over all the collision domains as the objective of our channel re-assignment algorithm presented in the following chapter. Also, the aforementioned analysis enables to find a *reference* value for the maximum total utilization that ensures that the network is actually able to carry the offered traffic load. Thus, the condition that the maximum total utilization exceeds such reference value can be used as an indication that a channel re-assignment is needed. We applied such criterium in a simulation conducted with real traffic traces, which showed that re-assigning channels by using our heuristic allows a remarkable throughput increase with respect to the strategy of leaving the channel assignment unchanged (in sec. 5.5.1).

4.5 Routing

Routing in wireless mesh networks, due to the bandwidth limits imposed by interference, has to route traffic on links in order not to exceed the available bandwidth on each link. Routing algorithm does not always succeed in achieving such an objective. Indeed, traditional destination-based routing protocols do not take into account the link bandwidth availability resulting from a given channel assignment and route packets along the shortest paths computed by using certain link metrics. Finding a set of link costs such that a given set of traffic demands are routed so that the link available bandwidths are not exceeded is a difficult problem [39]. Also, such a solution would be tightly coupled to a particular set of traffic demands and the network performance may decrease as the traffic demands vary. Example of such routing algorithms are the OLSR (Optimized Link State Routing) protocol [40] or the AODV protocols, and IEEE 802.11s. We briefly describe these three routing algorithm in the following.

At the end of the chapter we also describe Layer-2.5, a new routing algorithm which is able to take into account the bandwidth limits imposed by the channel assignment algorithm.

4.5.1 OLSR

OLSR (Optimized Link State Routing) is a protocol based on the link state paradigm standardized in the RFC 3626 [41]. It belongs to the family of proactive routing protocols, i.e. protocols that calculate the paths to all the destinations in advance, before they are needed. Each node sends in broadcast to the other the set of the links it is using, which allows all the nodes to have the complete knowledge of the network topology. Having such a knowledge, nodes are able, by applying the Dijkstra algorithm, to determine the cheapest paths to all the destinations of the network.

An implementation of the OLSR protocol which is considered to be stable and is used for many large-scale wireless mesh networks is Unik Olsrd [42]. Unik Olsrd can use both the hop count metric, as specified by RFC, and the ETX (Expected Transmission Count) metric [70]. ETX calulates the quality of each link by taking into account the packet delivery ratio seen on the link, i.e. the percentage of probe packets which are successfully delivered on the link. Experiments have shown that the ETX metric gives better results compared to the hop count metric in most of the scenarios tested [70]. For this reason, the new version of the

OLSR standard which is being defined will probably support also the ETX metric or other metrics based on the calculation of the link quality, e.g. the airtime metric as defined in IEEE 802.11s. Unik Olsrd has been used in this thesis to evaluate the new channel re-assignment scheme, as specified in the 5 chapter.

4.5.2 AODV

The AODV (AdHoc On-demand Distance Vector) protocol, standardized in RFC 3561 [43], belongs to the family of the reactive protocols. Indeed, it calculates a path towards a destination only when it is needed. The node which needs to discover a path sends in broadcast a *route request* message which specifies the destination for which the path is needed. Nodes that receive the message, reply with a *route reply* message, if they know a path to the destination, or re-broadcast the packet after having added the address of the previous hop in the message, until it gets to the destination. The destination, then, by exploiting the information in the route request message is able to send a route reply message to the sender with the information on the path. A disadvantage of AODV, shared with other reactive routing protocols, is that it requires some time to find the path when it is needed, while pro-active protocols like OLSR have all the paths ready in advance. This is balanced, though, by the fact that often AODV generates less overhead, as it just send control messages to discover new paths when they are needed.

4.5.3 IEEE 802.11s

The IEEE 802.11s working group has recently ratified a new standard for wireless mesh networks that defines the behavior of a new MAC (Medium Access Control) able to deliver packets through multiple hops at the datalink layer.

IEEE 802.11s constructs paths using a protocol called HWMP (Hybrid Wireless Mesh Protocol). Such protocol comprises both a pro-active and a reactive component. The pro-active component constructs a forwarding tree whose root is one of the mesh gateways. Such a tree provides the nodes with a path to both the gateway (useful to get to the Internet) and to any other node. Such a path is not the best possible, as there might exist other paths which do not traverse the mesh gateway.

The reactive component is used to construct paths between nodes when needed, i.e. on-demand. The paths are constructed by using the same procedures of AODV, with the difference that packets are sent at the datalink layer instead of at the IP layer.

As the reactive component needs some time to find the required path, packets are sent at the beginning using the path made available by the pro-active part. When the path calculated by the reactive part is ready, packets are sent through the new (better) path.

4.5.4 Layer-2.5 routing

Layer-2.5 (L2.5) is a new routing paradigm which tries to overcome the limits of conventional destination-based routing protocols. As previously stated, conventional destination-based routing protocols fail in respect the bandwidth limits imposed by the channel assignment algorithm. L2.5 is able instead to send traffic on each link of the network in proportion to the defined flow rates, which represent the bandwidth limits imposed by the channel assignment algorithm.

In order to do so, each node u records the amount of bytes sent on each outgoing link, and chooses for each packet the neighbor v among the set of candidate next-hops with the minimum cost. The cost, $\Delta_u(v)$, is calculated as:

$$\Delta_u(v) = \frac{f(u \to v)}{\sum_{\forall u \to i, i \in C} f(u \to i)} - \frac{b(v)}{\sum_{\forall u \to i} b(i)} \tag{4.4}$$

where $f(u \to i)$ represents the flow-rate of the link between u and i, $b(i)$ represents the bytes sent on link between u and i and$\Delta_u(v)$ represents the difference between the desired and the actual utilization of the link between u and v. From the formula of the cost it can be derived that the greater the flow rate, the more traffic is sent on the link.

L2.5 constructs the set of candidate next-hops by considering all the paths between the sender and the receiver that have a length equal or smaller than a certain threshold. Such threshold is equal to the length of the shortest path times an α coefficient, which is an input parameter of the forwarding paradigm. The parameter represents the degree of freedom given to L2.5 to select paths. The greater the α, the greater the allowed length and the more the paths that can be used. More available paths allow to more easily respect the flow rates, as the size of the set of candidate next-hops increases (in average). On the other side, as the allowed path length increases, more resources of the network are used, because packets take in average longer paths. More details can be found in [44]. A problem of L2.5 is that it suffers from a loose control over the paths taken by packets, which can result in *routing cycles*, as discussed in sec. 6.1 where we propose a new forwarding paradigm able to overcome such a limit.

Chapter 5

A channel and rate re-assignment algorithm for wireless mesh networks

5.1 Introduction

As previously stated, routing and channel assignment in wireless mesh networks are not independent problems [37]. Indeed, nodes using the same channel in a neighborhood have to share the channel capacity and hence the amount of bandwidth available on a link depends on how many nodes are using the same channel in the neighborhood. Then, the way channels are assigned affects the amount of bandwidth available on links and hence the channel assignment problem must be jointly studied with the routing problem. However, the joint channel assignment and routing problem has been shown to be NP-complete. Therefore, the proposals that recently appeared in the literature addressing such joint problem solve the channel assignment problem and the routing problem separately. A common approach is to first solve the routing problem, i.e., how to determine the amount of flow (referred to as the *flow rate*) to be routed on each link, and then to solve the channel assignment problem, i.e., how to assign channels in such a way that the resulting bandwidth available on each link exceeds the link flow rate.

Since the assignment of channels depends on the set of flow rates, it should be re-computed upon a variation of the traffic load. However, frequent re-computations of the channel assignment are not desirable. Indeed, a new execution of the channel assignment procedure does not take the current assignment into account and thus will likely return a completely different assignment of channels with respect to the current one. Enforcing the new assignment will thus require changing the channels assigned to several radios. Switching channel on a radio breaks the network connectivity for a much longer time than that required by the radio hardware to shift to the new frequency. Indeed, routing protocols take some time to assess that a previously active link is no longer available or a new link is actually reliable. That is necessary due to the varying conditions of the wireless medium and is done to avoid routing oscillations. Hence, when a radio is assigned a new channel, the routing protocol takes some time to start using the links established on the new channel instead of the links on the previous channel. The consequent packet losses may also induce the TCP entities to decrease the congestion window and increase the retransmission timer, thus lowering the throughput for an additional period. To support such statements, we conducted some experiments in the ORBIT testbed [45], which showed that a channel switch can break the network connectivity for up to 55 seconds.

Switching channel on a radio therefore results into pruning all the links using that radio from the network topology for a certain period of time. Thus, it is clear that the more radios switch channel, the higher the impact on the network performance. For this reason, we designed a simple heuristic that takes the current channel assignment into account and aims to adjust at most a configurable number of channels in order to cope with a variation in the set of flow rates in the best manner possible. Through a thorough simulation study, we show that our heuristic, besides being beneficial in the short term due to the limited number of required channel switches, also ensures a higher throughput in the longer term, with respect to both other channel assignment algorithms and the strategy of leaving the channel assignment unchanged. Indeed, as the channel assignment problem is NP-complete [37], most existing algorithms are heuristics that only provide a sub-optimal solution. Our channel re-assignment algorithm, instead, starts from one such solution and makes some adjustments to find a better solution.

Constraining the number of channel adjustments will likely prevent our heuristic from obtaining the optimal solution for the new set of pre-computed rates. However, neither does an algorithm that assign channels from scratch, since the channel assignment problem is NP-complete. On the other hand, our heuristic allows (by means of a configurable parameter) to keep the number of channel adjustments and hence the computational complexity low.

The proposed algorithm also takes advantage of the availability of multiple transmission rates as provided, e.g., by the current standards defined by the IEEE 802.11 Working Group. We show that the transmission rate on a link can be tuned to minimize the interference experienced due to other links transmitting on the same channel. Hence, our algorithm also attempts to determine the most suitable transmission rate for each network link.

The proposed heuristic requires the knowledge of the complete network topology and the whole set of pre-computed flow rates. Hence, it is suited to a centralized implementation and can be run by a network management station. However, a distributed implementation is also possible, provided that the required information is exchanged among mesh routers. A link state routing protocol such as OLSR (Optimized Link State Routing) [41], for instance, may be easily extended to carry the flow rate and the frequency channel associated with each link included in a Topology Control message. Such values would be propagated to all the mesh routers, thus providing each of them with all the information necessary to execute our proposed algorithm.

The work described in this chapter has been carried on in collaboration with a researcher of the COMICS [22] group [V.1].

The rest of this chapter is organized as follows. In section 5.2 we give an overview of some channel assignment schemes that can be related to this proposal. In section 5.3 we formalize the system model and the channel re-assignment problem. In Section 5.4 we formalize the operation of the proposed algorithm by means of its pseudo-code. In section 5.5 we show the results of the simulation studies and the experiments carried out to evaluate the performance of our algorithm. In section 5.6 we conclude the chapter with a discussion of the benefits of the proposed channel assignment algorithm.

5.2 Channel assignment in literature

The channel assignment problem in multi-radio WMNs has been investigated in the literature recently. Many proposals aim to minimize some network-wide measure of interference and do not study the channel assignment problem in conjunction with the routing problem. For instance, in [46] the goal is to find a channel assignment which minimizes the size of the largest collision domain subject to the constraint that the induced graph must still be K-connected. polynomial time recursive heuristic based on the use of a conflict graph is proposed in [47]. A centralized algorithm is presented in [48] which also takes the traffic generated by mesh clients into account. In [49], an interference-free channel assignment is sought by using superimposed codes. MesTiC [50] is a rank-based channel assignment, where the rank of a node is a function of its aggregate traffic, its number of hops from the gateway and its number of radio interfaces. In [51], both centralized and distributed algorithms are presented, which aim to minimize the number of pairs of links that are interfering. A distributed channel assignment algorithm and a distributed routing protocol are proposed in [52]. Dhananjay et al. [53] present a distributed protocol for channel assignment and routing in dual-radio mesh networks.

Other proposals study the joint channel assignment and routing problem. An iterative routing algorithm based on traffic profiles is proposed in [54]. In [55] an approximate solution for the joint channel assignment and routing problem is developed which optimizes the network throughput subject to fairness constraints. in [55] to produce an interference free link schedule. The problem how to verify the feasibility of a given set of flows between source-destination pairs is investigated in [56]. In [57], a distributed joint channel assignment, scheduling and routing algorithm is presented. In [37], tuning the transmission rate is exploited to present a channel and rate assignment heuristic. In [58] the authors develop a centralized solution to the joint logical topology design, interface assignment, channel allocation and routing problem. For a more comprehensive survey of channel assignment algorithms for wireless mesh networks, we refer the reader to [59]. There has been also some work on the channel assignment problem in wireless sensor networks. Due to hardware limitations, sensors have a single radio interface and thus the proposed algorithms usually assign channels in a dynamic manner. A traffic-aware channel assignment algorithm is proposed in [60], while in [61] a middleware positioned between the MAC and PHY layers is proposed to find the best channel at runtime and communicate it to a single-channel MAC protocol. A protocol to detect the radio interference among nodes and a collision-free TDMA schedule based on the results of such detection are proposed in [62].

All the works mentioned so far, however, do not consider the problem how to re-configure the wireless mesh network after a change in the traffic flows. Such a problem has been tackled in a few papers. The approach in [63] does not consider the standard CSMA/CA access technique but assumes the existence of a link layer synchronization among the nodes which enables them to organize their data transmissions in different time slots with no contention. Hence, the proposed algorithm reconfigures the channel assignment and the link scheduling as a consequence of a change in the traffic matrix. Our approach, instead, assumes the standard contention based access technique and hence does not perform link scheduling. In [64], a distributed channel re-assignment heuristic is proposed that aims to cope with the traffic variation due to mesh clients handoffs. The proposed approach solely makes a channel switch on the edge mesh routers aggregating clients traffic and does not ensure that all the node pairs remain connected after the channel switch, thus requiring changes in the routing tables of the mesh routers involved. With respect to the approach [65], MVCRA-R is an enhanced version under many aspects, as it supports the re-configuration of transmission rates and an improved definition of the link priorities.

5.3 Problem Formulation

We consider the WMN architecture defined in sec. 4.2. The system model of the WMN and the definition of *total utilization* of a collision domain are given in sec. 4.3.

The channel re-assignment problem

Given the values of flow rate f for every link $e \in E$ (see system model in sec. 4.3) and a channel assignment, the channel re-assignment problem is to change the channels assigned to at most a given number of radios so that the total utilization of every collision domain (or, equivalently, the maximum total utilization) is below a given threshold and the network topology is preserved (meaning that there must be a link between every two nodes that were connected before the channel re-assignment). Being equivalent to the channel assignment problem, but with the additional constraint on the number of radio changes, the channel re-assignment problem is NP-complete, too [37]. Hence, it is not possible to determine in polynomial time whether a solution to the channel re-assignment problem exists for a given threshold. Consequently, the heuristic we propose aims to minimize the maximum total utilization over all the collision domains while changing the channels assigned to at most the given number of radios.

In the attempt to minimize the total utilization, we also exploit the availability of multiple transmission rates. Indeed, the total utilization $U_{tot}(e) = \sum_{e_0 \in \mathcal{D}(e)} \frac{f(e_0)}{c(e_0)}$ of the collision domain of link e is also affected by the capacity of all the links in that collision domain. At a first glance, we may conclude that we only have to select the highest transmission rate possible for all the links in order to minimize the total utilization of the collision domain. However, decreasing the transmission rate on link e brings with it a lower SINR threshold, which means the transmission on more links may be compatible with the transmission on e. In general, $\mathcal{D}(e|c(e) = r_i) \subseteq \mathcal{D}(e|c(e) = r_j)$ for $i < j$. Thus, decreasing the transmission rate on link e may help reduce the total utilization of its collision domain. Also, decreasing the transmission rate on a link e has no effect on the composition of the collision domain of the other links (since the transmission power does not change). However, it affects the total utilization of the other collision domains since the ratio $\frac{f(e)}{c(e)}$ increases. Thus, our proposed channel re-assignment algorithm, presented in the next section, adjusts the channel and the transmission rate on each link, while considering the impact on the total utilization of the other collision domains.

5.4 Minimum Variation Channel and Rate Re-assignment Algorithm (MVCRA-R)

In this section we present the MVCRA-R (Minimum Variation Channel and Rate Re-Assignment) algorithm. We show the operation of our algorithm through the pseudo-code in figures 5.1 to 5.5. MVCRA-R is passed the current assignment of channels, the new set $f(e)$ of flow rates and the $MaxNumChanges$ parameter, which determines the maximum allowed number of changes to the channels assigned to the radios. In order to determine what radios need to be assigned a new channel, we first compute the total utilization of all the collision domains as de-

```
MVCRA-R$(G(V,E), \{f(e)\}_{e\in E}, MaxNumChanges, \lambda_0)$
 1  $U_{tot}(e) \leftarrow \sum_{e_0\in\mathcal{D}(e)} \frac{f(e_0)}{c(e_0)} \quad \forall e \in E$
 2  $Q \leftarrow \{e\}_{e\in E}$
 3  $Num\_Changes \leftarrow 0$
 4  while $Q \neq \emptyset$ AND $(Num\_Changes < MaxNumChanges)$
 5      do $(u \overset{c_{old}}{\rightarrow} v) \leftarrow$ EXTRACT_MAX$(Q)$
 6         $\left(c_{sel},\, c(u \overset{c_{sel}}{\rightarrow} v)\right) \leftarrow$ MIN_UTOT$(u, v, \mathcal{C})$
 7         CHANGE_IF$(u, c_{sel})$
 8         CHANGE_IF$(v, c_{sel})$
 9         $E \leftarrow E - \{u \overset{c_{old}}{\rightarrow} v\} \cup \{u \overset{c_{sel}}{\rightarrow} v\}$
10         while $Q_P$ is not empty
11             do $(s \rightarrow t) \leftarrow$ EXTRACT_MAX$(Q_P)$
12                $\mathcal{S} \leftarrow \mathcal{A}(s) \cap \mathcal{A}(t)$
13                if $\mathcal{S} = \emptyset$
14                  then if $\sum_{k\in\mathcal{C}} count_s(k) > \sum_{k\in\mathcal{C}} count_t(k)$
15                         then $\mathcal{S} \leftarrow \mathcal{A}(s)$
16                         else $\mathcal{S} \leftarrow \mathcal{A}(t)$
17                $\left(c_{sel},\, c(s \overset{c_{sel}}{\rightarrow} t)\right) \leftarrow$ MIN_UTOT$(s, t, \mathcal{S})$
18                CHANGE_IF$(s, c_{sel})$
19                CHANGE_IF$(t, c_{sel})$
20                $E \leftarrow E - \{s \overset{c_{old}}{\rightarrow} t\} \cup \{s \overset{c_{sel}}{\rightarrow} t\}$
```

Figure 5.1: Pseudo-code MVCRA-R

termined by the current channel assignment and the new set of flow rates (line 1). All the links of the communication graph are then inserted into a priority queue Q and are extracted one by one (line 4) in decreasing order of priority. The priority of a link l is given by its flow to capacity ratio times the number of links whose collision domain includes l and has a total utilization above a given threshold λ_0. The rationale is that we want to extract first those links that allow as many collision domains as possible to benefit from a channel switch. The $Num\_Changes$ variable holds the current number of channel adjustments and should not exceed the $MaxNumChanges$ parameter.

When a link $u \rightarrow v$ is extracted (we denote by c_{old} the channel it is currently assigned), the goal is to determine a new channel c (independently from the channels currently assigned to u and v), and a new rate r, that minimize the total utilization of its collision domain (line 5). This is achieved by invoking the MIN_UTOT function (fig. 5.2), which analyzes the effects of assigning each of the potential channels to link $u \rightarrow v$ and returns the most convenient one. In particular, in order not to take decisions that might aggravate the total utilization of other collision domains, the MIN_UTOT function also considers, for each channel c in the set $\mathcal{S}$, the total utilization of the collision domain of all the links $x \overset{c}{\rightarrow} y$ which would have $u \overset{c}{\rightarrow} v$ in their collision domain. In case a link $u \overset{c}{\rightarrow} v$ were established, all such total utilizations would be increased by the same amount, i.e., $\frac{f(u \overset{c}{\rightarrow} v)}{c(u \overset{c}{\rightarrow} v)}$. In order to keep track of the effects of assigning a channel c to the extracted link on such collision domains, it suffices to only consider the maximum among such total utilizations, which is denoted by $U'_{max}(c)$ (line 2 in fig. 5.2). If a link $u \overset{c}{\rightarrow} v$ were established, we would also need to consider the total utilization of its collision domain. $U_{tot}(u \overset{c}{\rightarrow} v)$ may be decreased by reducing the transmission rate on $u \overset{c}{\rightarrow} v$, because a lower rate may allow to reduce the size of the collision domain. For the purpose of determining the most suitable rate, the ADJUST_RATE function (fig. 5.3) is invoked. Such a function starts by considering the highest rate possible and then proceeds by iteratively trying lower rates, as long as $U_{tot}(u \overset{c}{\rightarrow} v)$ is greater than $U'_{max}(c)$. We note that the rate is actually decreased only if it allows to reduce the total utilization $U_{tot}(u \overset{c}{\rightarrow} v)$ (line 5 in Fig. 5.3). Then, the

```
MIN_UTOT($u, v, \mathcal{S}$)
1  for each $c \in \mathcal{S}$
2      do $U'_{max}(c) \leftarrow \max_{x \xrightarrow{c} y \,|\, u \xrightarrow{c} v \in D(x \xrightarrow{c} y)} U_{tot}(x \xrightarrow{c} y)$
3         $r_c \leftarrow$ ADJUST_RATE($u \xrightarrow{c} v, U'_{max}(c)$)
4         $U_{max}(c) \leftarrow \max(U'_{max}(c), U_{tot}(u \xrightarrow{c} v))$
5  return $\left( \operatorname{argmin}_{c \in \mathcal{S}} U_{max}(c),\, r_c \right)$
```

Figure 5.2: Pseudo-code MIN_UTOT

```
ADJUST_RATE($e, U_{max}$)
1  $m \leftarrow \max\{i \in 1 \ldots M \,|\, e \in E_I \wedge c(e) = r_i\}$
2  $m_{min} \leftarrow m$, $min \leftarrow U_{tot}(e)$
3  while $m > 1$ AND $U_{tot}(e) > U_{max}$
4      do $m \leftarrow m - 1$, $c(e) \leftarrow r_m$
5         if $U_{tot}(e) < min$
6            then $min \leftarrow U_{tot}(e)$
7                 $m_{min} \leftarrow m$
8  return $r_{m_{min}}$
```

Figure 5.3: Pseudo-code ADJUST_RATE

MIN_UTOT function computes the collision domain of $u \xrightarrow{c} v$ considering the rate returned by ADJUST_RATE and uses the $U_{max}(c)$ variable to hold the maximum between $U_{tot}(u \xrightarrow{c} v)$ and $U'_{max}(c)$. MIN_UTOT returns the channel that minimizes $U_{max}(c)$ and the rate selected for that channel.

