UNIVERSITY OF THESSALY
GREECE

Resource Allocation in Wireless Ad hoc Networks

Doctor of Philosophy
in
Electrical & Computer Engineering
by

Kostas Choumas

September 2015

The Dissertation of Kostas Choumas is approved by:

_____

_____

_____

Committee Chairperson

Electrical & Computer Engineering,
University of Thessaly, Greece

ACKNOWLEDGMENTS

The research work made during the course of my Ph.D has taken place at the Department of Electrical and Computer Engineering, University of Thessaly, Greece.

I would like to thank my advisors Prof. Leandros Tassiulas and Assistant Prof. Thanasis Korakis, for their guidance, motivation and provisioning of the funding required for my research, since my graduate thesis and during the course of my Ph.D. Their work has been a true inspiration and working together keeps me excited after all these years later. I would also like to express my thanks and appreciation to the members of my dissertation committee, Assistant Prof. Antonios Argyriou and Prof. Catherine Housti, for all their support and insight.

I would also like to thank my lab-mates, and everyone who have aided my research growth the past six years. Besides their help, they made my staying in University of Thessaly much more pleasurable.

Finally, I would like to dedicate this thesis to my parents, my sister and my wife for their love and support all these years. They mean the most to me and is their continuous support that made the accomplishment of this thesis possible.

Dedicated to my family

ABSTRACT OF THE DISSERTATION

Resource Allocation in Wireless Ad hoc Networks

by

Kostas Choumas

Doctor of Philosophy, Post Graduate Program in Electrical and Computer Engineering,
University of Thessaly, Greece.
September 2015

Although most of the daily scenarios of using wireless networking are clearly based on a type of infrastructure, such as an access point or a base station, there is a lot of interest for the wireless ad hoc networks. There is a bouquet of scenarios that require wireless connectivity and could not be supported without the wireless ad hoc networking. However, the efficient operation of the ad hoc networks is closely related to the problem of the efficient resource allocation. Resource allocation in the wireless ad hoc networks is a multi-challenging problem, with various open issues related to the routing, scheduling and rate adaptation subproblems. In this dissertation, we propose novel schemes that cope with these challenges, utilizing practices and ideas that originate from the Backpressure and Opportunistic Routing algorithms. We either mathematically prove that our schemes succeed an optimized performance, or we adopt heuristic approaches that are inspired from the aforementioned algorithms. The evaluation of the proposed schemes is based on experimentation using large scale wireless testbeds and extended software tools that adapt to our requirements. The capability for experimentation on these wireless testbeds is provided due to our work on extending the state-of-the-art testbed framework. Our evaluation results are very promising, since we compare our algorithms with other well known schemes and we showcase significant improvements comparing to them.

ΠΕΡΙΛΗΨΗ ΤΗΣ ΔΙΔΑΚΤΟΡΙΚΗΣ ΔΙΑΤΡΙΒΗΣ

Ανάθεση Πόρων σε Ασύρματα Αδόμητα Δίκτυα

από τον

Κωνσταντίνο Χούμα

Διδάκτορας του Μεταπτυχιακού Προγράμματος
του τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών.
Πανεπιστήμιο Θεσσαλίας, Ελλάδα
Σεπτέμβριος 2015

Παρόλο που τα καθημερινά σενάρια χρήσης της ασύρματης δικτύωσης βασίζονται συνήθως σε κάποια μορφή υποδομής, όπως είναι τα σημεία πρόσβασης (access points) ή οι σταθμοί βάσης (base stations), υπάρχει εξίσου μεγάλο ενδιαφέρον και για τα ασύρματα αδόμητα δίκτυα. Υπάρχει μία πληθώρα σεναρίων που χρειάζονται ασύρματη συνδεσιμότητα και δεν μπορούν να υποστηριχτούν χωρίς την ασύρματη αδόμητη δικτύωση. Ωστόσο, η αποδοτική χρήση των ασύρματων αδόμητων δικτύων είναι στενά συνδεδεμένη με το πρόβλημα της αποδοτικής ανάθεσης πόρων. Η ανάθεση πόρων στα ασύρματα αδόμητα δίκτυα είναι ένα πρόβλημα με πολλές προκλήσεις, με πολλαπλά υπό διερεύνηση θέματα που σχετίζονται με την δρομολόγηση πακέτων, τον χρονικό καθορισμό των μεταδόσεων όπως και την προσαρμογή των ρυθμών εισαγωγής πακέτων μέσα στο δίκτυο. Σε αυτή την διατριβή, προτείνουμε καινοτόμα σχήματα που απαντούν σε αυτές τις προκλήσεις, χρησιμοποιώντας πρακτικές και ιδέες που προκύπτουν από τον Backpressure αλγόριθμο και την Opportunistic routing λογική. Εναλλακτικά, είτε αποδεικνύουμε μαθηματικά ότι τα προτεινόμενα σχήματα έχουν βέλτιστη συμπεριφορά, είτε υιοθετούμε ευριστικές προσεγγίσεις. Η αξιολόγηση τους βασίζεται σε πειραματισμό σε μεγάλης κλίμακας ασύρματες πειραματικές υποδομές και διευρυμένα εργαλεία λογισμικού που καλύπτουν τις δικές μας πειραματικές ανάγκες. Η δυνατότητα για ένα τέτοιο πειραματισμό προκύπτει από την δουλειά μας για επέκταση ενός αναγνωρισμένου λογισμικού για διαχείριση τέτοιων υποδομών. Τα αποτελέσματα της αξιολόγησης μας είναι πολλαπλώς υποσχόμενα, καθότι συγκρίνουμε τους αλγόριθμους μας με άλλους αναγνωρισμένους αλγόριθμους και δείχνουμε σημαντικές βελτιώσεις σε σχέση με τους δεύτερους.