Since the channel c_{sel} returned by MIN_UTOT may not be currently assigned to any radio on u and v, the CHANGE_IF function is invoked (lines 7 and 8 in fig. 5.4) to set a radio interface of nodes u and v to c_{sel}. The CHANGE_IF function (fig. 5.4) is passed the node u and the channel c that has to be assigned to one of u's radios. If channel c is already assigned to one of u's radios or u has an available radio, then nothing else needs to be done (lines 1-2). Otherwise, we attempt to assign channel c to the radio of u that causes the least disruption in the network configuration. For this purpose, the MIN_DISRUPT function is invoked (fig. 5.5), which computes, for every channel k currently assigned to node u, the set W_k of links that would

```
CHANGE_IF$(u, c)$
1  if $(c \in \mathcal{A}(u)$ or $|\mathcal{A}(u)| < k(u))$
2     then return
3  $(k, W_k) \leftarrow$ MIN_DISRUPT$(u, c)$
4  $Q_P \leftarrow Q_P \cup W_k$
5  $Q \leftarrow Q - W_k$
6  $\mathcal{A}(u) \leftarrow \mathcal{A}(u) - \{k\} \cup \{c\}$
7  $count_u(c) + +$
8  $\overline{W}_k \leftarrow \left\{u \overset{k}{\to} w \in E \vee w \overset{k}{\to} u \in E\right\} - W_k$
9  for each $x \overset{k}{\to} y \in \overline{W}_k$
10       do $\mathcal{S} \leftarrow \mathcal{A}(x) \cap \mathcal{A}(y)$
11          $\left(c_{sel},\, c(x \overset{c_{sel}}{\to} y)\right) \leftarrow$ MIN_UTOT$(x, y, \mathcal{S})$
12          $E \leftarrow E - \{x \overset{k}{\to} y\} \cup \{x \overset{c_{sel}}{\to} y\}$
13 $Num\_Changes$ ++
```

Figure 5.4: Pseudo-code CHANGE_IF

be disrupted by switching a radio on u from channel k to c. Clearly, all the links between u and the nodes that still share a common channel with u after the channel switch can be easily fixed by using one of the common channels. That happens when a neighbor of u has a radio on channel c or u and its neighbor share more than one channel before the channel switch. As an example, fig. 5.6 illustrates the case where channel 2 has to be assigned to one of the radios on u. In such example (where, for simplicity, links entering u are not shown), $\mathcal{A}(u) = \{1, 3, 5\}$ and $W_1 = \emptyset$ (because $2 \in \mathcal{A}(b)$ and $5 \in \mathcal{A}(u) \cap \mathcal{A}(a)$), $W_3 = \{u \overset{3}{\to} c,\, u \overset{3}{\to} e\}$ and $W_5 = \{u \overset{5}{\to} d\}$ (because $1 \in \mathcal{A}(u) \cap \mathcal{A}(a)$). For each channel k currently assigned to node u, a weight ω_k is computed (line 2), which is composed of two factors. The first factor is a function of the (normalized) number of times channel k has been assigned (by CHANGE_IF) to node u. Thus, the weight of the channels that have been previously assigned to u is higher in order to make it less likely that they are replaced by new channels. The second factor is the sum of the flow to capacity ratio of all the links that would be disrupted by replacing channel k with c on node u. Such a factor accounts not only for the number of links that would be disrupted, but also for the amount of flow they carry. Minimizing the number of

```
MIN_DISRUPT$(u, c)$
1  $W_k \leftarrow \{u \xrightarrow{k} w \in E \vee w \xrightarrow{k} u \in E \mid$
           $(\mathcal{A}(u) - \{k\} \cup \{c\}) \cap \mathcal{A}(w) = \emptyset\}$        $k \in \mathcal{A}(u)$
2  $\omega_k \leftarrow \left(1 + \frac{count_u(k)}{\sum_{k_0 \in \mathcal{C}} count_u(k_0)}\right) \cdot \sum_{l \in W_k} \frac{f(l)}{c(l)}$        $k \in \mathcal{A}(u)$
3  return $\left(\operatorname{argmin}_{k \in \mathcal{A}(u)} \omega_k,\ W_k\right)$
```

Figure 5.5: Pseudo-code MIN_DISRUPT

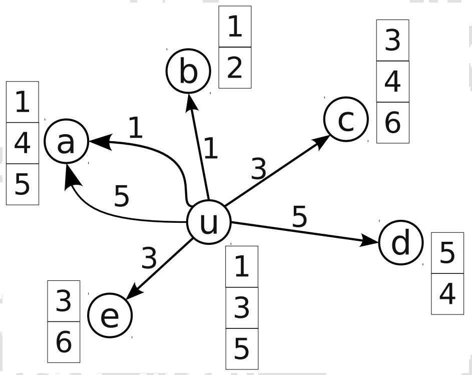


Figure 5.6: Example to illustrate MIN_DISRUPT

links to be repaired as a consequence of a channel switching is important to meet the constraint on the maximum allowed number of radio changes. Accounting for the amount of flow on the pending links is important as well, because a pending link might be assigned a channel that minimizes further disruptions rather than one that minimizes the maximum total utilization over all the collision domains that include it. Hence, it is preferable to disrupt links carrying a lower amount of flow in order to minimize the impact on the total utilization of the collision domains that will include the pending links.

MIN_DISRUPT returns the channel k with the minimum weight ω_k, which then has to be replaced by c on node u. Consequently, all the links on u that were using channel k must be assigned a new channel. In order to do so, CHANGE_IF inserts all the links belonging to W_k into the queue Q_P of the *pending* links, i.e., links that need additional channel switches to be repaired (line 4 in fig. 5.4). Such links are also removed from the queue Q, since they will be processed when extracted from the queue Q_P. Channel c replaces channel k on node u and the counter of the number of times that channel c has been assigned to u is increased by one (lines 6–7). Then, all the links that were using channel k and are not in W_k are fixed by being assigned a new channel. For this purpose, MIN_UTOT is called to determine the common channel between the two end nodes that minimizes the impact on the total utilization of the other collision domains. Finally, $Num\_Changes$ is increased to reflect the channel adjustment on u and CHANGE_IF returns.

As mentioned above, a call to CHANGE_IF from MVCRA-R (lines 7 and 8 in fig. 5.4) may bring some links into a pending state, where they need to be assigned a new channel. MVCRA-R will thus extract the links from Q_P one by one, in decreasing order of priority, until the queue is empty (line 11). The priority of a pending link is its flow to capacity ratio. The goal is to extract (and repair) first those links that will contribute more to the total utilization of the collision domains in which they will be included. For each extracted link $s \rightarrow t$, the set $\mathcal{S}$ of the common channels between s and t is determined. If $\mathcal{S}$ is empty, we necessarily need to change a channel on either s or t. In such a case, $\mathcal{S}$ is filled with the channels of the node that has experienced the highest number of

Table 5.1: Network topologies characteristics

Topology	Nodes	Radios	Average node degree	Area (m^2)
A	22	57	4.36	125×155
B	22	57	4.54	185×235
C	28	75	5.35	195×210

channel switches (line 14), so that a channel is changed on the other node. The MIN_UTOT procedure is invoked to determine the channel of S and the rate minimizing the resulting maximum total utilization. Then, CHANGE_IF is called to actually assign the selected channel to one of the radios on s and t. Clearly, one or both of these calls (depending on whether or not s and t shared a common channel) return immediately, because one or both of the end nodes already have a radio on the selected channel. Finally, the extracted pending link is switched to the selected channel (line 20). When MVCRA-R ends, the queue of the pending links is empty, thus ensuring that all the links have been assigned a channel and hence the network topology is preserved.

5.5 Performance Evaluation

In this section we present the results of the simulation study and the experiments we carried out to evaluate the performance of MVCRA-R algorithm. The goal of the simulation study is to show that updating the network configuration by running MVCRA-R allows to increase the network throughput with respect to both leaving the channel assignment unchanged and updating the network configuration by running a channel assignment algorithm that ignores the current configuration. We remark that we do not simulate the transient stage when radios switch channels (results may be affected by inaccuracies in the simulator). Instead, simulations start from the new network configuration determined by the channel (re-)assignment algorithms. Thus, simulations aim at evaluating the throughput in the long term.

The experiments we conducted with real hardware, instead, aim to gain some insight into the effects of switching channels. We show that a large number of simultaneous channel switches severely impacts the network performance, thus justifying our objective of limiting the number of channel switches.

5.5.1 Simulation study

We conducted a simulation study to compare MVCRA-R to MVCRA (Minimum Variation Channel Re-Assignment) [65] and FCRA (Flow-based Channel and Rate Assignment) [37] in terms of maximum total utilization, number of radios that have to switch channel and average network throughput. FCRA is a greedy channel assignment heuristic that extracts all the links one-by-one and assigns each link the channel that currently minimizes the maximum total utilization among all the collision domains including the extracted link.

Simulation setup

We consider three network topologies (whose main parameters are reported in Table 5.1) where each node is equipped with two or three radios. Given the planar coordinates of the nodes, a software we implemented on our own is used to build the network topology based on the interference model described in Section 5.3. We assume the gain G_{uv} of the radio channel between u and v to be the reciprocal of the square of the distance between u and v and the thermal noise to be -20dbm. The SINR thresholds are set to allow a rate of $54Mbps$ when the nodes are within $30m$, $48Mbps$ within $32m$, $36Mbps$ within $37m$, $24Mbps$ within $45m$, $18Mbps$ within $60m$, $12Mbps$ within $69m$, $9Mbps$ within $77m$ and $6Mbps$ within $90m$. We assume 6 non-overlapping channels are available.

The initial set of flow rates is determined as follows. A subset of mesh nodes is identified as source or destination of traffic flows. Each source-destination pair is associated with a demand, whose amount of traffic is initially determined according to a random variable, as specified below. Each traffic demand is routed along either the shortest path or the three shortest paths [66] between the source and the destination. The sum of the amount of traffic routed on a link over all the source-destination pairs determines the flow rate on that link. FCRA is then used to compute the initial channel assignment. After a variation in the traffic demands, the way flows are routed is not changed. Hence, the share of a flow rate associated with a traffic demand is multiplied by the ratio of the new amount to the previous amount of that traffic demand.

We consider two types of traffic variation, denoted as “Increase” and “Swap”. There are 8 traffic demands in both cases. In the Increase strategy, the initial amount of traffic (denoted by L) is the same for all the demands and then the amount of traffic of each demand is scaled by a factor derived from a uniform distribution $U(0.5 + \alpha, 2\mu - 0.5 - \alpha)$. The mean value of such distribution is μ, which is chosen such that $\mu L = 4$, i.e., the average amount of traffic of a demand after the variation is $4Mbps$ in all the cases. We performed simulations for $L \in \{1.5, 2, 2.5\}$ and $\alpha \in \{0, 0.1, 0.2, 0.3\}$. In the Swap strategy, the initial amount of traffic of each demand is derived from a probability distribution. We considered 10 different distributions in total: $U(1, 5)$, $U(1, 6)$, $U(1, 7)$ and mixtures (with different probabilities) of two uniform distributions such as $U(1, 2)$ and $U(3, 4)$ or $U(1, 2)$ and $U(5, 6)$. Then, the amounts of traffic of all the demands are swapped, in the sense that the demand with the highest amount of traffic gets the minimum amount of traffic, the demand with the second highest amount of traffic gets the second minimum amount of traffic, and so on. The network throughput is measured by means of simulations conducted with the *ns-3* network simulator. The physical layer specified in IEEE 802.11a is used for all the simulations. Each simulation lasts 60 seconds.

Performance of MVCRA-R with different values of $MaxNumChanges$

The aim of this set of simulations is to evaluate the performance of MVCRA-R with different values of $MaxNumChanges$: 5, 10 and 15. The traffic variations described in the previous subsection (Increase and Swap) have been simulated in each of the three topologies considered and for each of the two strategies to route the initial traffic demands. MVCRA-R has been used to compute the new channel assignment starting from the initial channel assignment and the set of flow rates as of after the traffic variation.

Figure 5.7 (left side) shows the distribution of the maximum total utilization

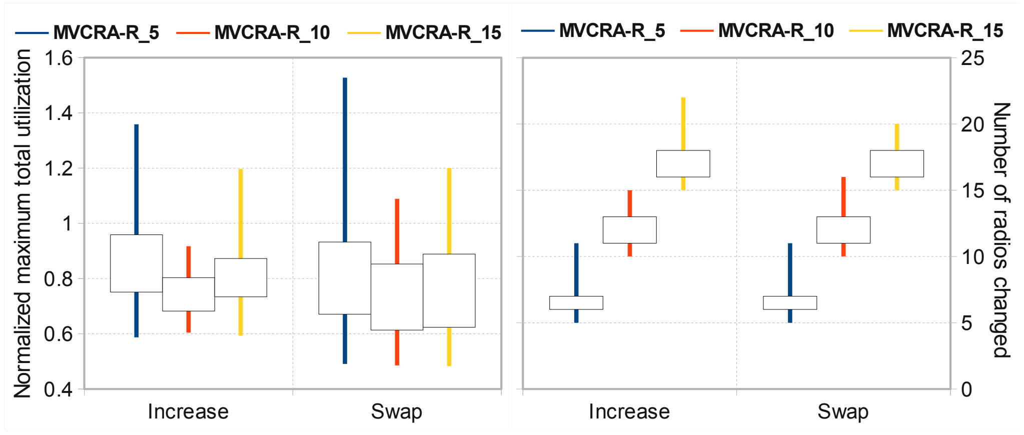


Figure 5.7: Performance of MVCRA-R with different values of $MaxNumChanges$

achieved in all the simulations (for each of the two types of traffic variation) normalized to the maximum total utilization resulting from leaving the channel assignment unchanged. Thus, a normalized maximum total utilization below 1 means that re-assigning the channels enabled a reduction in the maximum total utilization. A vertical line spans from the minimum to the maximum value over all the simulations, while a white box spans from the first quartile to the third quartile. Figure 5.7 (right side) shows the distribution of the number of radios actually changed by MVCRA-R in all the simulations.

It can be observed that MVCRA-R is able to meet, with a good approximation, the constraint on the maximum allowed number of radio changes, for all the values of $MaxNumChanges$ we tested. Regarding the maximum total utilization, the performance of MVCRA-R is worse when it is allowed to change at most 5 radios. Indeed, we can observe that in some cases the achieved maximum total utilization is more than 20% higher than the maximum total utilization obtained by leaving the channel assignment unchanged. The best performance is achieved for $MaxNumChanges$ equal to 10. Indeed, raising $MaxNumChanges$ to 15 does not bring any benefit, thus showing that changing 10 radios is sufficient to improve the maximum total utilization in the topologies and for the traffic loads we considered. In the following, we implicitly assume that MVCRA-R is used with $MaxNumChanges$ equal to 10.

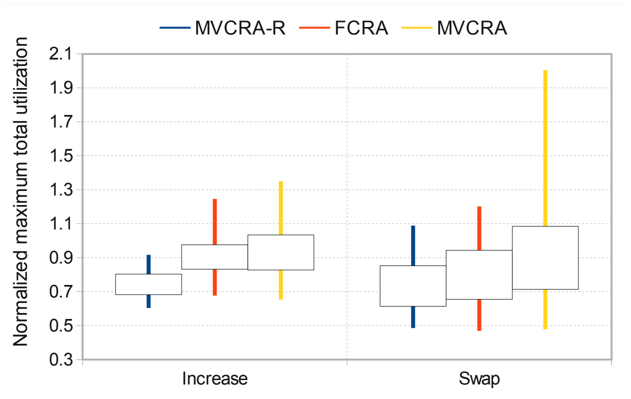
Single variation in the traffic demands

Our goal is to evaluate the gain achieved by different channel assignment algorithms with respect to the strategy of leaving the channel assignment unchanged when a (single) variation in the traffic demands occurs. The traffic variations described in the previous subsection (Increase and Swap) have been simulated in each of the three topologies considered and for each of the two strategies to route the initial traffic demands. We evaluate how MVCRA-R, FCRA and MVCRA react to each such traffic variations by feeding them with the previous channel assignment (ignored by FCRA) and the set of flow rates as of after the traffic variation.

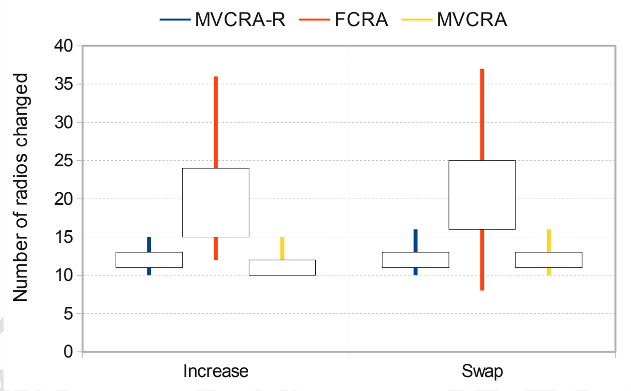
Figure 5.8a shows the distribution of the maximum total utilization achieved by each algorithm normalized to the maximum total utilization resulting from leaving the channel assignment unchanged. The best performance is achieved by MVCRA-R, which achieves a 25% (on the average) reduction in the maximum total utilization with respect to the strategy of leaving the channel assignment unchanged, both in the Increase and in the Swap cases. FCRA achieves, on the average, a 10% (20%) reduction in the Increase (Swap) case, while MVCRA achieves a 7% (10%) reduction in the Increase (Swap) case.

Figure 5.8b shows the distribution of the number of radios changed after each traffic variation. It can be observed that MVCRA-R and MVCRA make approximately the maximum allowed number of changes (about 11.5 on the average both in the Increase and in the Swap cases), while FCRA changes 20 radios on the average, both in the Increase and in the Swap cases.

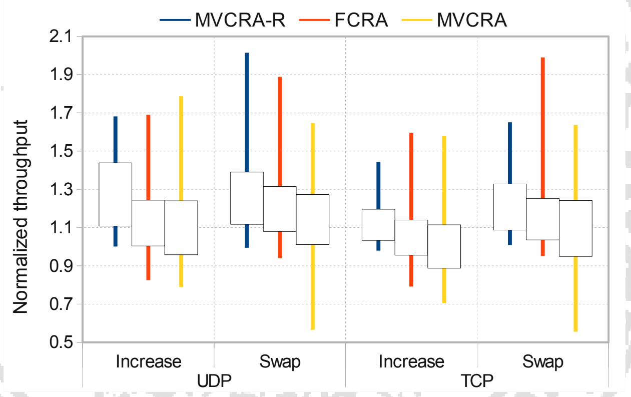
We conducted ns-3 simulations to evaluate the throughput achieved in the configurations computed by each of the algorithms after each traffic variation. The distribution of the average throughput over the whole simulation time is reported in Fig. 5.8c for both UDP and TCP traffic. The results show that MVCRA-R enables a throughput increase with respect to both the other algorithms and the strategy of leaving the channels unchanged. Indeed, for UDP traffic, MVCRA-R



(a) Normalized max total utilization



(b) Number of radios changed



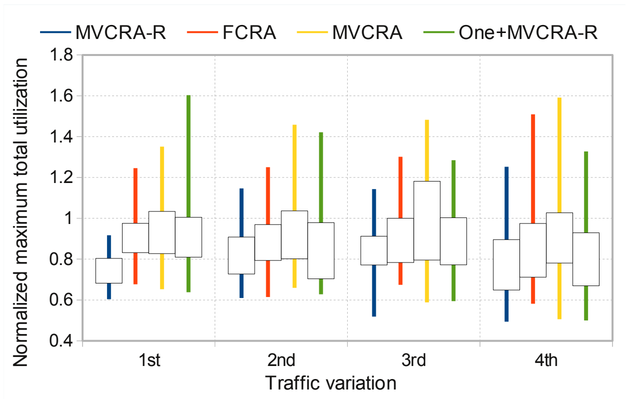
(c) Normalized throughput

Figure 5.8: Single traffic variation

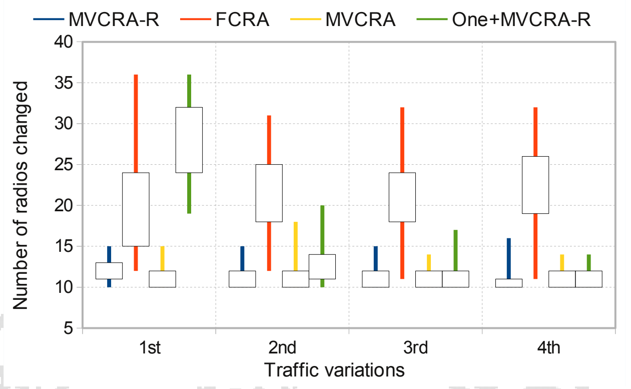
achieves, on the average, a throughput increase of 28% in the Increase case and of 24% in the Swap case, while FCRA achieves, on the average, a throughput increase of 14% (Increase) and of 21% (Swap). MVCRA, instead, achieves, on the average, a throughput increase of 10% (Increase) and of 14% (Swap). For TCP traffic, MVCRA-R achieves, on the average, a throughput increase of 12% in the Increase case and of 23% in the Swap case, while FCRA achieves, on the average, a throughput increase of 8% (Increase) and of 17% (Swap). MVCRA, instead, achieves, on the average, a throughput increase of 2% (Increase) and of 10% (Swap).

Repeated variations in the traffic demands

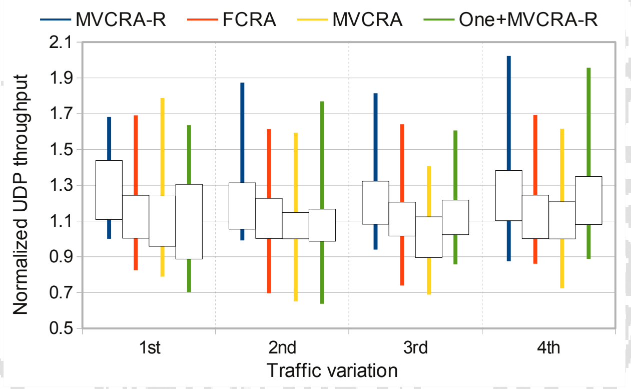
The aim of this set of simulations is to evaluate the gain achieved by different channel assignment algorithms with respect to the strategy of leaving the channel assignment unchanged when repeated variations in the traffic demands occurs. The first variation is that described in the previous subsection under the Increase strategy. The second variation consists in scaling the amount of traffic of each demand by a factor derived from a uniform distribution $U(0.5, 1)$, the third from a distribution $U(0.75, 1.75)$, the fourth from a distribution $U(0.5, 1.5)$. For every variation, each algorithm is given the configuration it computed at the previous traffic variation. $One + MVCRA\text{-}R$ denotes the results achieved by MVCRA-R when the initial configuration is such that each node uses just one radio set to a common channel. Figures 5.9a and 5.9c show the maximum total utilization and average throughput normalized to those achieved with the initial channel assignment computed by FCRA. It can be observed that MVCRA-R outperforms the other algorithms after all the traffic variations. Also, the performance of $One + MVCRA\text{-}R$ rapidly becomes comparable to the best one, thus showing that MVCRA-R is able to adapt to the current configuration and rapidly recover even from very poor configurations. Figure 5.9b shows the number of radios changed after each variation. We note that MVCRA-R, when starting from a configuration where a single channel is used, does not meet the constraint on the maximum allowed number of radio changes. This result shows that MVCRA-R,



(a) Normalized max total utilization



(b) Number of radios changed



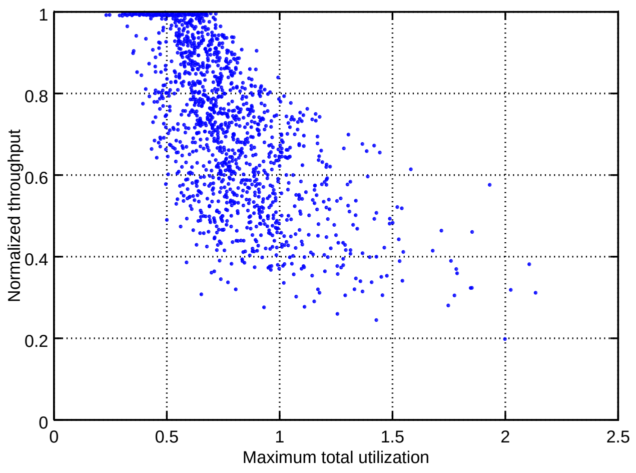
(c) Normalized UDP throughput

Figure 5.9: Repeated traffic variation

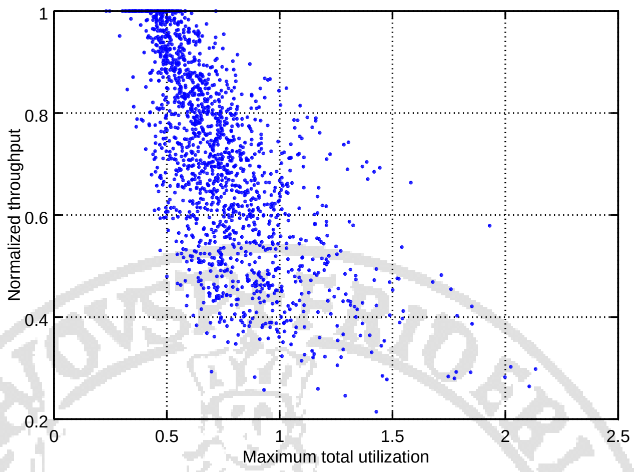
when given a poor network configuration, prefers to rapidly decrease the maximum total utilization, even though that requires to change a higher number of radios. However, it can be observed that after the subsequent traffic variations $One + MVCRA\text{-}R$ tends to change a number of radios comparable to MVCRA-R and MVCRA.

Correlation between maximum total utilization and throughput

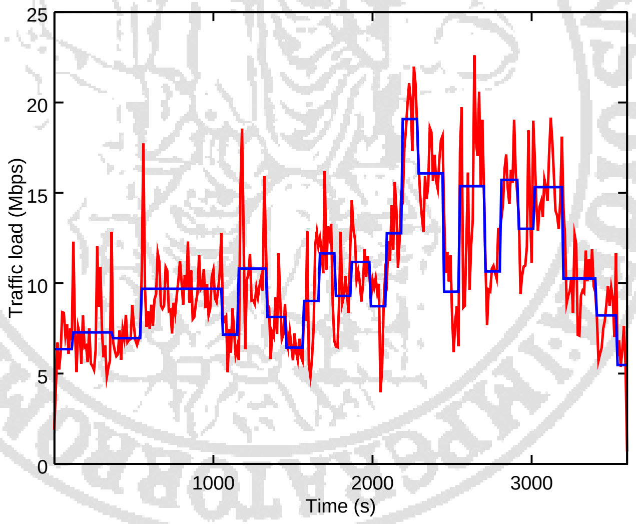
In this section, we present all the simulations we performed (whose results have been shown in the previous subsections) and, for each simulation, we relate the maximum total utilization with the average throughput normalized to the total amount of all the traffic demands. The results are shown in Fig. 5.10a (UDP traffic) and Fig. 5.10b (TCP traffic). It can be observed that, as the maximum total utilization increases, the achieved throughput tends to be a smaller fraction of the traffic demands. In order to measure the statistical dependence between these two variables, we computed the Spearman rank correlation coefficient [67]. Such coefficient ranges from -1 to 1 and indicates how well the relationship between two variables can be described using a monotonic function. A value of 1 (-1) denotes a perfect monotone increasing (decreasing) relationship. A value between 0.5 and 1 (-0.5 and -1) denotes a strong positive (negative) correlation. In our analysis, we got a value of -0.785 for UDP traffic and -0.755 for TCP traffic, thus indicating the strong negative correlation between the maximum total utilization and the achieved throughput (normalized to the total traffic load). Such result substantiates our choice to consider the minimization of the maximum total utilization as the objective of our channel re-assignment algorithm. Also, this analysis confirms that (4.2) and (4.3) can be used to provide reference values for the maximum total utilization that ensures that the network is actually able to carry (a high fraction of) the traffic load. Indeed, the RHS of (4.2) yields 0.53 with a packet payload size of $1400B$ (which is the value used in our simulations). Despite the network links use distinct transmission rates and the collision domains are such that two links may be active simultaneously, it turns out that, for all the simulations where the maximum total utilization is below 0.53, in the 84.5% of the cases the achieved



(a) Correlation (UDP traffic)



(b) Correlation (TCP traffic)



(c) Traffic traces total load

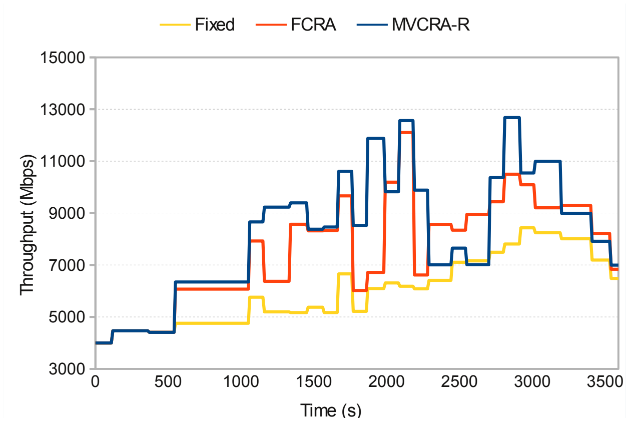
throughput is over 95% of the traffic load and only in the 3% of the cases the achieved throughput is less than the 70% of the traffic load. Also, if we only consider the simulations where the maximum total utilization is between 0.53 and 0.70, it turns out that only in the 32.3% of the cases the achieved throughput is over 95% of the traffic load. A similar trend is observed for the simulations with TCP traffic.

Simulation with real traffic traces

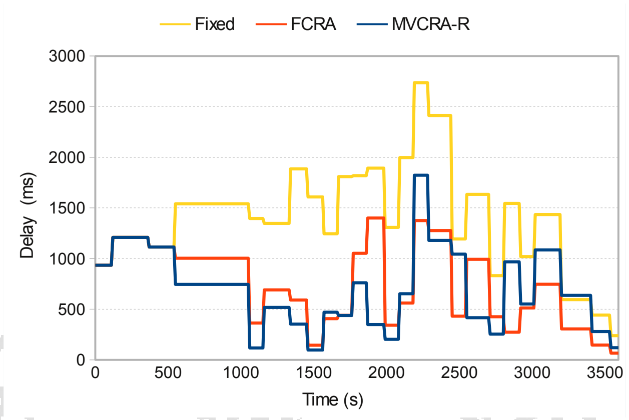
We performed a simulation study where the traffic injected into the network is based on real traffic traces. We considered six traffic traces collected at the gateway router of the wireless network at the UCSD (University of California, San Diego) Computer Science building [68]. Each of such traces records the traffic collected in one hour. For each trace, we only considered TCP packets and classified each of them as *upstream* or *downstream*. To this end, we identified all the TCP SYN segments and recorded the corresponding 4-tuple (IP source address, source port, IP destination address, destination port). Then, all the TCP packets matching a 4-tuple have been marked as upstream (TCP connections have been likely opened by the hosts of the wireless network), while TCP packets having source and destination IP addresses and ports swapped with respect to a 4-tuple have been marked as downstream.

We considered topology C (Table 5.1) and selected two nodes as gateways. Three other mesh nodes act as *aggregation* nodes. Each pair of aggregation node and gateway is associated with a traffic trace. In particular, upstream packets are sent from an aggregation node to a gateway, while downstream packets are sent from a gateway to an aggregation node. In such a way, it is as though the wireless mesh network were used as access network by the hosts of the UCSD wireless network. We use the term traffic demand to generically refer to the upstream or downstream flow deriving from a traffic trace.

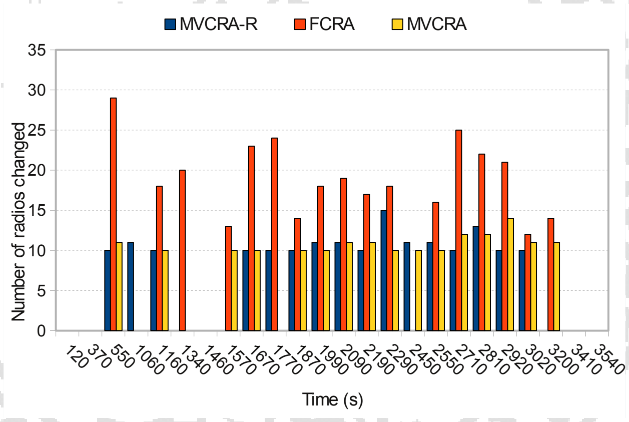
We considered time slots of duration 10 seconds and computed the average traffic load (deriving from all the six traffic traces) over each time slot. Then, consecutive time slots with similar traffic load are aggregated into time intervals.



(a) Average throughput



(b) End-to-end delay



(c) Number of radios changed

Figure 5.10: Simulation with real traffic traces

The average traffic load in each time slot and in each of the 25 time intervals is shown in Fig. 5.10c. The initial flow rates are computed by routing the average load of each traffic demand over the first time interval along the three shortest paths between the corresponding pair of aggregation node and gateway, while the initial channel assignment is computed by FCRA. The initial channel assignment is kept for the duration of the first time interval. At the beginning of every time interval, the maximum total utilization resulting from the channel assignment used in the previous time interval and the average load of each traffic demand over the next time interval is computed. If such maximum total utilization exceeds a given threshold, a new channel assignment is computed. The threshold we use in our experiment is 0.4, i.e., the value obtained by substituting the average packet size resulting from the traffic traces we considered ($850B$) into the RHS of (4.2). We implemented an ns-3 module to generate traffic according to a given trace. We used static routing in our simulation in order to avoid the transient throughput loss due to the time required by nodes to discover new links established on new channels (see Section 5.5.2). We did so because such throughput loss is dependent on the particular routing protocol used and we want to evaluate the throughput achieved in a steady state.

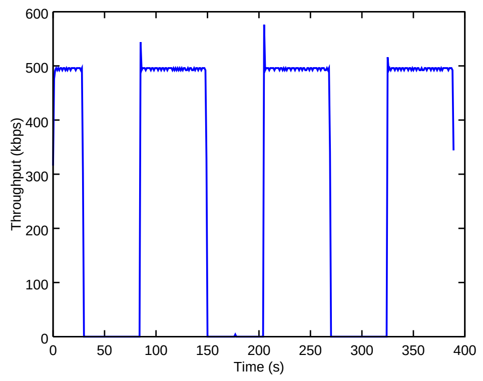
The results of the experiment we conducted are shown in Fig. 5.10. The average throughput achieved by each channel assignment algorithm (except MVCRA, which is omitted for clarity) is shown in Fig. 5.10a. Here, “Fixed” refers to the strategy of leaving the channel assignment unchanged. It can be observed that the fixed strategy achieves the lowest average throughput in every time interval, while MVCRA-R achieves the highest average throughput in almost all the time intervals. The highest average throughput over all the simulation duration is achieved by MVCRA-R (8163 $Mbps$), which attains a 9% throughput increase with respect to FCRA, 13% with respect to MVCRA and 35% with respect to the fixed strategy. The average delay experienced by the packets sent in each time interval is shown in Fig. 5.10b. It can be observed that the fixed strategy leads to the highest average delay, while MVCRA-R enables a 50% reduction, FCRA a 46% reduction and MVCRA a 40% reduction.

Fig. 5.10c shows the number of radios changed by each algorithm at the beginning of each time interval. MVCRA-R and MVCRA change 10.81 radios on the average, thus roughly satisfying the constraint on the maximum allowed number of radio changes and requiring far less radio changes than FCRA (19 on the average). Finally, as far as the maximum total utilization is concerned, it turns out that MVCRA-R achieves the mininum average value (0.43), followed by MVCRA (0.46), FCRA (0.48) and the fixed strategy (0.65).

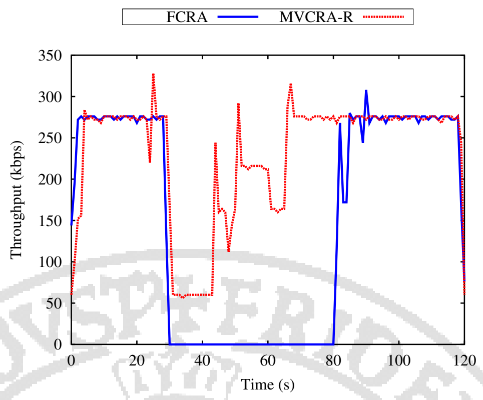
The results of the simulation conducted with real traffic traces showed that MVCRA-R outperforms FCRA (higher throughput and far less radio changes required) and enables a considerable gain in throughput with respect to the leaving the channel assignment unchanged.

5.5.2 Experimental results

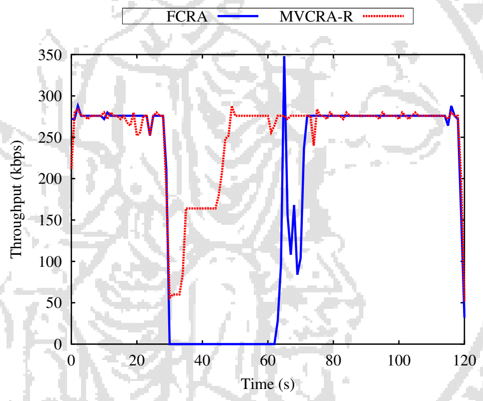
We performed experiments using the ORBIT wireless testbed [45] to gain some insight into the effects of switching channels on the network performance. Each node of the testbed has two IEEE 802.11 interfaces and runs Linux (kernel version 2.6.35). We selected OLSR (Optimized Link state Routing), the well-known implementation of the OLSR protocol, which we have described in sec. 4.5.1 standardized in RFC 3626 [41]. We first present the results of a simple experiment involving two nodes. The sender (S) generates TCP traffic at the constant rate of $500kbps$ destined to the receiver (R). The sender has two radios set on channels 36 and 44 (802.11a), while the receiver has a radio set on channel 36 (802.11a). A link is thus established on channel 36. After 30 seconds, the radio on the receiver switches from channel 36 to channel 44. Then, every two minutes we keep switching the radio on the receiver between channel 36 and 44. The average throughput over time slots of 1 second is shown in Fig. 5.11a. We can notice that, as soon as the first channel switch takes place, there is no connectivity between the sender and the receiver for about 55 seconds. Such a delay can be explained as follows. We recall that OLSR nodes periodically broadcast HELLO messages that are used to determine the quality of the link to a neighbor. In order to avoid fluctuations due to transient noise/interference, RFC 3626 introduces



(a) Simple experiment



(b) Larger topology (TCP traffic)



(c) Larger topology (UDP traffic)

Figure 5.11: Throughput measured in the experiments in the ORBIT testbed

the hysteresis strategy and two thresholds: an established link is no longer used when its quality drops below the lower threshold, while a new link is considered established when its quality exceeds the higher threshold. Hence, a number of HELLO intervals must elapse before the OLSR daemon on the sender considers the link on channel 36 as lost and the new link on channel 44 as established. Until that happens, the sender keeps transmitting using the radio set on channel 36 and thus the packets are not received, as the receiver switched its radio to channel 44. Things are also made worse by TCP, which keeps increasing the retransmission timeout during the absence of connectivity. Thus, when the network connectivity is restored, the sender TCP entity may be waiting for the re-transmission timer to expire and thus delays the re-transmission of the queued segments until after the timer expires. Indeed, repeating the experiment with UDP traffic showed that the throughput is null for about 42 seconds.

The results shown in Fig. 5.11a are obtained by using the default value for the HELLO interval (2 seconds). Clearly, all the protocol parameters can be tweaked in order to reduce the time taken by the routing protocol to switch to using the new link. For instance, we repeated the previous experiment by using an HELLO interval of 1 second and obtained that the connectivity is lost for 28 seconds instead of 55. However, reducing the HELLO interval increases the overhead. Also, the hysteresis thresholds might be changed, but considering a link as established or lost after a few successful/unsuccessful HELLO message transmissions might lead to routing instability. Indeed, an active link may suffer bursty transmission failures due to collisions or transient noise/interference. For the same reason, reactive protocols such as AODV [43] (and hence the default path selection protocol defined in IEEE 802.11s) take some time to assess that a link is not active anymore before initiating a new route discovery procedure. We believe that identifying the links that are established/lost following a radio channel switch by means of the usual procedures based on the link quality is not efficient. A cross-layer exchange of information between the routing and MAC layers may help speed up the process of switching to the newly established link. Investigating such a possibility is left for future work.

We now analyze the effects of multiple channel switches in a bigger network. We used 13 nodes in the ORBIT testbed, two of which are used as senders and other two are used as receivers. Each sender generates two traffic flows destinated to each of the receivers. One of the nodes acts as channel assignment server, in the sense that it communicates (via a TCP connection) to each of the other nodes the new channels their radios must be set to. The actual channel switching is triggered by the reception of a UDP message that is sent by the channel assignment server 30 seconds after the start of the experiment and re-broadcast by every node receiving it (before switching channels). In the attempt to speed ud the recovery from the breakages caused by switching channels, we enabled the use of the link quality as link metric instead of the simple hop count provided by RFC 3626[1]. Indeed, using the hop count metric, the current path is only replaced when one of its links is marked as lost, while, using the link quality, a path with a better quality than the current one can be preferred even before a link of the current path is marked as lost.

The throughput measured in case of TCP traffic is shown in Fig. 5.11b. We conducted two experiments. In the one case, channels are re-assigned by FCRA, which results in 14 channel switches. In the other case, channels are re-assigned by MVCRA-R, which results in 3 channel switches. We can observe that, in the case of 14 channel switches, the time required to restore the steady state throughput is about 55 seconds, i.e., what we measured in the simple experiment. For most of such time interval, the network throughput is null. The reason is that the (approximately) simultaneous switching of a high number of radios breaks the connectivity between many nodes (as illustrated by the simple experiment). In our case, there is no path between the senders and the receivers made of nodes that keep their connectivity during the channel switching and hence the throughput is null. In the experiment where only 3 radios are changed, we can observe that the throughput drops to 60$kbps$ right after the channel switching. That happens because the path between a single sender-receiver pair is not affected by the channel switches. Also, the small number of radio changes allows to find alter-

[1]A second version of OLSR is being worked on within the IETF that enables the use of metrics other than the hop count

nate paths for the other sender-receiver pairs that do not include links affected by the channel switches. Those alternate paths are preferred, and hence used, by the routing protocol as soon as the quality of the current paths drops below their quality. Actually, that happens before the links using the new channels are considered established. The result is that restoring all the sender-receiver pairs takes about 35 seconds and thus is quicker than in the previous case. Figure 5.11c shows the throughput measured in case of UDP traffic. It can be observed that the throughput is restored quicker than the case of TCP traffic, thus confirming that the TCP mechanisms further delay the restoration of the steady state throughput. Also, the behavior in case of 3 and 14 channel switches is similar to those observed with TCP traffic.

Though the experimental results reported in this section are not exhaustive, given the wide variety of parameters involved (routing protocol, routing protocol settings, transport layer variants, topology, etc.), we believe that they make evident that limiting the number of radios switching channel is beneficial because more paths are unaffected and can be used by the routing protocol to replace those affected by the link breakages.

5.6 Conclusions

In this chapter we presented the Minimum Variation Channel and Rate Re-Assignment (MVCRA-R) algorithm, which takes the current channel assignment and the new set of flow rates into account and attempts to minimize the maximum total utilization over all the collision domains while constraining the number of radios that can be assigned a new channel. With respect to MVCRA, MVCRA-R leverages the possibility to adjust the link transmission rates and presents some enhancements such as an improved definition of the link priorities. We performed extensive simulation studies that confirmed that MVCRA-R roughly meets the constraint on the maximum allowed number of radio changes and outperforms both MVCRA and a channel assignment algorithm such as FCRA in terms of maximum total utilization and network throughput. The simulation studies also confirmed the strong correlation between the maximum total utilization and the throughput, thus supporting our choice for the objective function of MVCRA-R. Also, I conducted experiments in a real wireless testbed to evaluate how switching channels affects the network performance. We believe that investigating measures to limit such impact constitutes an interesting subject for future work.

Submitted paper

[V.I] S. Avallone, G. Di Stasi. “A Traffic-Aware Channel and Rate Re-Assignment Algorithm for Wireless Mesh Networks” Requiring major revision to IEEE/ACM Transactions on Mobile Computing.

Chapter 6

A new MPLS-based forwarding paradigm for wireless mesh networks

6.1 Introduction

The routing protocol in a wireless mesh netowork is in charge of routing packets in such a way not to exceed the given available bandwidth on each link. Unfortunately, traditional destination-based routing protocols do not take into account the link bandwidth availability resulting from a given channel assignment and route packets along the shortest paths computed by using certain link metrics. Finding a set of link costs such that a given set of traffic demands are routed so that the link available bandwidths are not exceeded is a difficult problem [39]. Also, such a solution would be tightly coupled to a particular set of traffic demands and the network performance may decrease as the traffic demands vary.

To overcome the shortcomings of traditional destination-based routing protocols, a new Layer-2.5 forwarding paradigm was proposed (see sec. 4.5.4). In L2.5, forwarding decisions are not taken by looking up the routing table (which is not needed by L2.5), but are based on two objectives: i) balance the traffic among the outgoing links in proportion to their available bandwidth; ii) guarantee that all the packets reach the destination in a predetermined maximum number of hops. To achieve the latter objective, a node needs to know the minimum hop count of each neighbor to every destination and each packet carries a Time-To-Live field (initialized to the maximum number of hops) in the additional L2.5 header. Besides taking the bandwidth available on links into account, L2.5 has the potential for fast recovery from node/link failures, given that it is not needed to wait for nodes to re-compute the routing tables, but the node adjacent to the failed node/link can promptly blacklist the failed neighbor and balance the traffic among the remaining outgoing links. L2.5, however, suffers from a loose control over the paths taken by packets. Indeed, once the maximum path length has been fixed, a packet can take *any* path with length not exceeding the maximum one. Consequently, L2.5 fails to ensure that the paths followed by packets are cycle-free. Also, the performance of L2.5 is dependent on the available bandwidth values associated with the network links, and hence it may degrade if they are not suited to the actual traffic load.

In this chapter, we present a novel forwarding paradigm for multi-radio WMNs based on Multi-Protocol Label Switching (MPLS) [2] with the purpose to overcome the aforementioned limitations of both L2.5 and the destination-based routing protocols. The first contribution is the definition of the MPLS *splitting* policy, a new, standard-compliant, MPLS mechanism that enables each intermediate node to split the incoming traffic belonging to a specific Forwarding Equivalence Class (FEC) among a predefined set of neighbors according to predefined *split ratios*. As a result, different packets of a given FEC follow distinct paths between the ingress and egress nodes, which allows to better balance the traffic across the network with respect to single path routing protocols. Also, as in L2.5, the availability of multiple next hops for a given destination enables a fast local restoration in case of single node/link failures. Unlike L2.5, however, our approach allows a tight control over the paths taken by packets, thanks to the use of MPLS.

A fundamental role in the MPLS splitting policy is clearly played by the set of split ratios. The second contribution is the definition of an approach to compute, given the current channel assignment, a set of split ratios that ensure high throughput despite variations in the traffic load. To this end, rather than sticking to a given set of traffic demands, we adopt the *hose* traffic model [69], according to which we only have knowledge of the maximum amount of traffic entering and leaving the network at each edge node, but we do not have knowledge of the actual traffic matrix. Thus, given the current channel assignment, we address the problem to find a set of split ratios that optimize the *average* performance over all the *possible* sets of traffic demands. Consequently, it turns out that the set of split ratios does not need to be re-computed frequently, as they have been determined to guarantee high performance under different traffic loads.

The algorithm described in this chapter has been realized and evaluated in collaboration with a researcher of the COMICS group [22] [VI.1].

The rest of the chapter is structured as follows. In Section 6.1 we give an overview of the related work. The MPLS splitting policy is presented in Section 6.2. In Section 5.3 we formalize the problem to find a proper set of split ratios for the use with the MPLS splitting policy, while in Section 6.4 we present our approach to solve such a problem. In Section 6.5 we present the results of the simulation study we conducted to show that our approach achieves high throughput and is robust against variations in the traffic load and against single node failures. In Section 6.6 we draw the conclusions.

Related work

Most of the work related to routing in wireless mesh networks focused on link or path metrics proposed as improvements upon the hop count metric. Among the first link metrics to be introduced, the expected transmission count (ETX) [70] estimates the number of transmissions required to successfully send a packet to a neighbor. The authors in [71] introduce two metrics, the expected transmission time (ETT) and the weighted cumulative ETT (WCETT). ETT is the expected transmission time, which accounts for both the number of re-transmissions and the transmission rate used to send the packet. WCETT is an extension of ETT that considers the intra-flow interference in order to be applicable to multi-radio WMNs. MIC (metric of interference and channel switching) [72] and iAWARE (interference aware) [73] take the inter-flow interference into account in addition to the intra-flow interference. The authors in [74] introduce two more metrics: the modified expected transmission count (mETX) and the effective number of transmissions (ENT). Load-aware link metrics are instead proposed, e.g., in [75] and [76]. The above mentioned link metrics (and many others) are intended to be used with a single path destination-based routing protocol. Very often, the routing protocol used to test the proposed routing metric is one of those designed for ad hoc networks, like AODV (Ad hoc On demand Distance Vector) [43] or OLSR (Optimized Link State Routing) [41]. The routing protocol specified in the IEEE 802.11s draft standard [77], too, is basically a modified version of AODV that uses the Airtime link metric to associate each link with an estimate of the amount of time needed to successfully transmit a packet across that link. These routing protocols, being single path, have limited capabilities in terms of load balancing and require some time to discover alternative routes in case of link/node failures.