Contents

List of Figures

List of Tables

List of Abbreviations

AIFS	.....	Arbitration Inter-Frame Spacing
BP	.....	Backpressure
BCP	.....	Backpressure Collection Protocol
BRCP	.....	Backpressure-based Rate Control Protocol
CDP	.....	Congestion Diversity Protocol
CSMA/CA	.....	Carrier Sense Multiple Access with Collision Avoidance
DPRC	.....	Dynamic Routing and Power Control
EBoW	.....	Enhanced Backpressure over WiFi
ETT	.....	Expected Transmission Time
ETX	.....	Expected Transmission Count
HD	.....	High Definition
MMT	.....	Maximum Multicast Throughput
MMU	.....	Maximum Multicast Utility
MMU-W	.....	Maximum Multicast Utility - Wireless
MORE	.....	MAC-independent Opportunistic Routing and Encoding
NUM	.....	Network Utility Maximization
OR	.....	Opportunistic Routing
OSI	.....	Open Systems Interconnection
PLC	.....	Priority Linear Coding
PSNR	.....	Peak Signal-to-Noise Ratio
RLC	.....	Random Linear Coding
RHS	.....	Right-Hand Side
SDN	.....	Software Defined Networking
TCP	.....	Transport Control Protocol
TDMA	.....	Time Division Medium Access
MAC	.....	Medium Access Control
NC	.....	Network Coding
SRCR	.....	Source Routing
WiFi	.....	Wireless Fidelity
UDP	.....	User Datagram Protocol
ViMOR	.....	Video-aware Multicast Opportunistic Routing
qHD	.....	quarter High Definition

Chapter 1

Introduction

1.1 Motivation

AFTER its first appearance, WiFi or IEEE 802.11 is still today one of the dominant protocols for wireless connectivity. The proliferation of smart devices, like smart phones and tablets, was the cornerstone for the design and development of a bouquet of wireless protocols and architectures. In addition, the diversity of the communication services created the fertile ground for their manifold evolution. However, even if there are plenty new wireless technologies, such as LTE, WiMAX, etc., WiFi remains the most widely used one.

Although most of the daily scenarios of using wireless networking are clearly based on a type of infrastructure, such as an access point or a base station, there is a lot of research interest for the ad hoc networks. The ad hoc networks do not rely on an infrastructure, making the network orchestration a challenging problem from multiple perspectives. The most well know problem of these networks is their scaling capability, since they are not able to support as many devices as the networks with infrastructure do. However, there are multiple and valuable applications that rely on ad hoc networks, such as e.g. the army communication service in an invaded area, the sensors connectivity in a valley or building, the intracity WiFi meshes and many others. For all these reasons, during this dissertation we focus on the attractive and challenging aspect of the ad hoc wireless networking, proposing solutions for the routing, scheduling and rate adaptation problems. More specifically, we answer to questions related to problems like

- **Routing:** At which device should be forwarded a packet?
- **Scheduling:** When should each device transmit?
- **Rate adaptation:** At which rate should be forwarded the packets?

In order to answer sufficiently these questions, we focus on the application of the well known Backpressure (BP) scheme to the WiFi ad hoc networks. We elaborate the BP algorithm, as well as its improved version named Enhanced-BP, in order to design a compliant version for the WiFi networks. BP is an efficient scheme that has been aptly labeled as *routing without routes*. The most challenging

BP requirements are the centralized scheduler (scheduling decisions are taken from a central point that gathers information for the whole network) and the slotted time (the devices transmit during fixed periods called *slots* in a TDMA manner). It has been proven that BP succeeds in maximizing the throughput of the network flows (the rate of successful packet delivery to the destinations of the flows) keeping the network stable (limited buffers for the packets waiting to be transmitted), whenever this is feasible. The network flows (named also as commodities) are the streams serviced by a network and are identified by a set of source devices generating the flow packets and a set of destinations that deliver these packets. The BP scheduling and routing scheme is completely different from the widely used routing approaches, that decouple these two processes by using the scheduling of the CSMA/CA mechanism of WiFi and focusing on the retrieval of the best route from the source to the destination. BP enables the routing without routes, achieving an efficient throughput performance for all network flows, even when it is combined with a decentralized scheduling algorithm, as it is shown in [1].

However, [1] is satisfactory in cases where the rate of the injected packets of each flow is under threshold. When these rate vectors are located outside the *network capacity region*, that is the set of the throughput vectors could be delivered on this network, then a rate adaptation policy should be adopted. In [2], we propose a variety of rate adaptation policies on networks with multicast (or unicast) flows, using fixed routing trees that connect the source with multiple destinations. A source based congestion control would adapt the flow rate at the source device, serving the same rate to all destinations. Thus, in case that the quality of the wireless links connecting the destinations is not the same, then the source adapts to the rate of the worst link, deteriorating also the quality of the service delivered to the other end devices. In terms of video streaming, this would be translated to a state where all destinations perceive the same video of e.g. qHD resolution, while some of them could be able to receive HD video quality.

More specifically, we propose novel BP inspired schemes with predefined routing trees for multicast flows, that use virtual slots for the scheduling and combine a dropping mechanism for the rate adaptation. We replace the source based congestion control with a receiver based one, that using appropriately a dropping mechanism in each device, relaxes the source from forwarding the flow in its minimum rate. The receiver is able now to indicate backwards the state of its connection quality, triggering accordingly the extra packet drop inside the network, without deteriorating the perceived video rate at the other receivers. Under these new proposed schemes, the source forwards the flow using the maximum available rate, while the network efficiently reduces the forwarded rate by dropping packets intermediately, delivering to each destination the higher feasible quality according to its end connection capabilities. So, in case of a scalable encoded video[1] streamed to multiple destinations, the source transmits all video layers, while some of the packets including upper layers are dropped inside the network without reaching all destinations.