An adaptive load-aware routing scheme is proposed in [78]. The network is divided into multiple clusters and each cluster head estimates the traffic load in its cluster. If the estimated load gets higher, the cluster head increases the routing metrics of the routes passing through the cluster so that the traffic avoids overloaded clusters. This scheme requires a continuous adaptation of the link costs to the offered traffic load, which might lead to instabilities, and does not account for link/node failures. ExOR [79] is an opportunistic approach where a node broadcast a packet and the nodes that received it correctly agree on which of them has to further forward the packet, based on the distance to the destination. Such an approach exploits the broadcast nature of the wireless medium and avoids to retransmit a packet if at least one node has received it correctly. However, the protocol used by the receivers to agree on the closest node to the destination introduces some overhead. Also, ExOR is less effective in multi-radio WMNs because only the neighbors listening on the channel used by the sender can receive the packet. ROMER [80] builds a "forwarding mesh" around the minimum cost, stable, path and each packet is allowed to travel along one of the paths in the forwarding mesh based on the current conditions. ROMER and L2.5 share the principle of using multiple paths, but ROMER does not take into account the constraints on the available bandwidth resulting from the channel assignment. Other multi-path approaches, e.g. [81][82], aim to find link disjoint paths between the ingress and the egress. Though they can achieve load balancing, they do not allow to locally recover from a link/node failure.

Some papers have proposed to use MPLS in wireless mesh or ad hoc networks. For instance, [83] and [84] adopt MPLS to reduce the latency in forwarding packets by removing the need of performing route lookup operations and thus allowing to forward packets in a cut-through fashion. In [85], an architecture for resource reservation and QoS assurance is proposed which makes use of the MPLS signalling mechanism to establish a path and reserve resources on a per-flow basis. In [86], the use of MPLS is proposed to deal with mobility issues. In particular, MPLS is used in wireless mesh networks to create tunnels that are required by Mobile IP to function. So far, thus, the advantages that MPLS offers in terms of load balancing and fast restoration have not been exploited in the context of the

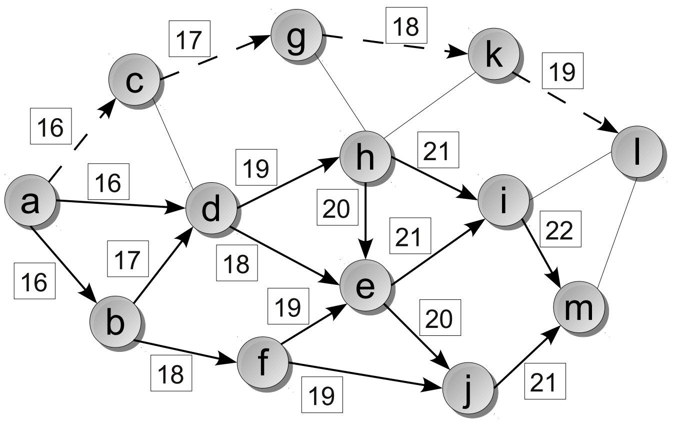


Figure 6.1: Example to illustrate the splitting mechanism

wireless mesh networks.

6.2 The MPLS splitting policy

According to RFC 3031 [2], MPLS nodes use three tables to forward packets: NHLFE (Next Hop Label Forwarding Entry), ILM (Incoming Label Map) and FTN (FEC-to-NHLFE Map). An entry of the NHLFE specifies the next hop, the operation to perform on the packet's label stack and (optionally) any additional information needed in order to properly dispose of the packet. RFC 3031 provides the following three operations: pop the label at the top of the stack, push a new label onto the stack, swap the label at the top of the stack and possibly push other labels. The ILM is used when forwarding packets that arrive as labeled packets. An entry of the ILM maps an incoming label to a *set* of NHLFE entries. The FTN, instead, is used when forwarding packets that arrive unlabeled. An entry of the FTN maps a FEC (Forwarding Equivalence Class) to a *set* of NHLFE entries. Normally, each FEC in a FTN entry is associated with a *single* NHLFE entry and each label in an ILM entry is associated with a *single* NHLFE entry. In such a way, there is a unique next hop for a given FEC or label at each node and thus all the packets of a FEC follow the same path (e.g., the dashed path from a to l in fig. 6.1).

As noted above, RFC 3031 explicitly mentions the possibility that a label or a FEC may be associated with a set of NHLFE entries, in order to perform, e.g., some sort of load balancing. We exploit such a possibility to allow the packets of a FEC to follow a predefined set of paths (as opposed to a single path) between the ingress and egress nodes. Such an approach is illustrated by fig. 6.1, where a continuous arrow departing from a node denotes a possible next hop (as specified in an NHLFE entry) for the packets that entered the network at a and are destined to m. It can be observed that nodes have multiple possible next hops from among they select the one which a given packet is forwarded to. As a consequence of such a choice, different packets of the same FEC may follow distinct paths (e.g., $a - d - h - i - m$ or $a - d - e - j - m$). All the possible paths taken by the packets of a given FEC are determined a priori and can be enforced by properly configuring the MPLS tables on the nodes. For instance, the behavior of node b is achieved by configuring an entry in its ILM that associates the incoming label 16 with two entries in the NHLFE: one that replaces label 16 with label 17 and sends the packet to node d and the other one that replaces label 16 with label 18 and sends the packet to node f. When an incoming packet with label 16 arrives, node b has to select either of the two NHLFE entries.

In case a FEC or a label is associated with multiple NHLFE entries, the procedures to choose an NHLFE entry among the given set are beyond the scope of RFC 3031. Here, we define a policy to select one of multiple NHLFE entries that fits our goal to balance the traffic among the outgoing links in proportion to their available bandwidth, while ensuring a fast reaction to node/link failures. We assume that each NHLFE entry also specifies, as an additional information, a *split ratio*, which is a value between 0 and 1. The split ratios associated with a set of NHLFE entries that correspond to the same FEC or to the same label must sum to 1. The goal of the splitting policy is to balance the traffic matching a given FEC or a given label among the neighbors specified by the corresponding NHLFE entries in proportion to the specified split ratios. For this purpose, the algorithm shown in fig. 6.2 is used to select an NHLFE entry (and hence a next hop) from among the set of NHLFE entries associated with a given FEC or with a given label. The procedure shown in fig. 6.2 is given the set of the split ratios $\{\rho_i\}$ associated with

```
MPLS SPLITTING($\{\rho_i\}$, $\{\overline{\rho}_i\}$, $B$, $p$)
1  for each NHLFE $i$ in decreasing order of $\rho_i - \overline{\rho}_i$
2      do if the next hop in NHLFE $i$ is reachable
3            then select NHLFE $i$
4                 $\overline{\rho}_i \leftarrow \frac{\overline{\rho}_i \cdot B + p}{B + p}$
5                 $\overline{\rho}_j \leftarrow \frac{\overline{\rho}_j \cdot B}{B + p} \quad \forall j \neq i$
6                 $B \leftarrow B + p$
7                 if $B > B_{max}$
8                   then $B \leftarrow B_{min}$
9                 return
```

Figure 6.2: Pseudo-code of the MPLS splitting policy

the set of NHLFE entries, the set of the actual utilizations $\{\overline{\rho}_i\}$ of each NHLFE entry (i.e., the ratio of the amount of traffic transmitted as specified by an NHLFE entry to the total traffic matching the FEC or the label), a counter B that records the amount of traffic matching the given FEC or label, and the size p of the packet for which an NHLFE entry must be selected. Before the traffic starts flowing, all the actual utilizations and the counter B are set to zero. Then, every time a packet matches a given FEC or label, the associated NHLFE entries are sorted and visited in decreasing order of the gap between the split ratio and the actual utilization. If the next hop neighbor included in the i-th NHLFE entry is marked as unreachable, then the NHLFE entry is skipped. To this end, we assume that a feedback is provided by the lower layers informing on the unavailability of a neighbor. Otherwise, the packet is sent as specified by the i-th NHLFE entry and the actual utilization of all the NHLFE entries and the total amount of traffic B are updated (lines 4–6). To avoid that B grows indefinitely, it is reset to a value B_{min} once it exceeds a given threshold B_{max}. B_{min} should be a value greater than zero to avoid that all the actual utilizations are reset after receiving the next packet (from line 5, $\overline{\rho}_j$ would be null if B were zero). Also, a node that is marked as unreachable can be included among the active neighbors again after a configurable amount of time.

Thus, the proposed MPLS splitting policy enables to balance the traffic matching a given FEC or label among the neighbors specified in the associated NHLFE entries in proportion to the corresponding split ratios. Also, by having multiple NHLFE entries (and hence next hop neighbors) already configured, our splitting policy enables a fast restoration against single node/link failures, as another NHLFE entry can be readily used to send the packet. We note here that it also makes sense to have NHLFE entries with an associated null split ratio. Such entries are not used to forward packets in normal conditions, but they are only used in case any other entry has been disabled due to the specified next hop being unreachable. Below is the proof that, in the absence of failures, NHLFE entries with a null split ratio are not used to send packets.

We denote by $\overline{\rho}_i^{(k)}$ the actual utilization of the i-th NHLFE entry after k packets have been sent. It can be shown by induction that $\sum_{i\in\mathcal{N}}\overline{\rho}_i^{(k)} = 1$ at any time (i.e., for $k \geqslant 1$), where $\mathcal{N}$ is the set of NHLFE entries associated with a given FEC or a given label. Indeed, by observing how the actual utilizations are updated (lines 4–6), it can be easily proven that $\sum_{i\in\mathcal{N}}\overline{\rho}_i^{(1)} = 1$ and $\sum_{i\in\mathcal{N}}\overline{\rho}_i^{(k)} = 1 \Rightarrow \sum_{i\in\mathcal{N}}\overline{\rho}_i^{(k+1)} = 1$. Consequently, since the split ratios are such that $\sum_{i\in\mathcal{N}}\rho_i = 1$, it turns out that $\sum_{i\in\mathcal{N}}(\rho_i - \overline{\rho}_i^{(k)}) = 0$ for every k. By induction, again, we can prove that $\rho_j = 0$ implies $\overline{\rho}_j^{(k)} = 0$ for every k (i.e., the j-th NHLFE entry is never used to send packets), in the absence of failures. Indeed, the first packet is sent using the NHLFE entry associated with the highest split ratio and hence $\overline{\rho}_j^{(1)} = 0$. If $\overline{\rho}_j^{(k)} = 0$ then $\rho_j - \overline{\rho}_j^{(k)} = 0$. Hence, in order for the j-th NHLFE entry to be selected to send the $(k+1)$-th packet, $\rho_i - \overline{\rho}_i^{(k)} < 0$ should hold for every $i \neq j$. However, this is not possible since $\sum_{i\in\mathcal{N}}(\rho_i - \overline{\rho}_i^{(k)}) = 0$ for every k. Thus, the j-th NHLFE entry is not selected and $\overline{\rho}_j^{(k+1)} = 0$. We have therefore shown that an NHLFE entry with a null split ratio is not used to send packets in the absence of failures. However, in case the neighbors specified in the NHLFE entries with non-null split ratios are unreachable, our splitting policy selects the NHLFE entry with a null split ratio to send the packet. Hence, NHLFE entries with a null split ratio (that may be present in a solution returned by our approach proposed in Section 6.4.2) can be usefully configured since they serve as backup routes in case of failures.

Finally, we note that a behavior similar to that of the MPLS splitting policy can be obtained by configuring a suitable number of single-path LSPs (Label Switched Paths) between the ingress and egress nodes and having the ingress node split the incoming traffic in proper proportions among such LSPs. The path followed by each packet is thus determined by the ingress node and cannot be modified by intermediate nodes. For instance, in the example shown in fig. 6.1, we may configure 13 single-path LSPs between a and m to achieve the same set of possible paths as allowed by the MPLS splitting policy. However, our splitting policy enables a local fast restoration in case of failures, while, in case multiple single-path LSPs are configured, the notice of a failure must be propagated back to the ingress node, which then excludes the LSP involved. Also, the MPLS splitting policy allows for a reduction in the configuration burden. Indeed, each single-path LSP requires an NHLFE entry for each link along the path and hence the total number of NHLFE entries required equals the sum of the path lengths of all the LSPs. An equal number of FTN or ILM entries is needed, too. Instead, the MPLS splitting policy requires as many NHLFE entries (and ILM or FTN entries) as the number of links involved. In the example of fig. 6.1, the MPLS splitting policy requires 14 NHLFE entries, while using a set of single-path LSPs requires 63 NHLFE entries, i.e., a number of entries 4.5 times greater.

6.3 The MPLS splitting-based routing problem

The model of WMN, the interference and the notion of the total utilization of a collision domain are the ones formalized in sec. 4.3.

Without loss of generality, here we consider a set $V_e = \{n_1, ...n_N\} \subseteq V$ of N edge nodes acting as both ingress and egress nodes. According to the hose traffic model, we only have knowledge of the maximum amount of traffic entering or leaving the network at each edge node. We denote by $\mathcal{I}^{max} = \{I_s^{max}\}_{s \in V_e}$ and $\mathcal{O}^{max} = \{O_d^{max}\}_{d \in V_e}$, respectively, the sets of the maximum amount of incoming and outgoing traffic at each edge node. However, we know neither the actual amount of traffic entering at each edge node nor what portion of traffic entering at a given edge node is routed towards each of the other N-1 edge nodes. A set of incoming flows $\mathcal{I} = \{I_s\}_{s\in V_e}$ is said to be *feasible* if $I_s \leqslant I_s^{max} \quad \forall s \in V_e$.

Our goal is to route the (unknown) traffic matrix, using MPLS and the splitting policy, in such a way to minimize the cost function defined in the next section. A routing *solution* consists of a set of split ratios $\{\rho^{s,d}_{u\to v}\}^{s\in V_e, d\in V_e-\{s\}}_{u\to v\in\mathcal{E}^{s,d}}$, where $\rho^{s,d}_{u\to v}$ represents the ratio of the flow between the ingress-egress pair (s,d) entering node u that is forwarded to node v and $\mathcal{E}^{s,d}$ represents the set of links along which the flow between the ingress s and the egress d is routed. Clearly, the equation $\sum\limits_{v\,|\,u\to v\in\mathcal{E}^{s,d}} \rho^{s,d}_{u\to v} = 1$ must hold for each u and for each ingress-egress pair (s,d). The set of split ratios determine, for each ingress-egress pair (s,d), a directed subgraph of G, $\mathcal{S}^{s,d} = (\mathcal{V}^{s,d}, \mathcal{E}^{s,d})$, where $\mathcal{V}^{s,d}$ is the set of nodes belonging to the links in $\mathcal{E}^{s,d}$. Given how the splitting policy works, it turns out that the packets flowing from ingress node s to egress node d can follow any of the paths between s and d in the subgraph $\mathcal{S}^{s,d}$. A routing solution is said to be *admissible* if, for every ingress-egress pair (s,d), the set of links $\mathcal{E}^{s,d}$, or, equivalently, the directed subgraph $\mathcal{S}^{s,d}$, meets the following constraints:

t') in order to avoid that packets take excessively long paths, the length of every path in $\mathcal{S}^{s,d}$ must be at most α times the length of the shortest path between s and d in G

t'') every path in $\mathcal{S}^{s,d}$ must be cycle-free

t''') the set of paths in $\mathcal{S}^{s,d}$ must guarantee protection against single node/link failures, i.e., if a single node/link fails, the upstream node must have an alternative path to the egress node d

The set of split ratios determine how the (unknown) traffic matrix is routed across the network. We denote by $f^{s,d}_{u\to v}$ the variable representing the amount of flow between the ingress-egress pair (s,d) that is routed on link $u\to v$. The total amount of flow routed on a link $u\to v$ is thus given by $f_{u\to v} = \sum\limits_{s\in V_e}\sum\limits_{d\in V_e-\{s\}} f^{s,d}_{u\to v}$.

We also denote by $\varphi^{s,d}_{u\to v} = \dfrac{f^{s,d}_{u\to v}}{I_s}$ the variable representing the amount of flow on link $u \to v$ contributed by node s and destined to node d, *normalized* to the actual (unknown) amount of traffic I_s entering source node s. We observe that $\varphi^{s,d}_{u\to v}$ is independent of the actual amount of traffic entering source node s and only depends on how traffic flows are routed. As shown in Section 6.4.4, the set $\Phi = \{\varphi^{s,d}_{u\to v}\}^{s\in V_e, d\in V_e-\{s\}}_{u\to v\in \mathcal{E}^{s,d}}$ suffices to determine the set of split ratios, and hence it can be considered as representative of a particular routing solution.

We denote by $\Gamma(\Phi, \mathcal{I})$ the cost of a particular configuration where a routing solution Φ is used to route a feasible set of incoming flows $\mathcal{I} = \{I_s\}_{s\in V_e}$. Our goal is to compute $\Gamma(\Phi)$, the average cost of a routing solution Φ, i.e., the average of $\Gamma(\Phi, \mathcal{I})$ over all the feasible sets $\mathcal{I}$:

$$\Gamma(\Phi) = \frac{1}{\prod\limits_{s\in V_e} I^{max}_s} \int_0^{I^{max}_{n_1}} \cdots \int_0^{I^{max}_{n_N}} \Gamma(\Phi, \mathcal{I}) dI_{n_1} \cdots dI_{n_N} \tag{6.1}$$

Given the maximum amount of traffic entering or leaving the network at each edge node, the MPLS splitting-based routing problem is to find a *feasible* admissible routing solution Φ that minimizes $\Gamma(\Phi)$. A routing solution is said to be feasible, given $\mathcal{I}^{max}$ and $\mathcal{O}^{max}$, if it obeys the following constraints:

f′) the amount of flow routed on each link must not exceed the link capacity, for *every* feasible set of incoming flows

f″) the amount of flow routed towards each egress node must not exceed the maximum amount of outgoing traffic of that egress node, for *every* feasible set of incoming flows

The feasible admissible routing solution that minimizes $\Gamma(\Phi)$ has the minimum cost on the average and hence it can be considered as the most robust routing solution against variations of the traffic matrix. In the next section, we first define $\Gamma(\Phi, \mathcal{I})$ and then present our approach to solve the MPLS splitting-based routing problem. Also, we show how a given routing solution Φ directly maps to the set of split ratios needed by our MPLS splitting policy.

6.4 Solving the MPLS splitting-based routing problem

In this section, we show how we solve the MPLS splitting-based routing problem defined in the previous section:

- we first define the cost $\Gamma(\Phi, \mathcal{I})$ of routing a given set $\mathcal{I}$ of actual incoming flows according to a particular routing solution Φ. Then, we compute the average cost $\Gamma(\Phi)$ of a routing solution Φ over all the feasible sets of incoming flows (Section 6.4.1)

- then, we address the problem to find a feasible admissible routing solution minimizing $\Gamma(\Phi)$. Requiring that the returned routing solution be admissible makes the problem to find a feasible routing solution minimizing $\Gamma(\Phi)$ hard to solve. Hence, our approach is to decouple the problem to find a set of directed subgraphs that make a routing solution admissible from the problem to find a feasible routing solution minimizing $\Gamma(\Phi)$ subject to the constraint that the flow between each pair of ingress and egress nodes can only be routed along the links of predefined subgraphs. We present a convex optimization problem to find an optimal solution to the latter problem (Section 6.4.2) and propose a heuristic to solve the former problem (Section 6.4.3)

- finally, we show how the set of split ratios can be derived from the values of the $\varphi_l^{s,d}$ variables (Section 6.4.4)

6.4.1 Computing the average cost of a routing solution

In this section, we define the cost $\Gamma(\Phi, \mathcal{I})$ of a particular configuration where a given routing solution $\Phi = \{\varphi_{u \to v}^{s,d}\}_{u \to v \in \mathcal{E}^{s,d}}^{s \in V_e, d \in V_e - \{s\}}$ is used to route a given feasible set of incoming flows $\mathcal{I} = \{I_s\}_{s \in V_e}$. In the previous section, we mentioned that $U_{tot}(e) \leqslant 1 \quad \forall e \in E$ is a sufficient condition to ensure that the flows allocated on the network links are schedulable. It follows that we may consider the average of the total utilization over all the collision domains as a measure of how efficient a particular routing solution is in routing the given set of incoming flows, given the current channel assignment. The higher the average, the less efficient the routing solution. In order to further penalize the solutions leading to high values for the total utilization of some collision domains, we consider a weighted average of the total utilizations. In particular, we consider the weighting function:

$$w(x) = \frac{e^x - 1}{e - 1}$$

and define the cost $\Gamma(\Phi, \mathcal{I})$ of a particular configuration as the average of $w(U_{tot}(e))$ over all the links $e \in E$:

$$\Gamma(\Phi, \mathcal{I}) = \frac{1}{|E|} \sum_{l_0 \in E} w(U_{tot}(l_0)) = \frac{1}{|E|} \sum_{l_0 \in E} \frac{e^{U_{tot}(l_0)} - 1}{e - 1} \tag{6.2}$$

The weighting function is such that $w(x) \leqslant x$ if $x \leqslant 1$ and $w(x) > x$ if $x > 1$, i.e., it decreases the weight of the total utilizations below 1 and increases the weight of the total utilizations above 1. The goal is thus to penalize the configurations with total utilizations larger than 1.

For conciseness, we define $\varphi_l^s = \sum\limits_{d \in V_e - \{s\}} \varphi_l^{s,d}$, i.e., φ_l^s is the amount of flow on link l originated at node s (independently of the destination node) and normalized to the actual (unknown) amount of traffic I_s entering node s. It follows that the actual amount of flow routed on link l can be expressed as $\sum\limits_{s \in V_e} \varphi_l^s I_s$. Hence:

$$\begin{aligned}
\Gamma(\Phi, \mathcal{I}) &= \frac{1}{|E|(e-1)} \sum_{l_0 \in E} \left[e^{\sum\limits_{l \in \mathcal{D}(l_0)} \sum\limits_{s \in V_e} \frac{\varphi_l^s I_s}{c(l)}} - 1 \right] \\
&= \frac{1}{|E|(e-1)} \sum_{l_0 \in E} \left[e^{\sum\limits_{s \in V_e} \sum\limits_{l \in \mathcal{D}(l_0)} \frac{\varphi_l^s I_s}{c(l)}} - 1 \right]
\end{aligned}$$

We compute $\Gamma(\Phi)$ by integrating $\Gamma(\Phi, \mathcal{I})$ over the region of all the feasible sets of incoming flows (eq. 6.1):

$$\begin{aligned}
&\Gamma(\Phi) = \frac{1}{|E|(e-1)\prod\limits_{s\in V_e} I_s^{max}}\cdot \\
&\sum_{l_0\in E}\int_0^{I_{n_1}^{max}}\cdots\int_0^{I_{n_N}^{max}}\left[e^{\sum\limits_{s=1}^{N}\sum\limits_{l\in\mathcal{D}(l_0)}\frac{\varphi_l^s I_s}{c(l)}}-1\right]dI_1\cdots dI_N \\
&= \frac{1}{|E|(e-1)\prod\limits_{s\in V_e} I_s^{max}}\cdot \\
&\sum_{l_0\in E}\left[\int_0^{I_{n_1}^{max}}\cdots\int_0^{I_{n_N}^{max}}\prod_{s=1}^{N} e^{\sum\limits_{l\in\mathcal{D}(l_0)}\frac{\varphi_l^s I_s}{c(l)}}dI_{n_1}\cdots dI_{n_N}-\prod_{s=1}^{N}I_s^{max}\right]
\end{aligned} \tag{6.3}$$

The integrating function is the product of N functions each depending on a different integration variable. Hence, the multiple integral can be decomposed as the product of N integrals:

$$\begin{aligned}
&\int_0^{I_{n_1}^{max}}\cdots\int_0^{I_{n_N}^{max}}\prod_{s\in V_e} e^{\sum\limits_{l\in\mathcal{D}(l_0)}\frac{\varphi_l^s I_s}{c(l)}}dI_{n_1}\cdots dI_{n_N} \\
&= \prod_{s\in V_e}\left[\frac{1}{\sum\limits_{l\in\mathcal{D}(l_0)}\frac{\varphi_l^s}{c(l)}}\cdot e^{\sum\limits_{l\in\mathcal{D}(l_0)}\frac{\varphi_l^s I_s}{c(l)}}\right]_0^{I_s^{max}} \\
&= \prod_{s\in V_e}\frac{e^{\sum\limits_{l\in\mathcal{D}(l_0)}\frac{\varphi_l^s I_s^{max}}{c(l)}}-1}{\sum\limits_{l\in\mathcal{D}(l_0)}\frac{\varphi_l^s}{c(l)}}
\end{aligned}$$

Hence, the average cost $\Gamma(\Phi)$ of a routing solution Φ over all the feasible sets of incoming flows is:

$$\Gamma(\Phi)=\frac{1}{e-1}\left[\frac{1}{|E|\prod\limits_{s\in V_e}I_s^{max}}\cdot\sum_{l_0\in E}\prod_{s\in V_e}\frac{e^{\sum\limits_{l\in\mathcal{D}(l_0)}\frac{\varphi_l^s I_s^{max}}{c(l)}}-1}{\sum\limits_{l\in\mathcal{D}(l_0)}\frac{\varphi_l^s}{c(l)}}-1\right] \tag{6.4}$$

variables

$$\varphi_l^{s,d} \in [0,1] \quad \forall l \in E,\, \forall s \in V_e,\, \forall d \in V_e - \{s\}$$
$$F^{s,d} \in [0,1] \quad \forall s \in V_e,\, \forall d \in V_e - \{s\}$$

minimize $\Gamma\left(\left\{\varphi_l^{s,d}\right\}_{l\in\mathcal{E}^{s,d}}^{s\in V_e, d\in V_e-\{s\}}\right)$

subject to

1. $\sum_{u\to v\in\mathcal{E}^{s,d}} \varphi_{u\to v}^{s,d} - \sum_{v\to u\in\mathcal{E}^{s,d}} \varphi_{v\to u}^{s,d} = \begin{cases} 0 & \text{if } u \neq s \wedge u \neq d \\ F^{s,d} & \text{if } u = s \\ -F^{s,d} & \text{if } u = d \end{cases}$

   $\forall u \in V,\ \forall s \in V_e,\ \forall d \in V_e - \{s\}$

2. $\sum_{d\in V_e-\{s\}} F^{s,d} = 1$

   $\forall s \in V_e$

3. $\sum_{s\in V_e-\{d\}} F^{s,d} I_s^{max} \leqslant O_d^{max}$

   $\forall d \in V_e$

4. $\varphi_{u\to v}^{s,d} = 0$

   $\forall s \in V_e,\ \forall d \in V_e - \{s\},\ v = s \vee u = d$

5. $\sum_{s\in V_e} \sum_{d\in V_e-\{s\}} \varphi_l^{s,d} I_s^{max} \leqslant c(l)$

   $\forall l \in E$

6. $\varphi_l^{s,d} = 0$

   $\forall s \in V_e,\ \forall d \in V_e - \{s\},\ l \notin \mathcal{E}^{s,d}$

Figure 6.3: Formulation of the METER problem

6.4.2 Finding an optimal feasible routing solution

The average cost $\Gamma(\Phi)$ of a particular routing solution Φ over all the feasible sets of incoming flows is expressed by equation (6.4). Here, we formulate a convex optimization problem, denoted as METER (Minimum avErage cosT fEasible Routing), to find a routing solution that is feasible given the maximum amount of traffic entering or leaving the network at each edge node and minimizes $\Gamma(\Phi)$, subject to the constraint that the flow between each pair of ingress and egress nodes can only be routed along a predefined set of links. Solving such a problem provides the normalized amount of flow $\varphi_l^{s,d}$ routed on each link l and belonging to each ingress-egress pair (s,d). From such information, as illustrated in section 6.4.4, we can derive the set of split ratios that each node needs to enforce our MPLS splitting policy.