The energy efficient resource allocation is the other challenging issue related to the aforementioned proposed algorithms. The multi-objective purpose of an ad hoc network is to fulfill successfully

[1] Scalable encoded video is divided to layers and its decoding does not require the reception of all layers, however, in a cost of the perceived video quality.

the requirements for efficient routing, scheduling and rate adaptation, however, in the most energy efficient manner. In case of an ad hoc sensor network is obvious that the network lifetime is proportional to the energy consumption of the sensors. In this dissertation, we also present an enhancement of the schemes in [2] by elaborating the dropping mechanism, in a way that the packet drops happen as soon as possible in the routing tree. We start with a study of a network with unicast flows using predefined routing paths, where we detect the energy inefficiency of the previous schemes, since they do not enforce the extra packets to be dropped at the source. Obviously, in the case of unicast flows with single paths, there is no reason to forward packets across the paths if they have to be dropped at the end. By resolving this issue, the energy consumption is reduced without affecting the throughput performance of the flows. Then, we extend this work for the support of multicast as well as multipath routing for unicast flows.

A completely different approach is presented later, where we investigate the potential of the Opportunistic Routing (OR) approach. OR is exploiting the inherent broadcast nature of the wireless medium and adapts very well in the lossy wireless environment. It creates cooperative diversity, leveraging the fact that the wireless transmissions are overheard in order to send information through multiple relays concurrently. In [3], we present an innovative OR scheme for video-aware multicast streaming in ad hoc two-hop networks. Although there are multiple other schemes that exploit OR for streaming over wireless networks, in this work we propose a novel scheme for scalable encoded video forwarding. We utilize Network Coding (NC), combined with a virtual slot mechanism and a packet delivery without acknowledgments, which enables enhanced scheduling and rate adaptation practices.

At the end, we showcase how the evaluation of all these schemes is possible through experimentation on large scale platforms, controlled and managed by an enhanced testbed framework built by us. We implement all the above protocols and conduct experiments on multiple facilities, such as the NITOS one [6]. This would not be feasible, if we had not extended a state-of-the-art testbed framework, enabling also the experimentation with Software Defined Networking (SDN) components, such as the Click Modular router [7]. In [4], the authors outline their efforts to build the appropriate extensions for the Click support and the SDN experimentation.

1.2 Thesis Outline

In this dissertation, we concentrate on the resource allocation problem on the wireless ad hoc networks, dealing with multiple challenges, such as the routing, scheduling and rate adaptation issues. We design and develop novel schemes that deal with these challenges, evaluating them on large scale ad hoc networks provisioned by wireless testbeds. Below, we enlist the included chapters below, as well as a summary of their content.

In **Chapter 2** we present the work included in [1], that describes and evaluates a routing and sheduling scheme for multihop WiFi ad hoc networks, inspired by the BP algorithm. This scheme, named EBoW lacks of a rate adaptation policy, that will be proposed in the following chapter.

In **Chapter 3** we present the work in [2], that deals with the rate adaptation challenge on the same multihop ad hoc networks, using the BP scheduling and enabling the multirate multicast (or unicast) streaming. The criterion for the resource allocation depend on the network objective for maximization of the sum throughput or users utilization, concluding to the MMT or MMU policies respectively.

In **Chapter 4** we present the enhancement of the work done in the previous chapter, focusing on the minimization of the energy consumption in an ad hoc network that operates under the previous schemes. The new scheme is named E-DLS.

In **Chapter 5** we present the work done in [3], a novel scheme named ViMOR, that is a completely different approach based on OR and NC, supporting video-aware multicast streaming over two-hop ad hoc networks.

Finally, in **Chapter 6** we present our contribution in extending the state-of-the-art framework for testbed control and management, in order to support the experimentation required for the evaluation of the previous schemes, as it is also described in [4].

Chapter 2

An 802.11 compliant version of the Enhanced-Backpressure algorithm

EXTENSIVE work has been done in wireless multihop routing with several ideas based on shortest path or load balancing routing algorithms, that aim at minimizing end-to-end delay or maximizing throughput respectively. Backpressure (BP) is a throughput-optimal scheme for multihop routing and scheduling, while Enhanced-BP is an incremental work that reduces end-to-end delay without sacrificing throughput optimality. However, the implementation of both theoretical schemes is not straightforward in the presence of 802.11 MAC, mainly because of their requirement for centralized scheduling decisions that is not aligned with the aspects of CSMA/CA.

This chapter presents a novel scheme, named Enhanced-BP over WiFi (EBoW), which is compatible with the decentralized operation of WiFi networks and efficiently utilizes the benefits of Enhanced-Backpressure design, combining throughput optimality with low end-to-end delay. EBoW router is implemented relying on Click framework for routing configuration. The performance of EBoW is evaluated both on a medium-scale outdoors wireless testbed as well as through experimentations in NS-3 simulator tool. The protocol has been compared against other state of the art routing protocols and we argue that EBoW is much more throughput efficient than the others, while succeeding similar end-to-end delay.

2.1 Introduction

As the demand for seamless connection increases, the use of wireless multihop networks as a communication infrastructure becomes more and more popular. The efficiency of a multihop mesh network is directly related to the routing protocol. On the one hand, shortest path routing algorithms achieve minimum end-to-end delay. On the other hand, when efficiency is measured in terms of throughput, the routing protocol should utilize multiple paths connecting a source-destination pair, avoiding the dominant shortest path approach.

A scheme that achieves throughput optimality is the well-known BP algorithm, which operates on

a time-slotted and centralized schedule and introduces scheduling and routing policies. Enhanced-BP is an improved mechanism that can be configured with an appropriate bias that inclines packets to move in the direction of their shortest paths. Enhanced-BP provides lower delay than BP, without sacrificing throughput efficiency. The seminal work of Tassiulas and Ephremides [8] constitutes the core of BP and Enhanced-BP, first introduced in the work of Neely *et al.* [9] as Dynamic Routing and Power Control (DPRC) and Enhanced-DPRC respectively. The work of Georgiadis *et al.* [10] offers a comprehensive survey.

Although both schemes are throughput optimal, they have not been implemented, mainly because of the centralized scheduling policy and the time-slotted assumption. These features do not fit with the dominant wireless communication protocols. In this chapter we propose an 802.11 compliant version of Enhanced-BP, named EBoW, which implements the Enhanced-BP aspects in a manner that is compatible with WiFi networks. The experimentation results of the proposed scheme show that Enhanced-BP principles can be efficiently applied to contemporary WiFi mesh networks.

The novelty of the proposed algorithm is twofold: i) it provides a distributed load balancing scheme, which connects nodes through multiple paths and optimizes throughput efficiency keeping low end-to-end delay, while ii) it is applicable to WiFi ad hoc meshes that support parallel flows, where a flow defines a stream of packets with specific source and destination. We prove that Enhanced-BP principles could be well adapted so as to be efficient even if a central scheduling mechanism cannot exist. Particularly, we introduce a scheme in which every node attempts to forward packets to less loaded and closer to the destination neighbors, in a similar way that Enhanced-BP scheme schedules. If there is no less loaded neighbor that is closer (or at least at the same distance) to the destination, then the node stops forwarding. This feature enables a simultaneously activated scheduling policy that offers more transmission opportunities to other neighboring nodes that experience collisions.

In order to evaluate the proposed routing and scheduling scheme in realistic conditions, we implement it using the Click Modular router [7]. By conducting experiments in a realistic wireless testbed, named NITOS [6], we compare our algorithm to other well-known schemes, and we show that it sometimes gains significant throughput increase due to load-balancing inherent characteristics. We also explore and verify our experimentation results in identical setups using NS-3 [11] simulator integrated with Click development.

The rest of the chapter is organized as follows. In Section 2.2 we introduce related work, while in Section 2.3 we describe BP and Enhanced-BP schemes in detail. Section 2.4 describes our proposed scheme and The numerical results are provided in Section 2.5, while Section 2.6 concludes the chapter.

2.2 Related Work

The main part of BP and Enhanced-BP [10] is the scheduling policy that requires a central node responsible for collecting information about the whole network. According to these schemes, each

packet is related to a particular *commodity*, which may be defined by its destination or its source-destination pair or something more specific. For the rest of the chapter, we assume that each commodity represents a particular destination, and there is a one-to-one correspondence between sets of commodities and destinations. Furthermore, each node maintains a set of internal network layer (layer 3 of OSI model) queues, while each queue corresponds to a commodity and stores all associated packets with this commodity. The length of a commodity queue is named as commodity *backlog*.

The scheduling policy of both schemes is actually a *maximum weight matching*[1] algorithm that chooses to transmit packets corresponding to particular commodities through specific links, so as to maximize the aggregate *link-commodity weight*. In contrast to traditional schemes where each link is assigned a single weight, BP and Enhanced-BP assign multiple weights to a link, corresponding to different commodities. BP utilizes a link-commodity weight that is linear to the difference among the commodity backlogs of the adjacent nodes of the corresponding link, which is denoted as *differential backlog*. In case of Enhanced-BP, the relative weight is also linear to the difference of the distances between the adjacent nodes of the link and the destination of the commodity, which similarly is denoted as *differential distance*.

Finally, the routing decision of each node is implied by the centralized scheduling policy, as each node transmits a packet associated with the commodity and through the link of the corresponding scheduling choice. Unlike traditional routing mechanisms for wired and wireless networks, BP and Enhanced-BP routing do not perform any explicit path computation from source to destination.

Several algorithms based on BP aspects have been proposed, which mainly require heuristic modifications of BP principles and sometimes introduction of new MAC protocols. XPRESS [12] is a well designed cross-layer architecture, which implements accurately most of the BP designs. Actually, it forces the wireless network to act like a hypothetical wireless switch, operating on a TDMA MAC, as it is originally proposed in theory. On the other hand, ad hoc wireless environments without centralized control are commonly used, and a TDMA MAC is not applicable in such environments.

Backpressure-based Rate Control Protocol (BRCP) [13] is an approach of BP scheduling over a predefined routing tree of sensors with a single commodity (or destination) related to the root of the tree. It is actually a distributed version, where each sensor decides the prioritization of its current transmission based on the current differential backlog. Obviously, this scheme does not apply on a wireless mesh with multiple commodities.

DiffQ [14] is a different scheduling approach for wireless mesh networks that supports multiple commodities. It enables packet transmissions of commodities through links with positive differential backlog and gives higher priority to those with larger differentials. This design requires changes of the 802.11 MAC layer. Furthermore, the routing decision depends on a shortest path routing protocol that does not feature load balancing characteristics and therefore does not take advantage of these BP inherent features.

The Backpressure Collection Protocol (BCP) [15] is a remarkable implementation and experi-

[1]Matching is a set of pairwise non-adjacent links and the maximum weight matching has the maximum aggregate weight.

mentation of BP design in wireless sensor networks over 802.15.4. In sensor networks there is only one commodity, since all packets have the same destination. So sensors include only one network layer queue, a feature that differentiates BCP from the BP scheme. Furthermore, BCP uses a BP related weight that is enriched with a penalty mechanism, while the use of an Enhanced-BP oriented weight seems to be more efficient.

Horizon [16] is another system design for distance-vector routing in wireless multihop networks, which is inspired by the BP principles and it is compatible with 802.11 MAC and TCP. It is the first system architecture that implemented these principles in a wireless system design. However, Horizon uses an inexplicably simplified forwarding algorithm that actually depends on backlogs and not on differential backlogs, as BP does. Congestion Diversity Protocol (CDP) [17] is another distance-vector routing protocol that is queue length aware and not BP inspired. In like manner, it does not depend on differential backlogs and moreover does not keep different queues for different commodities. Both protocols, Horizon and CDP, do not include scheduling policies. The proposed EBoW scheme is compared with these implementations.