The formulation of the METER problem is shown in fig. 6.3. Besides the set of normalized variables $\{\varphi_l^{s,d}\}_{l\in\mathcal{E}^{s,d}}^{s\in V_e,\, d\in V_e-\{s\}}$, we also consider a set of auxiliary variables $\{F^{s,d}\}^{s\in V_e, d\in V_e-\{s\}}$, each representing the amount of flow routed between an ingress node s and an egress node d, normalized to the actual incoming traffic at node s. The objective of METER is to minimize $\Gamma(\Phi)$. Constraints 1) represent the usual (normalized) flow conservation constraint that must be enforced at each node for every pair of ingress-egress nodes. Constraints 2) ensure that all the actual amount of incoming flow at each edge node is split among the other edge nodes. Constraints 3) ensure that the amount of flow routed towards each egress node does not exceed the maximum amount of outgoing traffic of that egress node, for *every* feasible set of incoming flows (constraint f'' of Section 5.3). Indeed, if the incoming set of flows is feasible, then $\sum\limits_{s\in V_e-\{d\}} F^{s,d} I_s \leqslant \sum\limits_{s\in V_e-\{d\}} F^{s,d} I_s^{max}$, where the left hand side is the actual amount of flow routed towards egress node d. Hence, if constraint 3) holds, constraint f'') holds as well. Constraints 4) prevent the incoming (outgoing) flow at an edge node to be re-routed back to the ingress node (from the egress node). Constraints 5) ensure that the amount of flow routed on each link does not exceed the link capacity, for *every* feasible set of incoming flows (constraint f' of Section 5.3). Finally, constraints 6) ensure that the flow between edge nodes s and d is only allocated on links that belong to

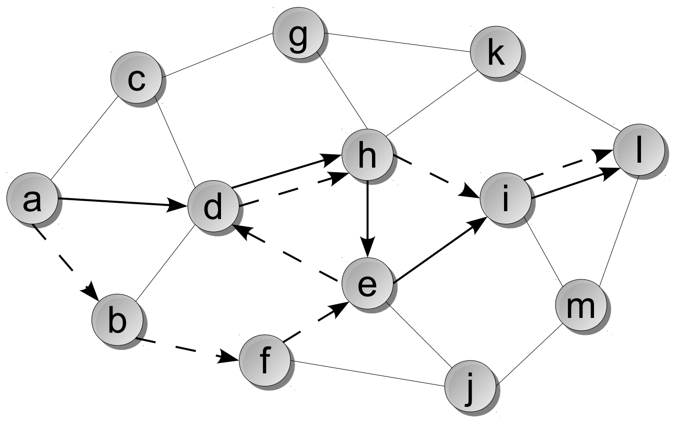


Figure 6.4: Example to illustrate loops in the directed subgraph

the predefined set $\mathcal{E}^{s,d}$. In such a way, if the predefined set of links are properly computed, the routing solution returned by the optimization problem is guaranteed to be admissible. In Section 6.4.3 we present an algorithm that finds, for a given ingress-egress pair (s, d), a directed subgraph that meets constraints t'), t'') and t''') of Section 5.3.

METER is a convex optimization problem, because the objective function is convex (see the Appendix) and the constraint functions are linear. Convex optimization problems have the property that a locally optimal point is also globally optimal. Hence, we use an interior point method [87] to find an optimal solution to the problem.

6.4.3 Finding a directed subgraph for an ingress-egress pair

We now address the problem to find, for a given ingress-egress pair (s, d), a set of links $\mathcal{E}^{s,d}$ or, equivalently, a directed subgraph $\mathcal{S}^{s,d}$ that guarantees that a routing solution is admissible. The approach we follow is to find a set of paths between s and d (in G) and insert all of their links into $\mathcal{E}^{s,d}$. Given the constraint on the maximum allowed length of any path between s and d in $\mathcal{S}^{s,d}$, a possible approach would be to use a k-shortest loopless path algorithm [88] to find all the loopless paths between s and d in G having a length less than the maximum one. A k-shortest loopless path algorithm indeed returns all the shortest paths in increasing

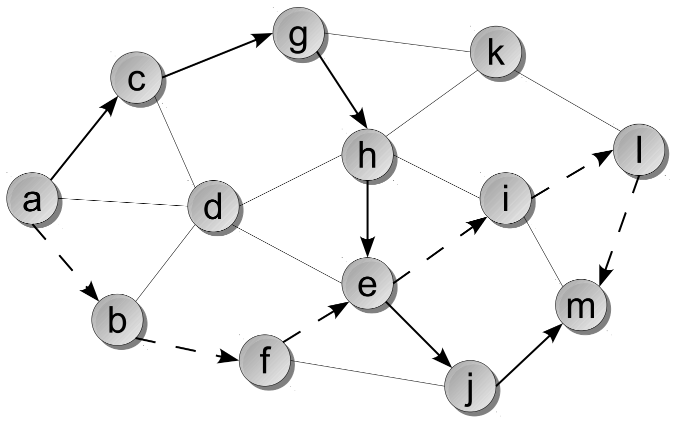


Figure 6.5: Example to illustrate paths exceeding the maximum length in the directed subgraph

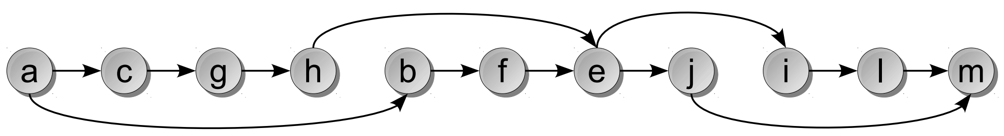


Figure 6.6: Topologically sorted nodes of the DAG in fig. 6.5

order of length. However, the following two issues should be taken into account:

- *only adding loopless paths to $\mathcal{E}^{s,d}$ does not ensure that all the possible paths in the subgraph $\mathcal{S}^{s,d}$ are loopless.* An example is illustrated in fig. 6.4, where the links of two loopless paths ($a-d-h-e-i-l$ and $a-b-f-e-d-h-i-l$) are added to $\mathcal{E}^{s,d}$. It can be noticed that the resulting subgraph includes paths (e.g., $a-b-f-e-d-h-e-i-l$) containing a cycle ($e-d-h-e$)

- *only adding paths with length less than the maximum allowed one does not ensure that all the possible paths in the subgraph have a length less than the maximum allowed one.* An example is illustrated in fig. 6.5, where we assume that the maximum length is 6 hops. If we add the links of two paths satisfying the constraint on the maximum path length ($a-c-g-h-e-j-m$ and $a-b-f-e-i-l-m$), the resulting subgraph includes a path ($a-c-g-h-e-i-l-m$) having a length (7 hops) exceeding the

```
DFS($D = (V_D, E_D), s$)
1   for each $u \in V_D$
2       do $color[u] \leftarrow$ WHITE
3          $pred[u] \leftarrow$ NIL
4          $maxdist[u] \leftarrow 0$
5   $sortedList \leftarrow \emptyset$
6   $acyclic \leftarrow$ TRUE
7   DFS_VISIT($D, s$)

DFS_VISIT($D = (V_D, E_D), u$)
 1  $color[u] \leftarrow$ GRAY
 2  for each $v \in V_D \mid u \rightarrow v \in E_D$
 3      do if $color[v] =$ WHITE
 4            then $pred[v] = u$
 5                 DFS_VISIT($D, v$)
 6            else if $color[v] =$ GRAY
 7                    then $acyclic \leftarrow$ FALSE
 8  $color[u] =$ BLACK
 9  if $sortedList \neq \emptyset$
10     then $maxdist[u] \leftarrow \max_{v \mid u \rightarrow v \in E_D} maxdist[v] + 1$
11  $sortedList \leftarrow u, sortedList$
```

Figure 6.7: Pseudo-code of the Depth-First-Search

maximum allowed length

Therefore, we present an algorithm that finds, for a given ingress-egress node pair, a directed subgraph of the network topology that meets constraint *t*′), *t*″) and *t*‴) of Section 5.3. Basically, the subgraph is initialized to contain the shortest path between the ingress and egress nodes and then it is augmented with other paths to fulfil constraint *t*‴). Every time we attempt to add a path to the subgraph, we check whether the augmented subgraph contains a cycle or a path having a length exceeding the maximum one. Fortunately, the subgraph we seek is a *directed acyclic graph* (DAG) and hence performing the above checks is as simple as running a Depth-First-Search (DFS) [89]. Also, a DFS in a DAG allows to sort the nodes in a topological ordering, which is such that if a link $u \rightarrow v$ exists in the DAG, then u precedes v in the ordering. Thus, the ingress (egress) node is the first (last) node in this ordering. As an example, fig. 6.6 shows a topological sort ordering of the DAG resulting from the two paths highlighted in fig. 6.5.

Figure 6.7 shows the pseudo-code of the Depth-First-Search visit used by our algorithm. The DFS procedure initializes the attributes (color, predecessor and maximum distance from the egress node) of all the nodes of the subgraph D, clears the list that will contain the nodes sorted in a topological ordering, sets the boolean variable $acyclic$ to true and calls DFS_VISIT on node s. The DFS_VISIT procedure performs the classic DFS visit starting from node u. In addition, if a *back* edge is detected (lines 6–7), the $acyclic$ variable is set to false. Indeed, a directed graph is acyclic if and only if a depth-first-search yields no back edges. To obtain a list of the nodes of the DAG sorted in a topological ordering, we can push nodes on the front of such a list as soon as they are marked as black (line 11). Also (lines 9–10), when inserting a node u in such a list (but the first node being inserted, which is the egress node), we compute the maximum distance in the DAG between u and the last node of the sorted list (the egress node) by increasing by 1 the maximum distance of each neighbor of u (at this point, all the neighbors v of u such that $u \to v$ exists in the DAG have been already inserted into the sorted list, by definition of topological ordering).

Therefore, a DFS on a subgraph D checks whether D is a DAG and, in that case, returns a sorted list of nodes in a topological ordering and the length of the longest path between each node in D and the last node in the topological ordering (in our case, the egress node).

We now present our algorithm, denoted as RDAS (Resilient Directed Acyclic Subgraph), to find, for a given pair (s,d) of ingress and egress nodes, a directed subgraph of a graph G that meets constraint t'), t'') and t''') of Section 5.3. Basically, RDAS initializes the subgraph D to the shortest path in G between the ingress and egress nodes and then explores the nodes in D in a reverse topological order, starting from the penultimate node. An attempt is made to ensure that the explored node u has two distinct next hops in D, so that to satisfy constraint t'''). To this end, a path between u and the egress node d is sought that does not include the current next hop of u and satisfies t') and t''). If such a path is found,

```
RDAS($G = (V, E), s, d, \alpha$)
 1  $D \leftarrow \emptyset$
 2  $SP \leftarrow$ SHORTEST_PATH($G, s, d$)
 3  PATH_ADD($D, SP$)
 4  DFS($D, s$)
 5  for each $u \in V$
 6       do $done[u] \leftarrow$ FALSE
 7  $u \leftarrow previous[back[sortedlist[D]]]$
 8  while $u <> $ NIL
 9       do if $done[u] =$ TRUE
10             then $u \leftarrow previous[u]$
11                  continue
12          if $|Adj(D, u)| = 1$
13             then $v \leftarrow front[Adj(D, u)]$
14                  $G_{Pruned} \leftarrow G$
15                  if $v = d$
16                     then REMOVE_EDGE($G_{Pruned}, u \rightarrow d$)
17                     else REMOVE_VERTEX($G_{Pruned}, v$)
18                  $found \leftarrow$ FALSE
19                  $L^{su} \leftarrow$ MAX_DIST_FROM_SOURCE($D, u$)
20                  $L^{ud} \leftarrow 0$
21                  while !$found$ AND
                          $L^{su} + L^{ud} \leqslant \alpha \cdot length[SP]$ AND
                          KSP_HAS_NEXT($G_{Pruned}, u, d$)
22                        do $P \leftarrow$ KSP_NEXT($G_{Pruned}, u, d$)
23                           $D_{Augm} \leftarrow D$
24                           PATH_ADD($D_{Augm}, P$)
25                           DFS($D_{Augm}, s$)
26                           if IS_ACYCLIC($D_{Augm}$) AND
                                MAX_DIST_TO_DEST($D_{Augm}, s$)
                                $\leqslant \alpha \cdot length[SP]$
27                              then $found \leftarrow$ TRUE
28                           $L^{ud} \leftarrow length[P]$
29                  if $found$
30                     then $done[u] \leftarrow$ TRUE
31                          $D \leftarrow D_{Augm}$
32                          $u \leftarrow previous[back[sortedlist[D]]]$
33                          continue
34          $done[u] \leftarrow$ TRUE
35          $u \leftarrow previous[u]$
```

Figure 6.8: Pseudo-code of the RDAS algorithm

it is added to D and the exploration restarts from the penultimate node in the new topological ordering of the nodes in D. An explored node is marked as *done*, so that it is explored just once. The algorithm ends when the ingress node is marked as done.

We now describe the exploration of a node in more details. If the explored node has been already marked as done, we continue by exploring its predecessor in the topological ordering of the nodes in D (lines 9–11). If the explored node has already more than one next hop in D, it is marked as done and its predecessor in the topological ordering is then explored (lines 34–35). If the explored node u has a single neighbor (v) in D, we attempt to find an alternative path to the egress node. For this purpose, we consider a copy (G_{pruned}) of the input graph G and prune the link $u \rightarrow v$, in case v is the egress node, or the vertex v otherwise. Then, we look for a path between u and the egress node d in the pruned graph. A k-shortest loopless path algorithm (we use the one proposed in [90], which is an efficient implementation of the Yen's algorithm [88]) provides, one-by-one and in increasing order of length, the shortest paths between u and d in the pruned graph. The path P returned by the k-shortest path algorithm is tentatively added to a copy (D_{Augm}) of the subgraph D. A DFS of D_{Augm} is run, which determines whether D_{Augm} is acyclic, finds the length of the longest path between s and d and topologically sorts the nodes. If D_{Augm} is acyclic and the length of the longest path is less than the maximum allowed path length, the path P is actually added to D, node u is marked as done and the exploration of the nodes restarts from the penultimate node in the new topological ordering (lines 29–33). Otherwise, a new path returned by the k-shortest path algorithm is considered.

In order to avoid that a number of shortest paths between u and d in the pruned graph are uselessly considered, we compute the length L^{su} of the longest path between the ingress node s and u in D (line 19). Given that we already have a topological ordering of the nodes in D, computing such a value only requires to relax all the edges of D (whose weights must be set to -1) [89]. Thus, as soon as the k-shortest path algorithm returns a path with a length L^{ud} such that $L^{su} + L^{ud}$ exceeds the maximum allowed path length, we can stop processing the shortest paths between u and d. Indeed, since shortest paths are returned in increasing order of length, we are sure that none of the following shortest paths can be added to the subgraph without violating the constraint on the maximum path length. In such a case, node u is marked as done (despite it only has one neighbor) and the predecessor of u in the topological ordering is explored. The algorithm ends when the ingress node s is marked as done and returns the directed subgraph D.

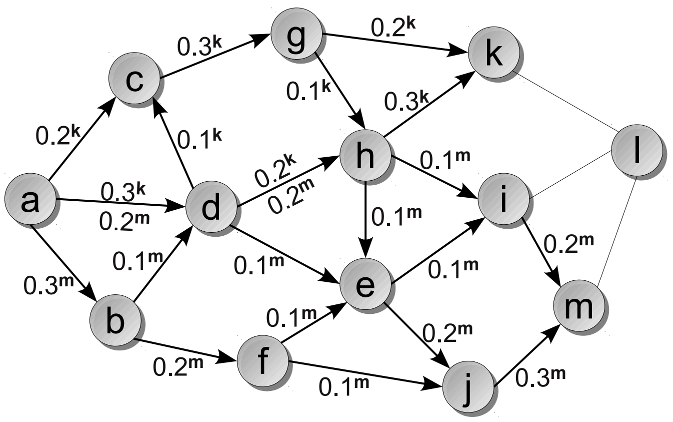
The complexity of RDAS is dominated by the complexity of the inner while loop (lines 21–28). In the worst case (since the nodes in D are a subset of those in G), a DFS on D requires $O(|V| + |E|)$, while obtaining the next path from the k-shortest path algorithm requires $O(|V|(|E| + |V| \log |V|))$. The outer while loop is repeated at most $|V|$ times (once for each node in D), hence the complexity of RDAS is $O(|V|^2(|E| + |V| \log |V|))$.

6.4.4 Computing the optimal set of split ratios

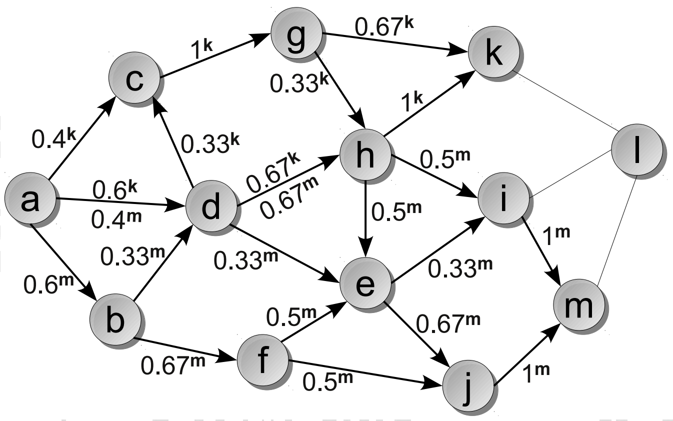
Solving the METER problem (fig. 6.3) provides the values for the set of variables $\{\varphi_l^{s,d}\}_{l\in\mathcal{E}^{s,d}}^{s\in V_e, d\in V_e-\{s\}}$ representing the amount of flow routed on each link and associated with each pair of ingress-egress nodes, normalized to the actual incoming flow at the ingress node. Figure 6.9a shows a network with some sample values for the flows routed between a and k and between a and m (the superscript next to a value indicates the destination of the flow) normalized to the incoming flow at a. It is then easy to derive the set of split ratios to be used by our MPLS splitting policy. Since the splitting policy balances the traffic among the outgoing links in proportion to the split ratios, in order to achieve the normalized flows given by the set of variables $\{\varphi_l^{s,d}\}_{l\in\mathcal{E}^{s,d}}^{s\in V_e, d\in V_e-\{s\}}$ we need to set the split ratios as follows:

$$\rho_{u\to v}^{s,d} = \frac{\varphi_{u\to v}^{s,d}}{\sum\limits_{u\to w\in\mathcal{E}^{s,d}} \varphi_{u\to w}^{s,d}}$$

Figure 6.9b shows the set of split ratios corresponding to the normalized flow values of fig. 6.9a.



(a) Sample values for the $\varphi_l^{s,d}$ variables



(b) Corresponding set of split ratios

Figure 6.9: Deriving the set of split ratios from the set of $\varphi_l^{s,d}$ variables

6.5 Performance evaluation

We consider several topologies for the evaluation of the proposed approach for the MPLS splitting-based routing problem. Each topology is generated by randomly placing 25 nodes in a 300 $\times$ 300 square meter area. Each node is endowed with two or three network radios. Given the planar coordinates of the nodes, a software we implemented on our own is used to build the network topology based on the interference model described in Section 5.3. Then, FCRA, a channel and rate assignment algorithm proposed in [37], is used to assign channels to radios and rates to links. Since FCRA is a traffic-aware channel assignment algorithm, different channel assignments can be obtained by feeding FCRA with different sets of link flows. We consider 3 edge nodes and therefore we have 6 ingress-egress pairs. METER, the convex optimization problem described in Section 6.4.2, is solved by using the open source software *Ipopt* (Interior Point OPTimizer).

The following subsections (except the first one) aim to evaluate the performance of our approach in terms of network throughput. To this end, experiments were carried out with the network simulator *ns-3*. We contributed to the implementation of the MPLS module in ns-3 and added the MPLS splitting policy[1]. In such experiments, we compare our approach (simply labelled as MPLS) based on the MPLS splitting policy with the split ratios determined as shown in Section 6.4 to our previous Layer-2.5 forwarding paradigm (L2.5) and to the routing protocol specified in the IEEE 802.11s draft standard [77] (802.11s). Unless explicitly stated, RDAS is run with $\alpha = 3$ in order to consider paths that are much longer than the shortest path, which likely consists of links between distant nodes utilizing low bit rates. In the ns-3 experiments, TCP traffic is generated according to the on-off model, with $T_{on} \sim U(0.5s, 1.5s)$ and $T_{off} \sim U(0.05s, 0.15s)$.