In contrast, SouRCe Routing (SRCR) is a shortest path routing protocol implemented in Roofnet [18]. It uses Expected Transmission Time (ETT) [19], which is the state of the art for defining wireless link weight for forwarding packets, and applies Dijkstra algorithm to explore the shortest route from a source to a destination. Since several evaluation works show that SRCR efficiently routes packets in mesh networks, the proposed algorithm will be compared to SRCR to measure the efficiency of the load balancing feature that comes as inherent characteristic in BP.

Summarizing, the most remarkable implemented routing protocols for WiFi ad hoc meshes with multiple flows are these of Horizon, CDP and SRCR. Horizon and CDP are two notable distance-vector routing schemes that attempt to use multiple paths for packet forwarding using BP principles. Their common feature is their effort to avoid overloaded paths. On the other hand, SRCR uses always the shortest path ignoring alternative less loaded routes.

2.3 BP & Enhanced-BP Explained

Before starting the description we will introduce the main terminology of BP and Enhanced-BP. A throughput vector includes the throughputs of all network flows, where a flow throughput is the average packet delivery rate of this flow. Furthermore, the capacity region of a certain scheme consists of the throughput vectors that this scheme is able to manage, while the capacity region of a network is the union of the capacity regions of all possible schemes. Figure 2.1(a) illustrates the above mentioned regions for a network with two flows. Tassiulas and Ephremides [8] proved that under a slotted environment, the capacity region of a network is the same as the capacity region of the BP scheme, which is superset of the capacity region of any other scheme. Neely *et al.* [9] also showed that Enhanced-BP has the same capacity region with BP. The goal of the proposed scheme is to succeed an extended capacity region similar to that of the BP and Enhanced-BP schemes.

Before proceeding, we introduce also some notations. We consider a multihop wireless network

with node, link and commodity sets denoted as $\mathcal{N}, \mathcal{L}$ and $\mathcal{C}$ respectively. If $i, j \in \mathcal{N}$ are two adjacent nodes in the network, then $l = (i, j) \in \mathcal{L}$ is a directional link. Furthermore, if $c \in \mathcal{C}$ denotes a commodity, then $\delta_c \in \mathcal{N}$ is the destination node that corresponds to commodity c. Finally, as it is shown in Figure 2.1(b), $Q_i^{(c)}$ symbolizes backlog of commodity c at node i and $E_i^{(c)}$ stands for the length of the shortest path (or the distance) from node i to destination δ_c. Consequently, differential backlog and differential distance of commodity c through link (i, j) are denoted as $\Delta Q_{ij}^{(c)} = Q_i^{(c)} - Q_j^{(c)}$ and $\Delta E_{ij}^{(c)} = E_i^{(c)} - E_j^{(c)}$ respectively. In addition, R_{ij} is the actual data rate of link (i, j).

The core of BP and Enhanced-BP schemes is the maximum link-commodity weight matching algorithm. The BP link-commodity weight for a commodity $c \in \mathcal{C}$ and a link $(i, j) \in \mathcal{L}$ is either zero or the positive product of the actual data rate R_{ij} of the specific link and the differential backlog $\Delta Q_{ij}^{(c)}$, as it is given in the following equation

$$w_{ij}^{(c)} = \max\{\Delta Q_{ij}^{(c)} \cdot R_{ij}, 0\}.$$

Furthermore, Enhanced-BP can be configured so as to incline packets to move in the direction of their shortest paths, using another appropriately defined link-commodity weight. More specifically, in Enhanced-BP differential distance $\Delta E_{ij}^{(c)}$ is added to the above mentioned differential backlog, as it is presented in the following equation

$$\bar{w}_{ij}^{(c)} = \max\{(\Delta Q_{ij}^{(c)} + \Delta E_{ij}^{(c)}) \cdot R_{ij}, 0\}.$$

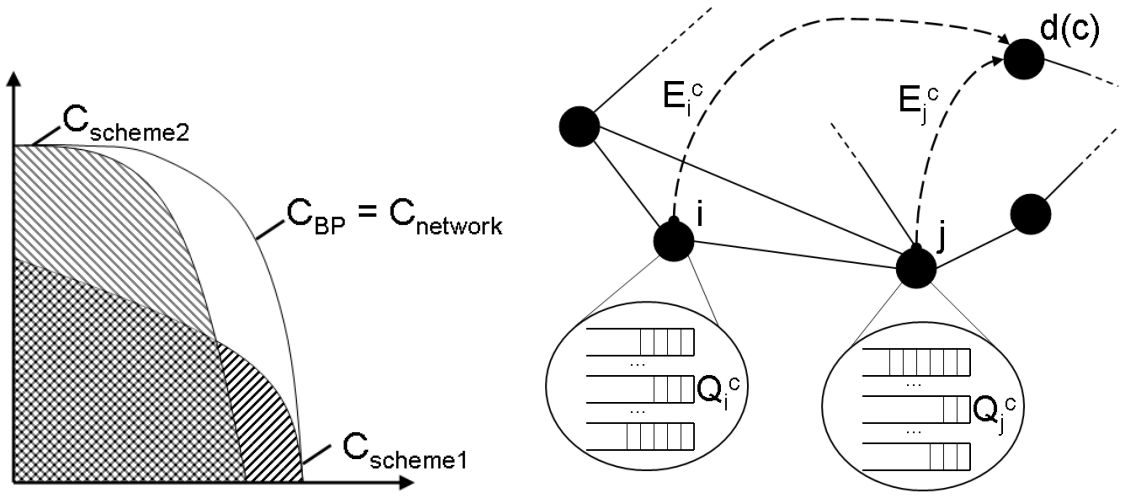
The distance should be estimated using hop count or another more sophisticated approach for measuring the shortest path length. Finally, the Enhanced-BP scheme succeeds the same throughput optimality as the BP scheme, while at the same time it reduces packet delivery time.

2.4 EBoW Design

2.4.1 Algorithm Description

The goal of the EBoW scheme is to improve throughput efficiency of wireless mesh networks, adopting the Enhanced-BP principles, while simultaneously succeeding low delay, similar to the shortest path routing schemes. Due to its characteristics, this scheme can dynamically avoid overloaded paths, create parallel routes and therefore balance the traffic load in the network, increasing in this way its throughput efficiency.

Below we describe the main principle behind EBoW explaining the practices adapted, in order to achieve the benefits mentioned. Consider a network with multiple nodes that operate in a multihop environment. In such a setup, EBoW runs in every node that can act as source, destination or relay. Every node maintains a set of internal network layer queues, where each queue corresponds to a commodity, as specified in the BP and Enhanced-BP schemes (further details in Section 2.2 and Section 2.3).



(a) Capacity region of a network with two flows

(b) Commodity backlogs and shortest distances of two adjacent nodes

Figure 2.1: Capacity region and Network snapshot.

When a node receives or generates a packet that needs to be forwarded, it identifies the related commodity, recognizing the destination of the packet, and pushes the packet to the corresponding network layer queue. Moreover, each node that has packets in its queues, initiates a procedure to schedule the transmission of a packet. The most important part of this procedure is the calculation of the link-commodity weights

$$\hat{w}_{ij}^{(c)} = (\Delta Q_{ij}^{(c)} + \Delta E_{ij}^{(c)}) \cdot R_{ij}. \tag{2.1}$$

Then, the node finds the link-commodity pair with the maximum weight, with positive differential backlog $\Delta Q_{ij}^{(c)} > 0$ and non-negative differential distance $\Delta E_{ij}^{(c)} \geq 0$. If there is at least one link-commodity pair that satisfies these conditions, the node selects the one with the maximum weight and transmits a packet from the corresponding queue. Then, the packet is passed down to the data-link layer (layer 2 of OSI model), while it is tagged to be transmitted through the link that is related with this pair. This algorithm is the routing policy of the EBoW scheme. On the other hand, in case there is no pair that meets these requirements, the node remains inactive and does not schedule transmissions. These requirements constitute the scheduling policy of the EBoW scheme.

The first scheduling condition requires a positive differential backlog $\Delta Q_{ij}^{(c)} > 0$ and exists because of the remark of Li *et al.* [20], who explained why the capacity of a chain of nodes is reduced due to the inherent characteristics of 802.11. Nodes centrally located in the chain experience more collisions than border nodes. This is due to the fact that border nodes inject more packets into the chain than the subsequent nodes can forward, so these packets are eventually dropped. The time that border nodes spend to send those extra packets decreases the overall throughput, since it prevents transmissions of subsequent nodes. Our proposed algorithm takes this observation into account and therefore introduces the positive differential backlog condition to prevent it. Therefore, in EBoW if a

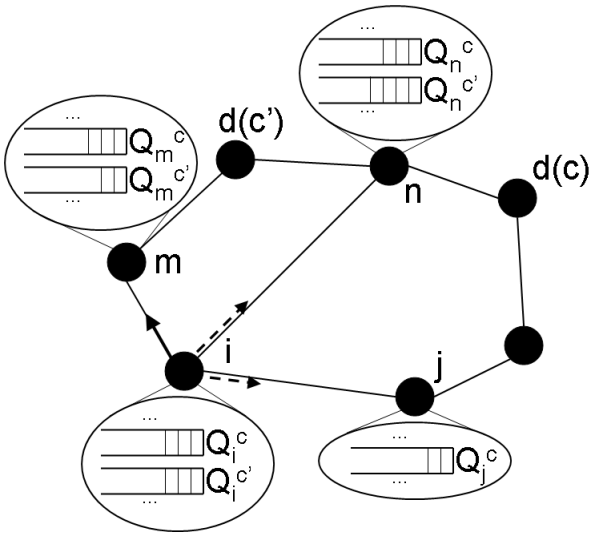


Figure 2.2: Network example illustrating EBoW design.

node observes that an adjacent node features a higher backlog than itself, it refrains from forwarding packets to it, thereby providing more transmission opportunities to this node. In this way, the capacity per hop is allocated more efficiently.

The second scheduling condition namely non-negative differential distance $\Delta E_{ij}^{(c)} \geq 0$ exists in order to avoid excessively long routes of packets towards their destinations. Eventhough the Enhanced-BP scheme takes into account distance to destination by incorporating it into the link weights, it often allows a packet to move further away from its destination. However, this is a typically undesired feature for traditional wireless applications, where users are interested in experiencing low end-to-end delay and jitter. This observation explains why the non-negative differential distance is a necessary condition for packet forwarding in our proposed scheme, which takes practical WiFi applications into consideration.

In essence, the main differentiation of EBoW with respect to the original Enhanced-BP scheme is that it is structured around a distributed architecture, which makes it more suitable for real WiFi applications. Enhanced-BP is a centralized algorithm which assigns non-negative weights to all link-commodity pairs and schedules packet transmissions based on the maximum weight matching principle. In contrast, in EBoW each node makes its forwarding decisions independently, according to the algorithm described before. More specifically, each node $i \in \mathcal{N}$ finds neighbor $j \in \mathcal{N} : (i,j) \in \mathcal{L}$ and commodity $c \in \mathcal{C}$ that maximize $\hat{w}_{ij}^{(c)}$ (ties broken arbitrarily) subject to $\Delta Q_{ij}^{(c)} > 0$ and $\Delta E_{ij}^{(c)} \geq 0$, and pushes down from the network to the data-link layer a packet that corresponds to commodity c for transmission through link (i,j). If there are no neighbor j and commodity c that satisfy these conditions, the node remains inactive.