Comparing METER-RDAS to a lower bound

Our approach to solve the MPLS splitting-based routing problem is to solve the METER-RDAS problem, where RDAS is used to compute the set of directed

[1]As of this writing, the MPLS module has not been merged yet into the mainline ns-3 code. The MPLS code is available at `http://code.google.com/p/ns-3-shop`

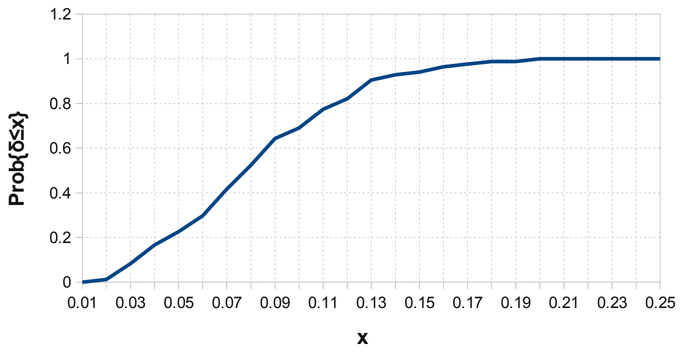
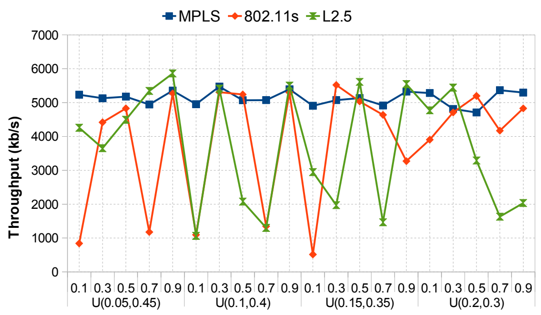


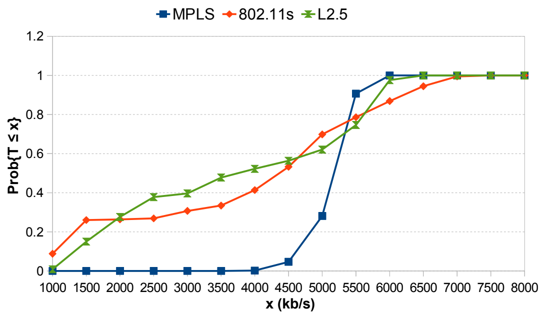
Figure 6.10: Empirical CDF of δ

subgraphs that are required to formulate the METER problem. Thus, our approach might not return the optimal feasible admissible routing solution of the MPLS splitting-based routing problem, because the returned routing solution is constrained to allocate flow on the links of the directed subgraphs computed by RDAS. Though the average cost of the optimal routing solution Φ^{opt} is difficult to find, it is straightforward to compute a lower bound to such value. To this end, we denote by $\mathcal{E}^{s,d}_{KSP}$ the set of links of all the paths between s and d in G whose length is at most α times the length of the shortest path. Such a set can be easily computed by using a k-shortest loopless path algorithm to find all the paths between s a d with length less than the maximum one. $\mathcal{E}^{s,d}_{KSP}$ is not guaranteed to satisfy constraints t'), t'') and t'''), but it certainly includes any set $\mathcal{E}^{s,d}$ leading to an admissible routing solution. Thus, if we solve the METER problem with $\mathcal{E}^{s,d}_{KSP}$ as the set of links that are allowed to carry the flow between s and d (we denote such a problem by METER-KSP), the obtained objective value, denoted as $\Gamma(\Phi^{KSP})$, represents a lower bound to the average cost of the optimal feasible admissible routing solution.

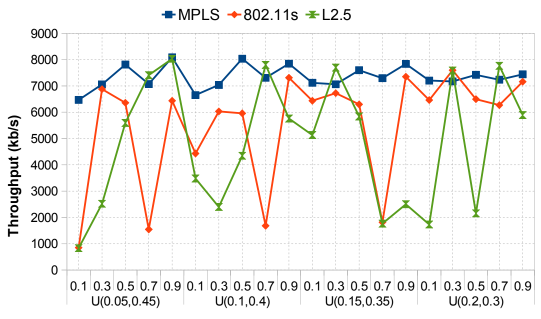
In order to compare the average cost of the routing solution returned by METER-RDAS, denoted as $\Gamma(\Phi^{RDAS})$, to the lower bound to the minimum average cost, we performed 100 different experiments with varying topologies, channel assignments, traffic loads ($\mathcal{I}^{max}$ and $\mathcal{O}^{max}$) and α values (ranging from 1.5 to 3). For each experiment, we solved both METER-RDAS and METER-KSP and com-



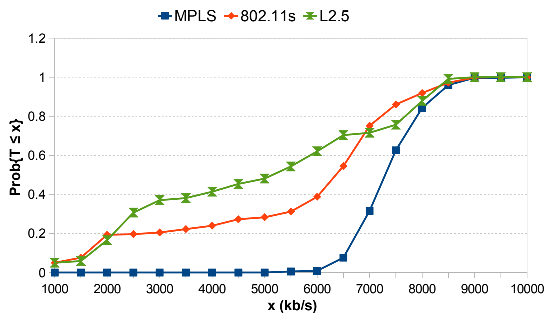
(a) Average throughput measured in each experiment



(b) CDF of the throughput measured in each time interval



(c) Average throughput measured in each experiment



(d) CDF of the throughput measured in each time interval

Figure 6.11: Average throughput achieved while varying the traffic load

puted $\delta = \dfrac{\Gamma(\Phi^{RDAS}) - \Gamma(\Phi^{KSP})}{\Gamma(\Phi^{KSP})}$, i.e., the percentage increase with respect to the lower bound. The empirical CDF (Cumulative Distribution Function) of the values of δ resulting from our experiments is shown in fig. 6.10. It can be observed that the percentage increase of the average cost of the routing solution returned when using RDAS is always below 20%, while the percentage increase is below 10% in the 70% of the cases and below 6% in the 30% of the cases. If we consider that such results refer to a comparison with a lower bound and that $\mathcal{E}^{s,d}_{KSP}$ is unlikely to lead to an admissible routing solution, we can assert that our approach to the MPLS splitting-based problem provides a routing solution whose average cost is very close to the minimum one.

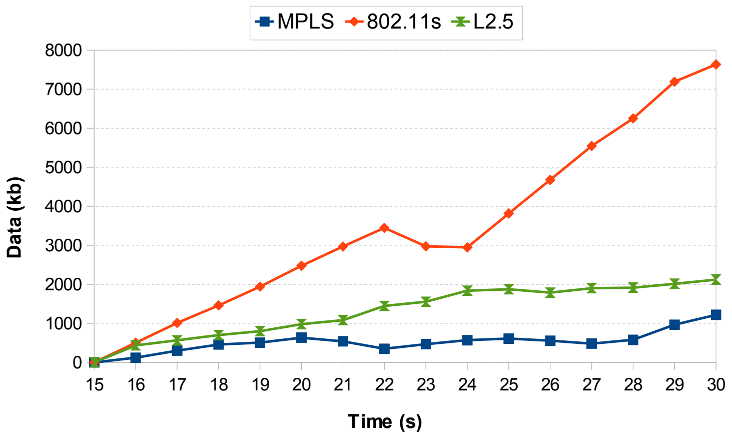
Robustness against variations in the traffic load

Our approach to solve the MPLS splitting-based routing problem has been designed to provide a set of split ratios ensuring high performance under different traffic loads. The experiments described in this section aim to show that our approach is actually more robust against variations in the traffic load than the other routing protocols. We report the results obtained for two different topologies where the maximum amount of traffic entering each edge node is uniformly distributed between $5Mbps$ and $10Mbps$. For each topology, we performed different experiments where the actual traffic load entering an ingress node and destined to a specific egress node is a random portion of the maximum amount of traffic that can enter that ingress node. For each experiment, such random values are derived from a uniform random variable U distributed between a and b, where $(a, b) \in \{(0.05, 0.45), (0.1, 0.4), (0.15, 0.35), (0.2, 0.3)\}$. Since there are 3 edge nodes, and hence the traffic entering an edge node is split between the two other edge nodes, the actual amount of traffic entering an edge node is, on the average, half the maximum amount in all the experiments. Clearly, the higher the variance of the random variable used, the higher the disproportion between the amount of flows entering an edge node and destined to the two other edge nodes. For each random variable used, we performed a number of experiments differing for the seed used to initialize the pseudo-random number generator.

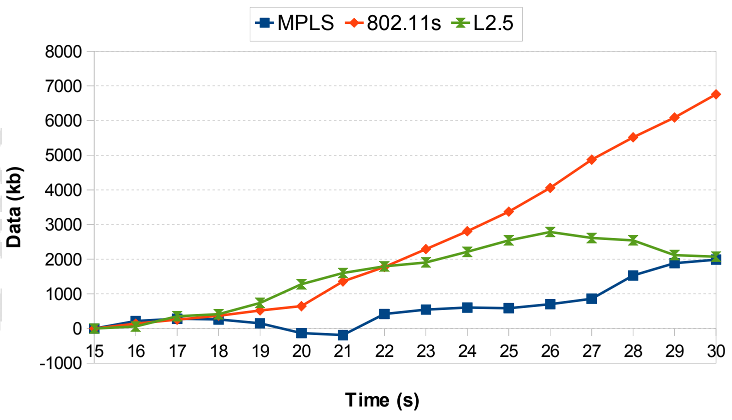
Figures 6.11a and 6.11c show the average throughput (over the whole simulation) measured in each experiment for each of the two topologies considered. On the x-axis, both the seed and the distribution of U used for each experiment are reported. We recall that all the experiments are characterized by the same (on the average) total amount of incoming traffic but they differ for how such amount is subdivided among the ingress-egress pairs. We can observe that the average throughput achieved by our MPLS splitting-based approach is much more stable over the different experiments than 802.11s and L2.5. The CDF of the throughput values measured in each $1s$ interval of all the experiments (figures 6.11b and 6.11d) also confirms that the throughput achieved by MPLS is much more stable than 802.11s and L2.5. The poor performance of 802.11s in some experiments can be explained by considering that 802.11s is a single path routing protocol and therefore, in case of a high traffic demand between an ingress-egress pair, the selected path may be easily congested, thus leading to a decrease in the throughput. The poor performance of L2.5 in some experiments can be explained by considering that each node attempts to utilize each link in proportion to predefined flow rates, that are returned by a traffic aware channel assignment algorithm. Thus, if the channel assignment has been computed based on a traffic load which is different than the actual offered load, the performance of L2.5 may decrease. Instead, as shown by such experiments, our approach to compute the split ratios ensures high robustness against variations in the traffic load.

Robustness against a single node failure

The experiments described in this section aim at evaluating the behavior of the routing protocols in the presence of a single node failure. We consider the same topologies as the previous section, but with a different channel assignment. Each experiment lasts 30 seconds and, after 15 seconds from the beginning, a node failure is simulated by increasing the noise level at every radio interface of that node to the point that the node is not able to send or receive packets. The throughput in every $1s$ interval is measured. We perform 22 experiments, each involving the failure of a different node (except the edge nodes), and compute the average



(a) Cumulative data loss



(b) Cumulative data loss

Figure 6.12: Data loss due to single node failures

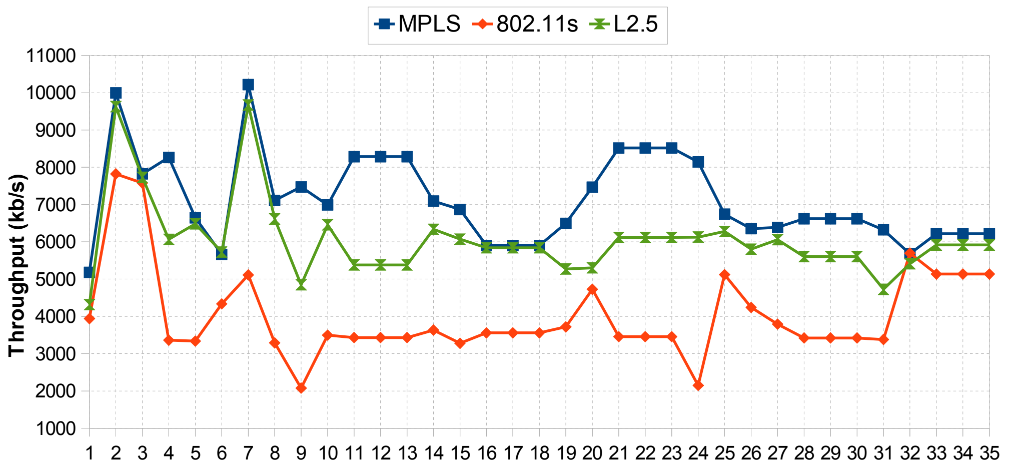


Figure 6.13: Saturation throughput

throughput in each $1s$ interval over all the experiments. In figures 6.12a and 6.12b we show, for each routing protocol, the cumulative decrease in the amount of data delivered to the egress nodes in the time intervals subsequent to the failure (averaged over all the 22 experiments) with respect to the case with no failure. It can be observed that both L2.5 and our approach based on the MPLS splitting policy outperforms 802.11s. In particular, after $10s$ from the failure, the data losses of 802.11s and L2.5 are, respectively, 6 times and around 3.5 times the data loss of our approach. We also looked at the data loss after a fixed amount of time ($10s$) since the failure for each of the 22 experiments and computed the empirical CDF. The results (averaged over multiple experiments) indicate that the probability that the data loss after $10s$ is less than $6Mb$ is 92% for our approach, 86% for L2.5 and 75% for 802.11s.

Saturation throughput

In this section we show the results of the experiments carried out to evaluate the *saturation* throughput, i.e., the maximum network throughput achieved by the routing protocols under test. We consider 35 different configurations with varying topologies, channel assignments and maximum amount of traffic entering at the edge nodes. For each configuration, the saturation throughput is determined by increasing the actual traffic load (even beyond the maximum amount) until the average throughput (over the whole simulation time) ceases to increase. The saturation throughput achieved by each routing protocol in each of the 35 configurations is reported in fig. 6.13. The average saturation throughput (over all the 35 configurations) achieved by our approach is 20% higher than L2.5 and 77% higher than 802.11s. The poor performance achieved by 802.11s can be explained by considering that it is a single-path routing protocol and hence it cannot distribute the traffic load over multiple paths. Our approach and L2.5, instead, achieve a higher throughput because they exploit such possibility. L2.5 achieves a lower throughput than our approach because it allows packets to take any path whose length does not exceed a maximum value, while with the MPLS splitting policy packets are routed according to split ratios that are the result of an optimization problem. Hence, our approach is able to better engineer how packets are routed.

6.6 Conclusion

We addressed the problem to develop a routing strategy for multi-radio wireless mesh networks that takes into account the constraints on the link bandwidth availability imposed by the current channel assignment. We proposed an approach based on MPLS, in order to ensure a tight control over the paths followed by packets. We introduced a novel mechanism, the MPLS splitting policy, which consists in allowing multiple candidate next hops at each intermediate node for a given FEC and partitioning the traffic of that FEC among such next hops in proportion to predefined split ratios. The MPLS splitting policy enables to balance the traffic load across multiple paths and allows for a fast local restoration. We then developed a technique to compute the set of split ratios in order to ensure high throughput despite variations in the traffic load. This goal is achieved by properly defining a cost function, computing directed subgraphs along which the flow of each ingress-egress pair can be routed and solving a convex optimization problem. Finally, we performed a thorough simulation study which confirmed that our approach outperforms other routing protocols in terms of network throughput and robustness against load variations and single node failures.

Submitted paper

[VI.1] S. Avallone, G. Di Stasi. A novel MPLS-based forwarding paradigm for wireless mesh networks. Submitted for revision to IEEE/ACM Transaction on Networking.

Chapter 7

AA-L2R: aggregation-aware forwarding paradigm for wireless mesh networks

7.1 Introduction

Routing algorithms based on the multi-path paradigm construct and use different paths between each couple of sender and receiver nodes. Such a technique is used in WMNs, e.g., to distribute traffic on links in order to approximate for each link a desired (pre-computed) flow rate, which represents a bandwidth limit imposed by the channel assignment algorithm. It has been shown in Sec. 5.5.1 that this allows to improve network performances, as interference is reduced. An example of an algorithm which adopts such a strategy is Layer-2.5 (sec. 4.5.4).

Another technique which allows to improve network performances is *packet aggregation*, which consists in aggregating several packets into a single transmission unit. Packet aggregation is particularly efficient when the overhead of a single transmission is high, as in case of IEEE 802.11 based networks.

The benefits of packet aggregation have been shown in several works, such as [91], [92] and [93]. The recent IEEE 802.11n [94] standard has also adopted aggregation as an optional feature to improve performance.

The joint use of packet aggregation and multi-path routing may result in sub-optimal performance. Indeed, multi-path routing spreads packets among different next-hop neighbors, whereas only packets that are about to be forwarded to the same next hop can be aggregated. Because of this trade-off, it is important to properly combine multi-path and packet aggregation in order to have the advantages of both approaches.

In the following, we present a new multi-path forwarding paradigm for wireless mesh networks. We define this paradigm *aggregation aware* as it conditions the choice of the next hop to the packet aggregation that can be achieved. Indeed, it tries to combine the advantages of multi-path routing and packet aggregation by both approximating the flow rates and exploiting aggregation opportunities when they arise. The algorithm makes use of cross-layer information, such as the status of the sending queues, and makes the forwarding decision so as to improve packet aggregation and hence MAC layer efficiency.

Simulation results show that our approach can improve network throughput by up to 15 percent and average delay by up to 25 percent.

The algorithm has been designed and tested in collaboration with some researchers of the COMICS group [22] and some researchers of the Computer Networking group of the University of Karlstad [VII.1].

The rest of the chapter is structured as follows. In section 7.2 we introduce our forwarding paradigm and some other paradigms we use as comparison. In section 7.3 we present simulation results. In section 7.4 we relate the proposed forwarding paradigm with the state-of-the-art. Finally, in section 7.5 we draw our conclusions.

7.2 Aggregation aware forwarding

The proposed forwarding paradigm needs to be aware, at each hop, of the set of candidate next-hops that are feasible for transmission. Any multi-path routing algorithm can be used to construct this set. We opted for a multi-path routing algorithm that makes use of the hop-count metric and considers all the paths with a length equal or smaller than a certain threshold [44].

In addition to the information regarding the available paths, the proposed forwarding paradigm needs to receive support by the datalink layer. In particular, the datalink layer has to support packet aggregation and export some information relative to the sending queues. As the datalink layer of IEEE 802.11a/b/g based interfaces does not support aggregation, we implemented an aggregation module, to be put between the datalink layer and the routing layer, that performs aggregation and export the required information, as described in the following section.

7.2.1 Packet aggregation module

The basic objective of packet aggregation is to improve MAC layer efficiency by aggregating several packets into a single transmission unit. This reduces the number of MAC-layer transmissions and the related overhead and significantly reduces contention for highly congested links. We use hop-by-hop aggregation, as it provides more aggregation opportunities than end-to-end aggregation [95]. Here, every node independently aggregates packets which should be transmitted to the same next hop. At the receiving interface, the node de-aggregates an aggregated packet and inserts the de-aggregated packets into the local network stack.

A single transmission unit is called *an aggregation packet*. Such aggregation packets can be link layer frames or IP packets, depending on the particular technique used. In this work we have used IP packets, as in [96] and [97]. since this allows to deploy aggregation without requiring changes or native support at the MAC layer. Therefore, our system can be used with 802.11a/b/g interfaces which, in contrast to 802.11n, do not support MAC layer aggregation. In our approach, an aggregation packet will contain several IP packets. An additional aggregation header allows to distinguish between aggregated and un-aggregated packets. A packet *is aggregated* when it is combined with at least *one other packet* inside an aggregation packet. We measure the efficiency of our method by calculating the aggregation ratio, which denotes the percentage of packets which *are aggregated*.

We implemented packet aggregation in an *aggregation module,* which extends the functions of a network interface. Such aggregation module stores and aggregates packets before passing them to the real network interface for transmission. Internally, it stores the packets in different queues, one for each next hop neighbor that can be reached through the network interface. When a packet is received by the aggregation module, it is timestamped and put into the appropriate queue, i.e. the queue of the next-hop the packet is destined to. The time-stamp is used later to determine how long the packet has been queued already. If there are more packets in a queue that can fit inside one MAC frame, an aggregation event is triggered, i.e. the aggregation module aggregates as many packets as possible from the queue while respecting the packet order. The resulting aggregation packet is sent as soon as the interface becomes ready. The remaining packets are left in the queue for the next aggregation event. An aggregation event is also triggered when the network interface becomes ready to transmit and one of the queues has packets to send. If more queues are ready, the queue with the *oldest* packet is served first to avoid starvation.

Packets can be purposely delayed in order to increase the aggregation ratio. The maximum amount of such artificial delay is controlled by the *Aggregation-*

MaxDelay parameter. An aggregation event is triggered when the first packet in (any) queue has stayed for at least *AggregationMaxDelay* time. When the network traffic is low, this parameter induces artificial delay, which increases the number of packets in the queue and thereby increases the aggregation ratio. When traffic is high, typically a queue contains enough packets to fill one MAC frame and packets are normally not delayed. Note that slightly delaying packets for aggregation may actually reduce the total end-to-end delay in a multi-hop environment. The reason is that the reduced contention implies reduced back-off times and fewer packet re-transmissions. This has been shown to be beneficial even for VoIP services, where the controlled delay introduced to aggregate packets increased the total achievable mean opinion score (MOS) [98].

The main idea of this paper is to exploit information on aggregation opportunities at the forwarding layer (see next section), which decides for each packet which next hop to use. Therefore, we inform the packet scheduler about the available space left in each aggregation queue and the remaining time for each aggregated packet to be sent.

7.2.2 Forwarding Strategies

In this section, the forwarding paradigms we consider for the evaluation are described.

L2.5R (Layer 2.5 Routing)

The first forwarding paradigm considered is L2.5R which has been defined in Sec. 4.5.4. It selects the next-hop in the set of candidate next-hops in order to approximate the flow rate defined for each link.

AA-L2R (Aggregation-Aware L2R)

AA-L2R (Aggregation Aware L2R forwarding paradigm), our proposed forwarding paradigm, selects the next-hop prioritizing the increase of the aggregation ratio over the fulfillment of the flow rates. The key idea is to first find all potential queues related to next-hops that allow the packet to be aggregated and then

```
AGGRAWARE-L2R($C, p$)
 1  $A \leftarrow \emptyset$
 2  for each $n \in C$
 3        do if $isNotEmpty(n)$ and $spareSpace(n) >= dim(p)$
 4              then $addElement(A, n)$
 5  if $dim(A) > 0$
 6     then $q = findFlowrateQueues(A)$
 7          $enqueue(p, q)$
 8     else  $F = findFreeQueues(C)$
 9          if $dim(F) > 0$
10             then $q = findFlowrateQueue(F)$
11                  $enqueue(p, q)$
12             else  $q = findFlowrateQueue(C)$
13                  $enqueue(p, q)$

FINDFREEQUEUES($B$)
1  $F \leftarrow \emptyset$
2  for each $q \in B$
3        do if $isEmpty(q)$
4              then $addElement(F, q)$
5  return $F$
```

Figure 7.1: Aggregation aware L2R pseudo-code

from all those queues to pick the one which best fulfills the flow-rates. If a neighbor associated to one of these queues is selected as next-hop, the packet can be aggregated with packets already in the queue. This reduces MAC layer overhead, because of the saved transmission as packets are aggregated before sending.

In Figure 7.1 we show the pseudo-code of AA-L2R. We define the *aggregation set A* for the given packet as the set of next-hops associated with queues that offer an aggregation opportunity, i.e. which allow to aggregate. A next-hop belongs to the set *A* if it belongs to the set of candidate next-hops and the following conditions hold: i) the associated queue is not empty; and ii) the *spare space* (SP) of the associated queue is greater than the packet size. The *spare space* of the generic queue i is defined as in the following:

$$SP = MTU - \sum_{p \in Q_i} p_{size} - H \tag{7.1}$$

where *MTU* is the Maximum Transmission Unit, *H* the aggregation header size, Q_i the set of packets in queue i.

If the aggregation set is empty, the packet cannot be aggregated at the moment. However, it could be aggregated with packets yet to come. This happens because the queues are empty or because the queues which are not empty do not have enough spare space to aggregate the packet. Here, the *free aggregation set F*, which contains the empty queues, is evaluated (see *FindFreeQueues* function). By choosing a queue of the free aggregation set, we leave unchanged the queues which hold at least a packet and can potentially aggregate a packet which arrives later on. In addition, the selected empty queue becomes as well a potential source of aggregation. If also the free aggregation set is empty, all queues hold some packets with a spare space smaller than the packet size. In this case, all the queues belonging to the candidate next-hops are considered eligible for sending the current packet. Once the eligible set of next-hops has been chosen, the L2.5R criteria is applied (by the *FindFlowrateQueue* function) for selecting among the specific next-hops to try to approximate the flow-rates (by applying Equation 4.4).