For example, in Figure 2.2, considering that the path length is measured in hops and the data rate of each link is equal to unit, then $\hat{w}_{ij}^{(c)} = ((3-2)+(2-2)) \cdot 1 = 1$ and $\hat{w}_{i,m}^{(c')} = ((3-2)+(2-1)) \cdot 1 = 2$, while $\Delta Q_{i,n}^{(c)} = 3-3 = 0$, $\Delta E_{i,m}^{(c)} = 2-3 < 0$, $\Delta E_{ij}^{(c')} = 2-3 < 0$ and $\Delta Q_{i,n}^{(c')} = 3-4 < 0$. As a result, node i forwards to neighbor m a packet that corresponds to commodity c'.

2.4.2 Distance Calculation and Broadcast Packets

The way that the algorithm estimates the distance between two nodes is another important point to mention. In the previous example, we used hop count for simplicity. In fact, the proposed algorithm uses a more sophisticated technique of distance measurement. The distance $E_i^{(c)}$ between node i and destination δ_c is expressed as a scale factor θ of the aggregate expected transmission time of a packet (queuing and processing delay is ignored), which is routed through the shortest path from source i to destination δ_c. The scale factor is adjusted so that $\Delta E_{ij}^{(c)}$ and $\Delta Q_{ij}^{(c)}$ to have the same range of values. The expected transmission time through a link is given by the well known formula of the ETT metric [19]. According to this formula

$$ETT = \frac{1}{d_f \cdot d_r} \cdot \frac{S}{B}, \tag{2.2}$$

where d_f and d_r are the expected forward and reverse link delivery probabilities (the product of these two is the probability of a successful acknowledged transmission), S is the average packet size and B is the average packet rate that the rate controller assigns. So, if e_{ij} stands for the ETT weight of link (i, j), then

$$E_i^{(c)} = \theta \cdot \sum_{\forall (k,l) \in \mathcal{L} \text{ in the shortest path from } i \text{ to } \delta_c} e_{kl}. \tag{2.3}$$

Every node needs to know all commodity backlogs of its neighbors, as well as the aggregate ETT of the shortest paths from itself and its neighbors to every destination. The ETT calculation for every link, especially of the forward and reverse probabilities d_f and d_r, is based on a probing mechanism that periodically forces nodes to send broadcast packets and inform neighbors about the number of the broadcast packets they have received. Due to this mechanism, each node estimates the ETT weights of its outgoing links, while simultaneously learning about the commodity backlogs of its neighbors, which are included in the broadcast packets.

However, nodes need to know ETT weights of links that are located multi-hops away from them. So, before a source starts forwarding a packet, it initiates a broadcast query flooding the network in order to reach the destination of the packet. While the query packet is passing through network links, it is tagged with the path that it followed until that point. The destination receives these queries and replies through all paths that the query flood explored. Similarly, each reply packet is tagged with the ETT weights of all links that this packet passed through. In this way, once the source (or a relay) receives the reply packets, it learns about all necessary ETT weights in order to estimate the distances from itself and its neighbors to the destination.

2.5 Experimentation Results

In this section, we present evaluation experiments of the implemented scheme under various scenarios. For the purposes of the experimental evaluation, we used the realistic medium-scale NITOS

testbed. Experimentation with implemented mechanisms in realistic infrastructure, may lead to results that depend on varying traffic, interference conditions and topology settings that cannot be fully controlled. In order to arrive at solid results regarding the evaluation of our protocol, we decided to run multiple executions of each experimental scenario that is presented in this section. More specifically, each experiment is run 5 times and lasts 10 minutes. The reported results present average values. Moreover, we decided to run the same experiments under fully controlled settings in a simulation environment and for this purpose we used the NS-3 platform. Comparison between the results obtained through experimentation on the different platforms, enhances the validity of the followed evaluation approach.

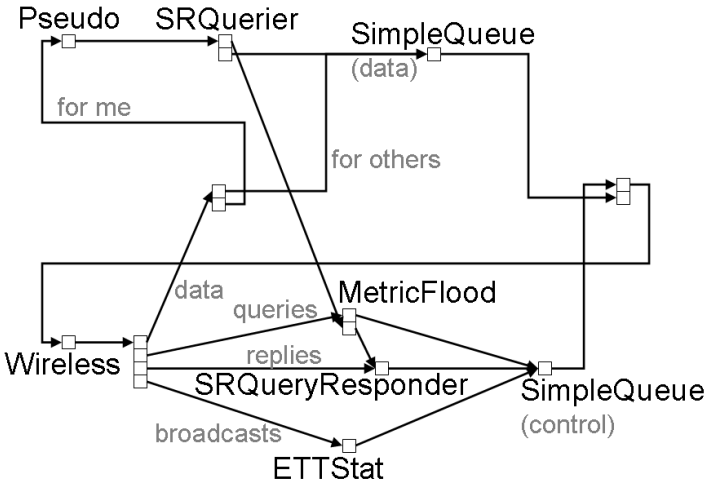
Through our experiments, we compare EBoW with SRCR, Horizon and CDP, which are considered as state of the art routing protocols for WiFi networks. For comparison reasons, we also implemented the Horizon and CDP mechanisms based on the Click framework and the SRCR Click configuration. Performance comparison between EBoW, SRCR, Horizon and CDP, is presented in terms of throughput and end-to-end delay. Our experiments are organized in two parts, where in the first part the execution of an experimental scenario in a simple proof-of-principle network is presented, while in the second part we conduct a complex experiment in a topology of 20 nodes that introduces randomness and approximates realistic conditions. Details about the implementation and the experimentation platforms in each case are presented below.