In Figure 7.2 we report an example of the forwarding process. A mesh node with three neighbors and 6 packets to forward is shown.

As can be seen, packets are placed in the different queues so as to maximize the aggregation ratio.

On the bottom of each packet we indicate the set of candidate next-hops, as given by the routing algorithm. When the forwarding decision has to be taken for packet 1, all the queues are empty. This means that the *aggregation set* is empty, i.e. no aggregation possibility exists, so the queue relative to the link with the greatest cost (as given by Eq. 4.4) is chosen, i.e. the queue B. Also for packet 2, the *aggregation set* is empty. In fact, even if the queue B would have a packet queued, the spare space is not enough to aggregate packet 2. The free candidate set is then evaluated, which returns the queue A. For packet 3, the aggregation set is equal to $\{A, B\}$. Both the queues A and B have a packet queued and the spare

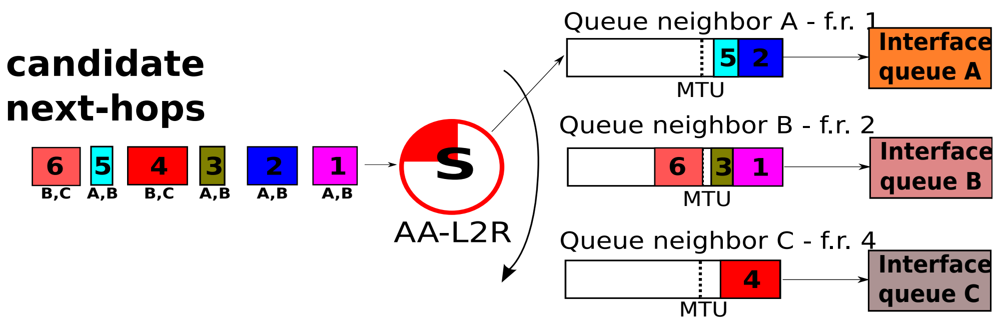


Figure 7.2: Forwarding process example (f.r. stands for flow-rate). On the bottom of each packet the set of candidate next-hops is shown.

space is enough for both to aggregate packet 3. The queue B is then chosen, due to the flow-rate criteria. For packet 4, the aggregation set is empty, while the free candidate set is equal to *C*, so C is chosen. Finally, for packet 5, the aggregation set is equal to *A* (as *B* has a waiting packet but the spare space is not enough to contain 5 packet), so A is selected. We suppose that, when the routing decision for packet 6 has to be taken, packets from 1 to 5 are still waiting in the queue, i.e. in the aggregation module. That can happen because of a artificial delay added in the aggregation module or when the interfaces are busy sending (or receiving) previously en-queued packets.

RR (Round-Robin)

Another forwarding strategy is *round-robin* (RR). Here, the traffic is load-balanced equally between all candidate next-hops. This will give the least possibilities for aggregation and most reordering since the packets have the largest possible spread among the candidate next-hops.

7.3 Evaluation

We performed a number of NS-2 [4] simulation studies to evaluate the performance of the considered forwarding paradigms. Unless otherwise noted we used the default NS-2.32 settings. The MAC/PHY layer was configured to simulate an IEEE 802.11 MAC/PHY layer with the MAC MTU set to 2304 bytes, in order

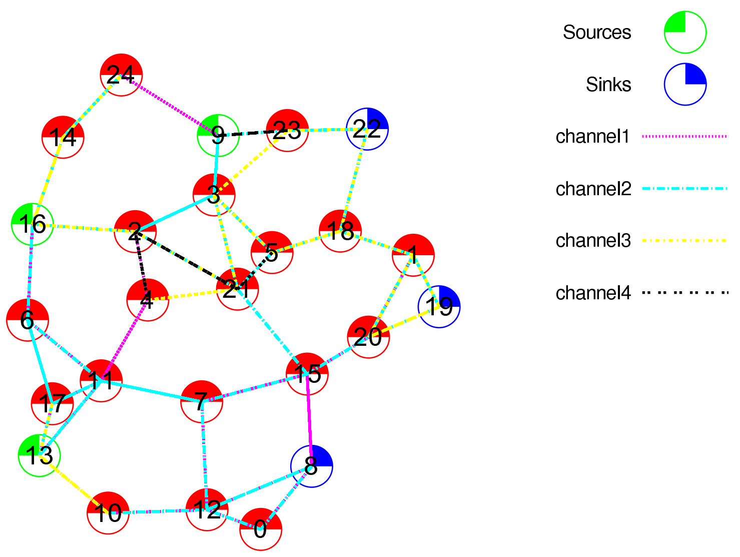


Figure 7.3: Simulated topology

to make the aggregation capabilities compliant with IEEE 802.11a/b/g standards [99].

In the simulations we use 54 Mbps PHY layer speed, and TCP Newreno[100] with selective acknowledgment (Sack) [101]. We considered a randomly generated topology of 25 nodes placed in an area of 300x300 meters (see Figure 7.3). Each node was equipped with a maximum of 3 radio interfaces. We randomly selected three source and sink nodes placed at opposite sides in the network. In future work, we will use significantly larger topologies and traffic configurations.

7.3.1 Methodology

We evaluated the behavior of the considered forwarding paradigms under two different traffic classes. The first class consisted of three UDP flows between each source-destination pair, resulting in 27 flows in total. Each flow had an exponential ON-OFF behavior, an average “on” time of 5 seconds and an average “off” time of 1 second.

The packet size was different for each flow: 200 bytes for the first, 700 for the second and 1400 for the third. The inter-packet departure time was set so as to have 380 kbit/s of bitrate for each flow, which implied the generation of 10 Mbit/s of bitrate in total by the sources. The second class consisted of TCP flows which were generated between each pair of source and sink nodes. We assumed FTP type traffic with infinite backlog to simulate large file transfers with a segment size of 1460, corresponding to a common IP packet size [102]. With a segment size of 1460 bytes, only one TCP DATA packet can fit inside each aggregation packet, which means that there is less room for improvement by aggregation. However, multiple TCP ACKs can be aggregated together with both TCP DATA and other TCP ACK packets.

Traffic generation was started after 9s of delay to allow the network to stabilize routing. Each simulation was run for 310 seconds with 25 repetitions. The output of the simulations was statistically collected and analyzed using the tool from [103]. The results are shown relative to the value of the *AggregationMaxDelay* parameter (x-axis), i.e. the amount of artificial delay, as inserted by the aggregation algorithm. In all simulations only shortest hop paths were used.

7.3.2 Simulation Results

UDP

In case of a highly loaded network, there is basically a high amount of contention among nodes for accessing the shared medium. This wastes network resources, since the nodes spend time in trying to obtain the medium, rather than actually sending packets. Figure 7.4 shows the average aggregation ratio with varying *AggregationMaxDelay*. The aggregation ratio is up to 48 percent higher for AA-L2R compared to RR and up to 23 percent better than L2.5R. AA-L2R also slightly improves the end-to-end throughput and significantly reduces end-to-end delay up to 25 percent over RR and 17 percent over L2R (see Figure 7.5 and 7.6). Packet aggregation can greatly help to reduce contention by reducing the number of packets to be sent. By exploiting the knowledge on the internal state of the queues made available by the aggregation module, AA-L2R can reduce the amount of contention compared to strategies that do not consider aggregation possibilities. We can also observe that both AA-L2R and L2R have a significantly better through-

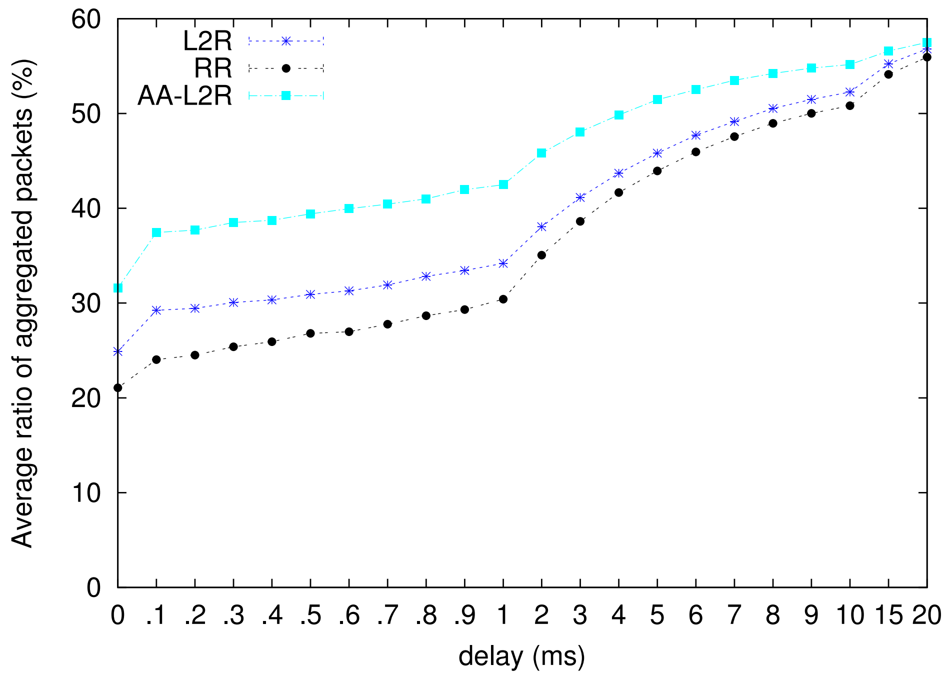


Figure 7.4: Average Aggregation ratio for UDP traffic scenario.

put than RR, which can be explained by the fact that RR neither approximates the flow-rates nor tries to increase the aggregation ratio.

TCP

TCP simulation results, which we omit due to space constraints, show that the aggregation ratio is almost identical for all three schemes when using TCP traffic with the full segment size of 1460 bytes. This follows earlier results in e.g. [92] where it is shown that large packets limit the improvement due to aggregation. Furthermore, as the aggregation is TCP un-aware, the artificial delay can increase TCP RTT as TCP DATA packets will be delayed but rarely can be aggregated, since we only have TCP DATA packets flowing in one direction.

When *AggregationMaxDelay* is smaller than $< 1ms$, both TCP round-trip time (RTT) and packet loss (including reordered packets) are similar or slightly lower for AA-L2R compared to both RR and L2R (omitted due to space constraints). The slightly lower RTT is reflecting the improved MAC layer efficiency

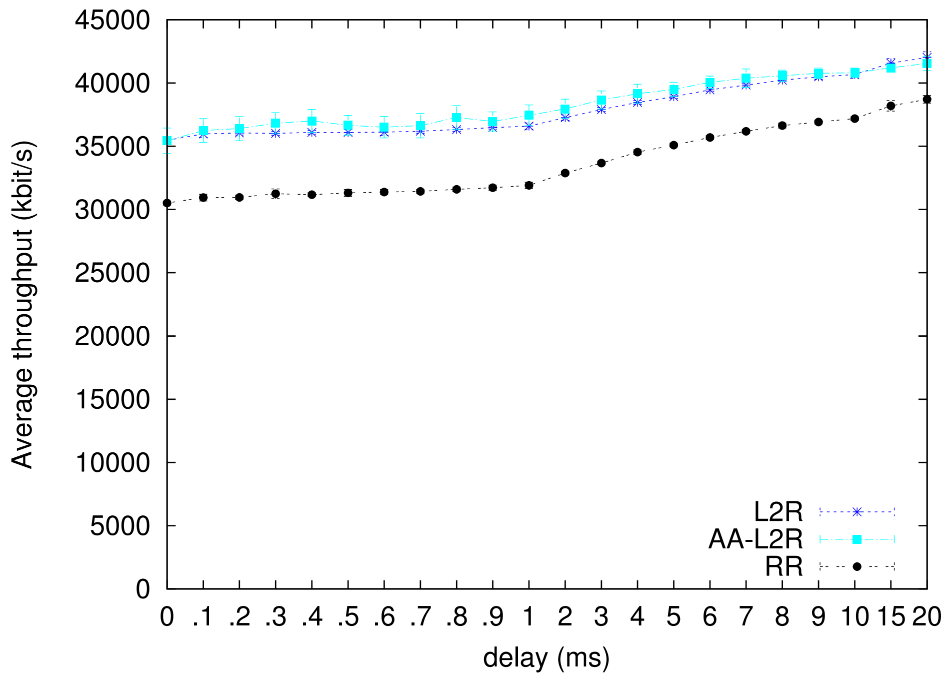


Figure 7.5: Average aggregate throughput for UDP traffic scenario.

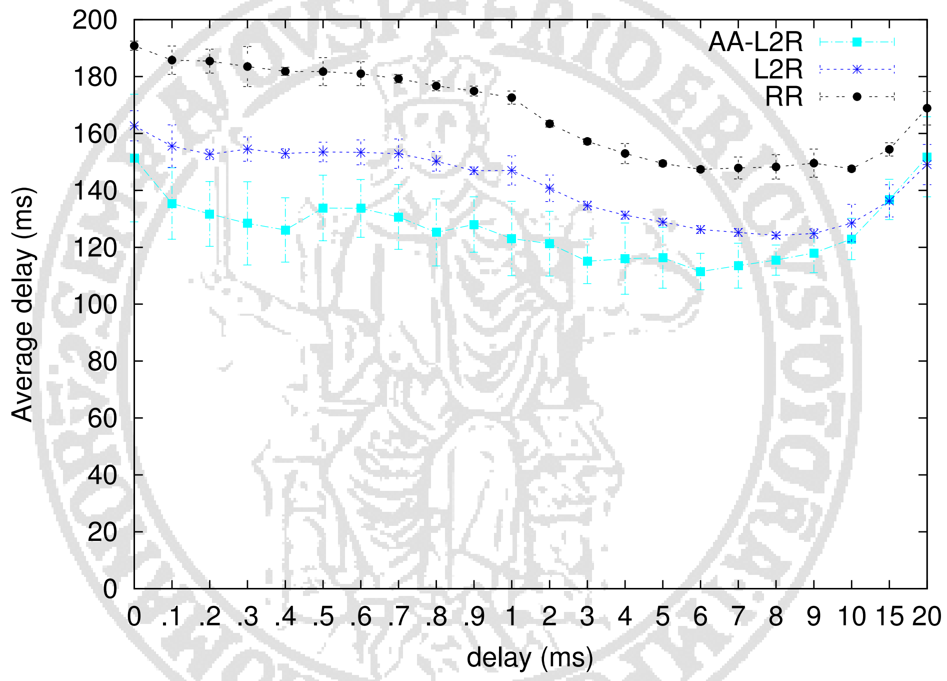


Figure 7.6: Average delay for UDP traffic scenario.

due to the more effective aggregation of TCP ACKs with AA-L2R. When *AggregationMaxDelay* is increased, the aggregation awareness of AA-L2R becomes a disadvantage.

With large TCP DATA packets and no reverse flows, TCP DATA can NOT be aggregated with TCP ACKs, AA-L2R will thus place them in empty queues. Therefore, TCP DATA packets will be delayed longer than when using AA-L2R compared to RR and L2R where more often a already occupied queue will be used and hence trigger an aggregation event. Although no TCP DATA packets will be aggregated due to this aggregation event it will make the first TCP DATA packet in the queue to be forwarded before the *AggregationMaxDelay* timer is due and therefore reduce packet delay. The preference of queues that are empty will also make AA-L2R switch next hop neighbors more often for packets within the same TCP flow than L2R, causing up to two times the number of reordered packets compared to L2R (see Figure 7.7). The highest amount of reordered packets however is experienced by RR. However, simulation results omitted due to space constraints show that the packet displacement is higher for AA-L2R indicated by a higher delay variation. This is due to the more aggressive use of empty queues by AA-L2R whereas RR uses all queues in a uniform manner.

As can be seen from Figure 7.8, TCP throughput is slightly lower for AA-L2R compared to both RR and L2R. This follows from the higher number of retransmitted TCP packets that can be observed in Figure 7.9. Since we use the TCP sack option, both the amount of reordering and the magnitude of packet displacement impact the amount of resent TCP packets. Additional simulation results, not shown here, indicate that RR has the lowest amount of TCP timeouts and that the increase in throughput for RR when *AggregationMaxDelay* is varied between 2 and 4 ms is due to a slight reduction of the number of lost packets and a maintained TCP RTT, compared to AA-L2R and L2R, in these simulations. This effect is due to a synchronization effect where a slightly higher amount of packets, not necessarily within the same flow, use the same next-hop neighbour with RR. This increases the aggregation ratio and reduces the contention and therefore the time packets spend in queue.

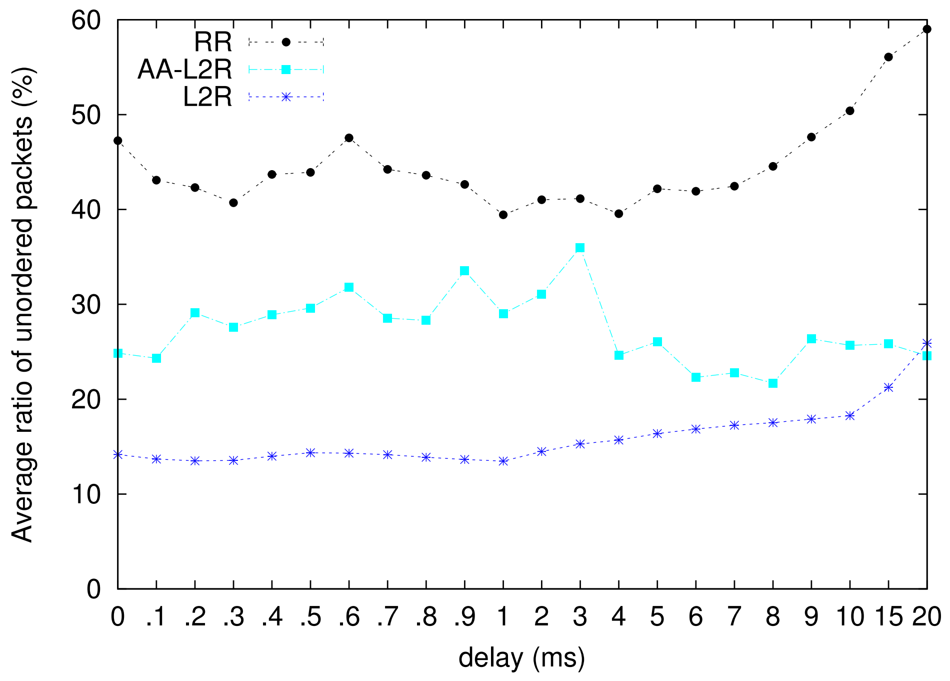


Figure 7.7: Packet reordering with TCP traffic

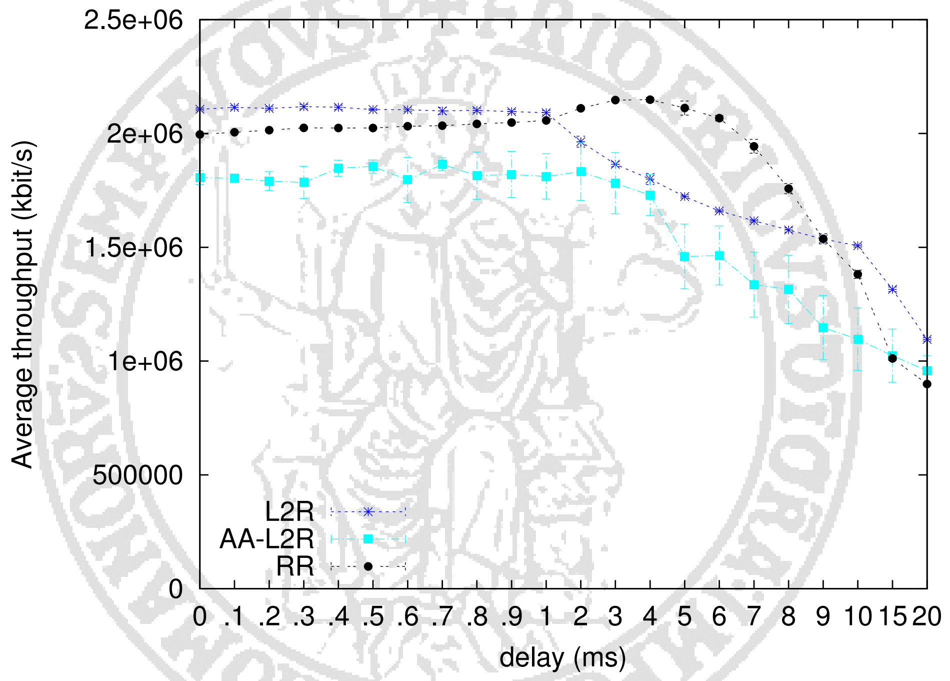


Figure 7.8: Average aggregate TCP throughput

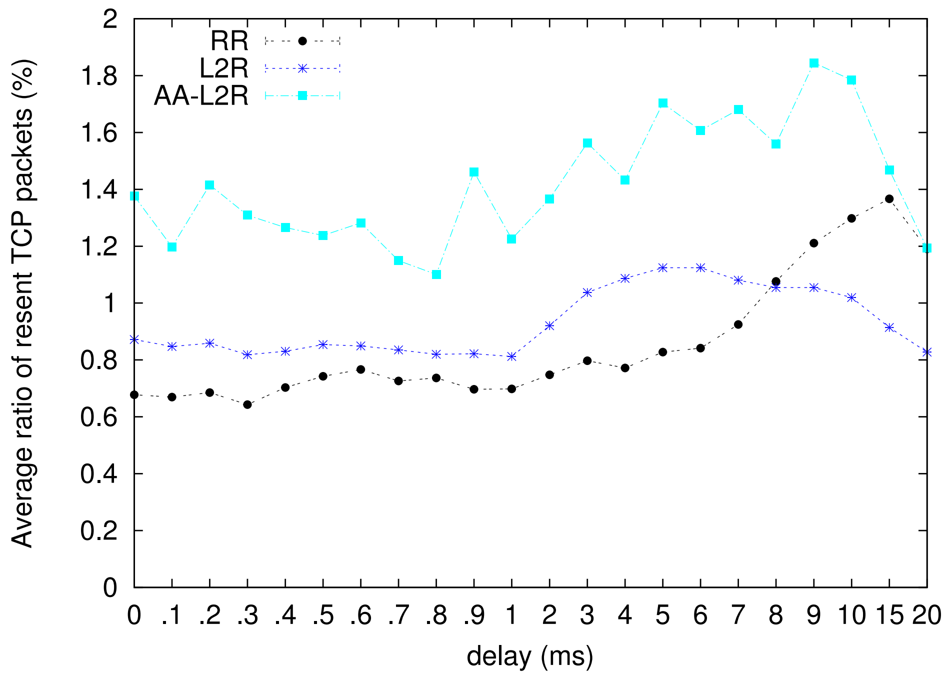


Figure 7.9: TCP Retransmitted packets

7.4 Related work

Several studies have considered the usefulness of packet aggregation on wireless networks in a great variety of operational conditions. For instance, the performance measurement study done in [104] has shown an improvement of up to 160 percent of throughput in a single-hop scenario with mixed traffic. In [92], TCP performance in small topologies with no hidden nodes was studied and an improvement of up to 73 percent was shown compared to not using aggregation. In [91] packet aggregation was studied with the objective of deriving the dependence of the throughput to the size of aggregation packets and to the link quality. Modern IEEE 802.11 WiFI cards change modulation schemes to compensate for changes in BER. For this reason, in this work we focused on the integration between packet aggregation and routing and have only used “good” links.

The recent WiFI standard IEEE 802.11n [94] exploits packet aggregation to improve performance [105]. In particular, the standard defines two different strategies for aggregation, one where the aggregation is done when packets enter the MAC layer (A-MSDU) and one when the packets leave the MAC layer (A-MPDU). The aggregation strategy most similar to the one performed by our aggregation module is the A-MSDU. We have however limited the size of the aggregated packets to 2304 bytes in order to be compliant to IEEE 802.11a/b/g devices whereas an IEEE 802.11n A-MSDU is allowed to be 7935 bytes. However, even if several works have considered the effect of packet aggregation on wireless networks, no study, to the best of our knowledge, is available which combines aggregation aware routing and forwarding for multi-radio WMNs.

7.5 Conclusions and future works

In this chapter a new multi-path aggregation aware forwarding strategy for WMNs has been described. The forwarding strategy selects for each packet a next-hop based on a set of candidate next-hops trying to both increase packet aggregation , in order to reduce the overhead of transmissions, and approximate the flow-rates, which represent bandwidth limits imposed by the channel assignment algorithm. While the former allows to reduce the overhead of transmissions, the latter reduces interferences, as links will carry an amount of flow proportional to their available bandwidth.

Simulation results show that the proposed aggregation aware forwarding paradigm is able to significantly increase the aggregation achieved leading to a lower delay and packet loss for user traffic. However, packet re-ordering might negatively impact TCP performance.

When using multi-path routing, packet reordering is a well known problem for TCP and causes severe performance degradation [106, 107]. In a general purpose WMN, it can be anticipated that most clients will run standard TCP variants that have limited mechanisms to handle reordering. As future work, we will therefore investigate approaches to minimize the effect of packet reordering, e.g. using TCP flow aware forwarding strategies.

Relevant publications

[VII.1] J. Karlsson, A. Kassler, G. Di Stasi, S. Avallone. “An Aggregation Aware Multi-Path Forwarding Paradigm for Wireless Mesh Networks”. MESHTEC, October 17, 2011. Valencia

Chapter 8

Conclusions

In this thesis we have described our contributions related to the definition of new channel assignment and routing algorithms for wireless mesh networks. Routing and channel assignment in wireless mesh networks are shown to be not independent problems and therefore need to be solved jointly. The joint problem is shown, however, to be NP-complete, which leads to solve the two problems separately. A common approach is to first solve the routing problem, whose solution gives a set of flow rates representing the amounts of flow to be routed on each link. Then, the channel assignment problem is solved, through heuristics that take into account the flow-rates and try to assign to each link the required bandwidth.

In this thesis we propose an heuristic for channel assignment that, differently from previous proposals, takes into account the current channel allocation and switches a defined maximum number of channels to adapt to new traffic demands. Our evaluation campaign shows that the proposed algorithm is able to maintain and even exceed the performance of previous algorithms, while switching channel of a limited number of radios. Thanks to the reduction of the number of channel switches, the algorithm is able to limit the transient period when the network throughput is reduced because of the time required to the radios to move to the new channels and to the routing algorithm to adapt to the new network setup.

We have also proposed a new MPLS standard compliant forwarding paradigm which is able to take into account the bandwidth constraints defined by the channel assignment algorithm [1] and, in addition, to be robust against failures, to avoid routing cycles and to give good results even for varying traffic demands. Through an extensive evaluation campaign we have verified these properties by comparing it to other paradigms, such as the one defined in the IEEE 802.11s standard.

We have proposed, moreover, a second forwarding paradigm for wireless mesh networks that tries to combine routing and packet aggregation. Such a forwarding paradigm, defined aggregation aware, takes the routing decision not only to make packets advance towards the destination, but also to exploit aggregation opportunities at the MAC layer when they arise. An extensive evaluation campaign shows that the proposed paradigm is able to significantly increase the aggregation

[1]The flow rates resulting from the initial solution of the routing problem.

ratio, to reduce the end-to-end average delay and to increase the aggregate network throughput for UDP traffic up to respectively 25% and 15%. Such a strategy can be applied to any multi-path routing paradigm adding to it very little complexity.

Before introducing the proposed algorithms, we have also described some tools, platforms and methodologies we have leveraged for their evaluation. We have also described a contribution related to heterogeneous networking testbeds which consists in the possibility of making experiments which involve the PlanetLab infrastructure and globally-distributed local OMF wireless mesh testbeds, used as access networks. Such an achievement, which is a first step towards a full-around federation of PlanetLab and OMF-based testbeds, makes it possible to test new channel assignment and routing algorithms in heterogeneous scenarios, which are able to better approximate the behavior of real networks.

Bibliography

[1] I. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Computer Networks,*, vol. 47, no. 4, pp. 445–487, 2005.

[2] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," IETF, RFC 3031, January 2001.

[3] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: An Overlay Testbed for Broad-Coverage Services," *ACM SIGCOMM Computer Communication Review,*, vol. 33, no. 3, pp. 3–12, July 2003.

[4] S. McCanne, S. Floyd, K. Fall, K. Varadhan *et al.*, "Network simulator ns-2," *The Vint project, available for download at http://www.isi.edu/nsnam/ns*.

[5] "Linux Packet Sockets," http://linux.die.net/man/7/packet.

[6] " MIT Object Tcl ." [Online]. Available: http://otcl-tclcl.sourceforge.net/otcl/

[7] "OMNeT Network Simulation Framework," http://www.omnetpp.org/.

[8] D. Raychaudhuri, M. Ott, and I. Secker, "ORBIT Radio Grid Tested for Evaluation of Next-Generation Wireless Network Protocols," in *Proceedings of TridentCom 2005*, Trento (Italy), February 2005, pp. 308–309.

[9] M. Ott, I. Seskar, R. Siraccusa, and M. Singh, “ORBIT Testbed Software Architecture: Supporting Experiments as a Service,” in *Proceedings of TridentCom 2005*, Trento (Italy), February, pp. 136–145.

[10] P. Saint-Andre, “RFC 3921, Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence,” 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3921.txt

[11] J. Ledlie, P. Gardner, and M. Seltzer, “Network Coordinates in the Wild,” in *Proceedings of NSDI 2007*, Cambridge (MA, USA), April 2007.

[12] H. Pucha, Y. C. Hu, and Z. M. Mao, “On the impact of research network based testbeds on wide-area experiments,” in *Proceedings of ACM IMC ’06*, Rio de Janeiro (Brazil), December 2006.

[13] A. Botta, R. Canonico, G. D. Stasi, A. Pescapè, G. Ventre, and S. Fdida, “Integration of 3G Connectivity in PlanetLab Europe,” *ACM/Springer Mobile Networks and Applications (MONET),*, vol. 15, no. 3, pp. 344–355, 2010.

[14] M. Dischinger, A. Haeberlen, I. Beschastnikh, K. P. Gummadi, and S. Saroiu, “Satellitelab: adding heterogeneity to planetary-scale network testbeds,” *SIGCOMM Computer Communication Review,*, vol. 38, no. 4, pp. 315–326, 2008.

[15] L. Peterson, S. Muir, T. Roscoe, and A. Klingaman, “PlanetLab Architecture: An Overview,” PlanetLab Consortium, Tech. Rep. PDN–06–031, May 2006.

[16] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, “Container-based operating system virtualization: a scalable, high-

performance alternative to hypervisors," *ACM SIGOPS Operating Systems Review,*, vol. 41, no. 3, pp. 275–287, 2007.

[17] L. Paterson and T. Roscoe, "The Design Principles of PlanetLab," *ACM SIGOPS Operating Systems Review,*, vol. 40, no. 1, pp. 11–16, January 2006.

[18] L. Peterson, V. Pai, N. Spring, and A. Bavier, "Using PlanetLab for Network Research: Myths, Realities, and Best Practices," PlanetLab Consortium, Tech. Rep. PDN–05–028, June 2005.

[19] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek, "The click modular router," *ACM Transactions on Computer Systems (TOCS)*, vol. 18, no. 3, pp. 263–297, 2000.

[20] B. Blywis, M. Günes, F. Juraschek, and J. Schiller, "Trends, advances, and challenges in testbed-based wireless mesh network research," *ACM/Springer Mobile Networks and Applications (MONET),*, vol. 15, no. 3, pp. 315–329, June 2010.

[21] T. Rakotoarivelo, M. Ott, G. Jourjon, and I. Seskar, "OMF: A control and management framework for networking testbeds," *ACM SIGOPS Operating Systems Review,*, vol. 43, no. 4, pp. 54–59, 2009.

[22] "COMICS (COMputer for Interactions and CommunicationS) group ," http://www.comics.unina.it/ .

[23] " NITLAB ," http://nitlab.inf.uth.gr/NITlab/ .

[24] T. Magedanz and S. Wahle, "Control framework design for future internet testbeds," *e & i Elektrotechnik und Informationstechnik*, vol. 126, pp. 274–279, 2009.

[25] GENI Planning Group, "GENI Design Principles," *IEEE Computer,*, vol. 39, no. 9, pp. 102–105, 2006.

[26] C. Elliott, "GENI: Opening up new classes of experiments in global neworking," *IEEE Internet Computing,*, vol. 14, no. 1, pp. 39–42, 2010.

[27] A. Gavras, A. Karila, S. Fdida, M. May, and M. Potts, "Future Internet research and experimentation: the FIRE initiative," *SIGCOMM Computer Communication Review,*, vol. 37, no. 3, pp. 89–92, 2007.

[28] A. Anadiotis, A. Apostolaras, D. Syrivelis, T. Korakis, L. Tassiulas, L. Rodriguez, I. Seskar, and M. Ott, "Towards maximizing wireless testbed utilization using spectrum slicing," in *Proceedings of TridentCom 2010*, Berlin (Germany), May 2010.

[29] "VNET+ subsystem of PlanetLab ," http://www.cs.princeton.edu/ sapanb/vnet/.

[30] S. Bhatia, "VSys: A Privilege Allocation Tool," Princeton university, Tech. Rep., September 2008. [Online]. Available: http://www.cs.princeton.edu/~sapanb/vsys/vsys.pdf

[31] Y. Liu, H. Zhang, W. Gong, and D. Towsley, "On the Interaction Between Overlay Routing and Underlay Routing," in *Proceedings of IEEE INFOCOM 2005*, Miami (Florida, USA), March 2005, pp. 2543–2553.

[32] V. Aggarwal, A. Feldmann, and C. Scheideler, "Can ISPs and P2P systems co-operate for improved performance?" *ACM SIGCOMM Computer Communications Review (CCR)*, vol. 37, no. 3, pp. 29–40, July 2007.

[33] H. Xie, R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Provider Portal for Applications," *ACM SIGCOMM Computer Communications Review (CCR)*, vol. 38, no. 4, pp. 351–362, October 2008.

[34] M. K. Stuart Cheshire and K. Sekar, "Nat port mapping protocol (nat-pmp)," Tech. Rep., April 2008. [Online]. Available: http://files.dns-sd.org/draft-cheshire-nat-pmp.txt

[35] J. Seedorf and E. Burger, "RFC5693 - Application-Layer Traffic Optimization (ALTO) Problem Statement," Oct. 2009. [Online]. Available: http://www.faqs.org/rfcs/rfc5693.html

[36] M. Alicherry, R. Bhatia, and E. Li, "Joint Channel Assignment and Routing for Throughput Optimization in Multi-Radio Wireless Mesh Networks," in *Proc. of ACM MobiCom '05*, 2005, pp. 58–72.

[37] S. Avallone, I. F. Akyildiz, and G. Ventre, "A channel and rate assignment algorithm and a layer-2.5 forwarding paradigm for multi-radio wireless mesh networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 1, pp. 267–280, February 2009.

[38] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu, "Impact of Interference On Multi-hop Wireless Network Performance," in *Proc. of ACM MobiCom*, 2003, pp. 66–80.

[39] S. Avallone, F. P. D'Elia, and G. Ventre, "Layer-2.5 routing in multi-radio wireless mesh networks," in *Proceedings of IEEE WiMesh*. IEEE, June 2009, pp. 19–24.

[40] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR) RFC 3626," *Acessado em*, vol. 10, 2009.

[41] ——, “Optimized Link State Routing Protocol (OLSR),” IETF, RFC 3626, Ottobre 2003.

[42] “UniK Olsrd,” http://www.olsr.org.

[43] C. Perkins, E. Belding-Royer, and S. Das, “Ad hoc On-Demand Distance Vector (AODV) Routing,” IETF, RFC 3561, July 2003.

[44] S. Avallone, I. Akyildiz, and G. Ventre, “A channel and rate assignment algorithm and a layer-2.5 forwarding paradigm for multi-radio wireless mesh networks,” *Networking, IEEE/ACM Transactions on*, vol. 17, no. 1, pp. 267–280, 2009.

[45] “Orbit Radio Grid Testbed,” http://www.orbit-lab.org/.

[46] J. Tang, G. Xue, and W. Zhang, “Interference-Aware Topology Control and QoS Routing in Multi-Channel Wireless Mesh Networks,” in *Proc. of ACM MobiHoc*, 2005, pp. 68–77.

[47] M. K. Marina and S. R. Das, “A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks,” in *IEEE BroadNets*, vol. 1, 2005, pp. 381–390.

[48] K. N. Ramachandran, E. M. Belding, K. C. Almeroth, and M. M. Buddhikot, “Interference-aware channel assignment in multi-radio wireless mesh networks,” in *Proc. of IEEE INFOCOM*, 2006, pp. 1–12.

[49] K. Xing, X. Cheng, L. Ma, and Q. Liang, “Superimposed code based channel assignment in multi-radio multi-channel wireless mesh networks,” in *Proc. of ACM MobiCom*, 2007, pp. 15–26.

[50] H. Skalli, S. Ghosh, S. Das, L. Lenzini, and M. Conti, “Channel assignment strategies for multiradio wireless mesh networks: Issues and solutions,”

IEEE Communications Magazine, vol. 45, no. 11, pp. 86–95, November 2007.

[51] A. Subramanian, H. Gupta, S. R. Das, and J. Cao, "Minimum Interference Channel Assignment in Multi-Radio Wireless Mesh Networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 12, pp. 1459–1473, December 2008.

[52] A. Raniwala and T. Chiueh, "Architecture and Algorithms for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network," in *Proc. of IEEE INFOCOM*, vol. 3, 2005, pp. 2223–2234.

[53] A. Dhananjay, H. Zhang, J. Li, and L. Subramanian, "Practical, distributed channel assignment and routing in dual-radio mesh networks," in *Proc. of ACM SIGCOMM*, August 2009, pp. 99–110.

[54] A. Raniwala, K. Gopalan, and T. Chiueh, "Centralized Channel Assignment and Routing Algorithms for Multi-Channel Wireless Mesh Networks," *ACM Mobile Computing and Communications Review*, vol. 8, no. 2, pp. 50–65, April 2004.

[55] M. Alicherry, R. Bhatia, and E. Li, "Joint Channel Assignment and Routing for Throughput Optimization in Multiradio Wireless Mesh Networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 1960–1971, November 2006.

[56] M. Kodialam and T. Nandagopal, "Characterizing the Capacity Region in Multi-Radio Multi-Channel Wireless Mesh Networks," in *Proc. of ACM MobiCom*, 2005, pp. 73–87.

[57] X. Lin and S. Rasool, "A distributed joint channel-assignment, scheduling and routing algorithm for multi-channel ad-hoc wireless networks," in *Proc. of IEEE INFOCOM*. IEEE, May 2007, pp. 1118–1126.

[58] A. Mohsenian Rad and V. Wong, "Joint logical topology design, interface assignment, channel allocation, and routing for multi-channel wireless mesh networks," *IEEE Transactions on Wireless Communications*, vol. 6, no. 12, pp. 4432–4440, December 2007.

[59] J. Crichigno, M.-Y. Wu, and W. Shu, "Protocols and architectures for channel assignment in wireless mesh networks," *Ad Hoc Networks*, vol. 6, no. 7, pp. 1051–1077, 2008.

[60] Y. Wu, M. Keally, G. Zhou, and W. Mao, "Traffic-Aware Channel Assignment in Wireless Sensor Networks," in *Proc. of WASA*. Springer-Verlag, 2009, pp. 479–488.

[61] G. Zhou, L. Lu, S. Krishnamurthy, M. Keally, and Z. Ren, "SAS: Self-Adaptive Spectrum Management for Wireless Sensor Networks," in *Proc. of ICCCN*, August 2009, pp. 1–6.

[62] G. Zhou, T. He, J. Stankovic, and T. Abdelzaher, "RID: radio interference detection in wireless sensor networks," in *Proc. of IEEE INFOCOM*, vol. 2, march 2005, pp. 891–901.

[63] A. Franklin, A. Balachandran, C. Murthy, and M. Marina, "Demand based state aware channel reconfiguration algorithm for multi-channel multi-radio wireless mesh networks," in *Proc. of CARMEN*. IEEE, 2010, pp. 1 –6.

[64] J. Rezgui, A. Hafid, R. Ben Ali, and M. Gendreau, "Meta-heuristics for Channel (Re-)Assignment Problem for Multi-Radio Wireless Mesh Networks," in *Proc. of INFORMS Telecom*, May 2010.

[65] S. Avallone, F. D'Elia, and G. Ventre, "A traffic-aware channel re-assignment algorithm for wireless mesh networks," in *Proc. of European Wireless*. IEEE, April 2010, pp. 683–688.

[66] D. Eppstein, "Finding the k Shortest Paths," *SIAM Journal on Computing*, vol. 28, no. 2, pp. 652–673, April 1999.

[67] C. Spearman, "The Proof and Measurement of Association between Two Things," *The American Journal of Psychology*, vol. 15, no. 1, pp. 72–101, January 1904.

[68] Y.-C. Cheng, "CRAWDAD data set ucsd/cse (v. 2008-08-25)," Downloaded from http://www.crawdad.org/download/ucsd/cse/wireless, August 2008.

[69] N. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. Ramakrishnan, and J. van der Merwe, "A flexible model for resource management in virtual private network," in *Proc. of ACM SIGCOMM*, August 1999, pp. 43–57.

[70] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "High-Throughput Path Metric for Multi-Hop Wireless Routing," in *Proc. of ACM MobiCom*, September 2003, pp. 134–146.

[71] R. Draves, J. Padhye, and B. Zill, "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks," in *Proc. of ACM MobiCom*, September 2004, pp. 114–128.

[72] Y. Yang, J. Wang, and R. Kravets, "Designing Routing Metrics for Mesh Networks," in *Proc. of IEEE WiMesh*, 2005, p. September.

[73] A. Subramanian, M. Buddhikot, and S. Miller, "Interference Aware Routing in Multi-Radio Wireless Mesh Networks," in *Proc. of IEEE WiMesh*, September 2006, pp. 55–63.

[74] C. Koksal and H. Balakrishnan, "Quality-Aware Routing Metrics for Time-Varying Wireless Mesh Networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 1984–1994, November 2006.

[75] L. Ma and M. Denko, "A Routing Metric for Load-Balancing in Wireless Mesh Networks," in *Proc. of AINA*, May 2007.

[76] D. Shila and T. Anjali, "Load-aware Traffic Engineering for Mesh Networks ," in *Proc. of WiMAN*, August 2007.

[77] 802.11 Working Group of the IEEE 802 Committee, *IEEE P802.11s/D5.0 – Draft STANDARD for Information Technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications – Amendment 10: Mesh Networking*, April 2010.

[78] K.W.Choi, W. Jeon, and D. Jeong, "Efficient Load-Awarw Routing Scheme for Wireless Mesh Networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 9, pp. 1293–1307, September 2010.

[79] S. Biswas and R. Morris, "ExOR: Opportunistic Multi-Hop Routing for Wireless Networks," in *Proc. of ACM SIGCOMM*, August 2005, pp. 133–143.

[80] Y. Yuan, H. Yang, S. Wong, S. Lu, and W. Arbaugh, "ROMER: Resilient Opportunistic Mesh Routing for Wireless Mesh Networks," in *Proc. of IEEE WiMesh*, September 2005.

[81] M. Marina and S. Das, "On-demand Multi-path Distance Vector Routing in ad-hoc Networks," in *Proc. of IEEE ICNP*, November 2001.

[82] C. Chang and W. Liao, "On Multipath Routing in Wireless Mesh Networks with Multiple Gateways," in *Proc. of IEEE Globecom*, December 2010.

[83] A. Acharya, S. Ganu, and A. Misra, "DCMA: A Label Switching MAC for Efficient Packet Forwarding in MultiHop Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 1995–2004, November 2006.

[84] D. Raguin, M. Kubisch, H. Karl, and A. Wolisz, "Queue-driven cut-through medium access in wireless ad hoc networks," in *Proc. of IEEE WCNC*, March 2004, pp. 1909–1914.

[85] M. Jabeen and S. Khan, "âLabel Switch Path with guaranteed Quality of Service in Mobile Ad Hoc Network," in *Proc. of ICWMC*, July 2006.

[86] R. Garroppo, S. Giordano, and L. Tavanti, "Network-Based Micro-Mobility in Wireless Mesh Networks: Is MPLS Convenient?" in *Proc. of IEEE Globecom*, December 2009, pp. 1–5.

[87] A. Wächter and L. T. Biegler, "On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[88] Y. Yen, "Finding the K Shortest Loopless Paths in a Network," *Management Science*, vol. 17, no. 11, pp. 712–716, July 1971.

[89] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms (2nd ed.* MIT Press, 2001.

[90] E. Martins and M. Pascoal, "A New Implementation of Yen's Ranking Loopless Paths Algorithm," in *Proc. of Optimization*, July 2001.

[91] R. Raghavendra, A. P. Jardosh, E. M. Belding, and H. Zheng, "Ipac: Ip-based adaptive packet concatenation for multihop wireless networks," in *Proc. Fortieth Asilomar Conference on Signals, Systems and Computers ACSSC '06*, 2006, pp. 2147–2153.

[92] J. Karlsson, A. Kassler, and A. Brunstrom, "Impact of packet aggregation on tcp performance in wireless mesh networks." in *WOWMOM*. IEEE, 2009, pp. 1–7.

[93] P. Dely, A. Kassler, N. Bayer, and D. Sivchenko, "An experimental comparison of burst packet transmission schemes in ieee 802.11-based wireless mesh networks," in *GLOBECOM*, vol. 2010, 2010, pp. 1–5.

[94] IEEE, "IEEE standard 802.11n," *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, pp. 1–502, 2009.

[95] M. G. M. N. D. G. Ashish Jain, "Benefits of packet aggregation in ad-hoc wireless network," Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, CU-CS-960-03, August 2003. [Online]. Available: http://www.cs.colorado.edu/department/publications/reports/docs/CU-CS-960-03.pdf

[96] M. Castro, P. Dely, J. Karlsson, and A. Kassler, "Capacity increase for voice over ip traffic through packet aggregation in wireless multihop mesh networks," *Future Generation Communication and Networking*, vol. 2, 2007.

[97] R. Riggio, D. Miorandi, F. De Pellegrini, F. Granelli, and I. Chlamtac, "A traffic aggregation and differentiation scheme for enhanced qos in ieee 802.11-based wireless mesh networks," *Computer communications*, vol. 31, no. 7, pp. 1290–1300, 2008.

[98] P. Dely, A. Kassler, N. Bayer, H. Einsiedler, and D. Sivchenko, "FUZPAG: A Fuzzy-Controlled Packet Aggregation Scheme for Wireless Mesh Networks," in *Proc. 7th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'10), Yantai, China*, 2010.

[99] *IEEE Standard 802.11, IEEE standard for wireless LAN medium access control (MAC) and physical layer (PHY) specifications*, IEEE Std., 1999.

[100] S. Floyd, T. Henderson, and A. Gurtov, "Rfc 3782-the newreno modification to tcp's fast recovery algorithm," IETF, 2004.

[101] E. Blanton, M. Allman, K. Fall, and L. Wang, "Rfc 3517: A conservative selective acknowledgment (sack)-based loss recovery algorithm for tcp," IETF, 2003.

[102] W. John and S. Tafvelin, "Analysis of internet backbone traffic and header anomalies observed," in *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2007, pp. 111–116.

[103] C. Cicconetti, E. Mingozzi, and G. Stea, "An integrated framework for enabling effective data collection and statistical analysis with ns-2," in *WNS2*

’06: Proceeding from the 2006 workshop on ns-2: the IP network simulator. New York, NY, USA: ACM, 2006, p. 11.

[104] D. Kliazovich and F. Granelli, “Packet concatenation at the IP level for performance enhancement in wireless local area networks,” *Wirel. Netw.*, vol. 14, no. 4, pp. 519–529, 2008.

[105] D. Skordoulis, Q. Ni, U. Ali, and M. Hadjinicolaou, “Analysis of concatenation and packing mechanisms in ieee 802.11 n,” in *Proceedings of the 6th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNET’07)*, 2007.

[106] K. Leung, V. Li, and D. Yang, “An overview of packet reordering in transmission control protocol (TCP): problems, solutions, and challenges,” *IEEE Transactions on Parallel and Distributed Systems*, pp. 522–535, 2007.

[107] H. Lim, K. Xu, and M. Gerla, “TCP performance over multipath routing in mobile ad hoc networks,” in *Communications, 2003. ICC’03. IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 1064–1068.