2.5.1 Implementation Framework

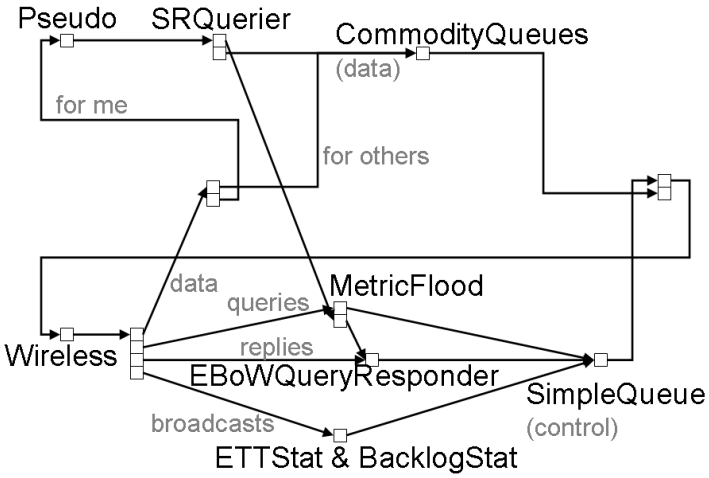
The software development of EBoW scheme is based on the Click framework, a novel software architecture for building flexible and configurable routers, that lie on network nodes and forward packets. A Click Modular router consists of packet processing modules called elements. Individual elements implement simple router functions like packet classification, queuing and interfacing with network devices. Complete router configurations are built by connecting elements into a graph, where packets flow along the graph's edges. Most of these elements are given by the existing framework, while it is possible to construct additional and more specific elements.

The Click Modular router is able to run as a linux-kernel module or a user-level executable on top of linux operated boxes. The Click framework includes a package of elements and a configuration of a user-level SRCR router (designed by the Roofnet [18] team). The configuration of the EBoW router shares a lot of common features with the corresponding one of the SRCR router, as it is illustrated in Figure 2.3. So, a brief description of the SRCR router follows, and then it is presented how the EBoW router is differentiated from this.

The user-level SRCR **Click Modular router** (see Figure 2.3(a)) uses a *pseudo*-interface that connects the application layer with the underlying network layer and a *wireless* interface that receives and transmits packets over the air. Once a packet is received from the application layer (through the *pseudo*-interface), it is forwarded to the *SRQuerier* element. The main goal of this element is to estimate the shortest path from this node to the final destination of the packet. If *SRQuerier* does not have already the information for this packet (through past investigation for the shortest path of another



(a) SRCR Router



(b) EBoW Router

Figure 2.3: Click Configurations.

packet with the same destination), it forwards the packet to the *MetricFlood* element. This element initiates a broadcast query flooding the network in order to reach the destination of the particular packet.

The query packets of the flood visit the *MetricFlood* elements of the relays until the destination. Once the destination is reached, the query packet is forwarded from *MetricFlood* to the *SRQueryResponder* element. *SRQueryResponder* initiates a reply packet that follows the shortest path to the initial node, visiting the corresponding element of each relay until the source. The shortest path is recognized by *SRQueryResponder* in the following way: the destination receives multiple query packets that were transmitted through the flooding method, through multiple paths. Each query packet is tagged with the path that it followed, as well as with the ETT weights of the intermediate links. So the destination explores the intermediate link weights and finds the shortest route applying Dijkstra algorithm.

Once the source receives the reply packet, it learns also about the shortest path to the destination, because the reply packet is similarly tagged with the path that it followed. So *SRQuerier* has the relative information for the following packets with the same destination. Therefore, the following packets are forwarded to the *SimpleQueue* element, which is able to store up to 50 packets, and implements the network layer queue. There is also another *SimpleQueue* element that stores the control packets, like query, reply and broadcast packets. The broadcast packets are generated and processed by the *ETTStat* element, for estimation of the ETT weights of the outgoing links, as it was described earlier in Subsection 2.4.2. Both queues store the packets, and forward them to the *wireless* interface on demand.

The user-level EBoW **Click Modular router** (see Figure 2.3(b)) uses the same underlying probing mechanism (*ETTStat*) that estimates the ETT weights of the network links, while it features an additional probing mechanism (*BacklogStat*) for broadcasting commodity backlogs of each node to its neighbors. Furthermore, as it was mentioned in Subsection 2.4.2, either the source or the relay of a packet needs to know the distances from itself and its neighbors to the destination, so as to be able to estimate the differential distances. So, *SRQueryResponder* is replaced by the new EBoW*QueryResponder* element, which responds with a broadcast reply flood, instead of sending a single reply packet through the shortest path.

Finally, the most important difference between EBoW and SRCR configurations, is the structure that stores the data packets. The EBoW router uses the *CommodityQueues* element instead of the *SimpleQueue* one. This structure includes a variety of internal network layer queues, one for each known commodity. Once a packet that corresponds to a new commodity arrives, the structure produces a new internal network layer queue. When a packet is to be forwarded from *CommodityQueues* to the *wireless* interface, the *CommodityQueues* element estimates the EBoW weights, pulls a packet from the appropriate internal queue and tags it for forwarding through the appropriate link. In case that there is no link-commodity pair that meets the previous mentioned requirements (see Section 2.4), then the element does not forward any packet to the interface.

In summary, the distinctive features of EBoW router are three:

- The extension of probing mechanism that is used also to propagate commodity backlog information.
- The modification of *SRQueryResponder* element in order to respond more than once through every path that queries followed until the destination.
- The replacement of *SimpleQueue* element with an appropriate *CommodityQueues* one, that includes dynamically added and removed network layer queues and applies the EBoW scheme.

2.5.2 Experimentation Platforms

NITOS provides multiple wireless nodes for easy setup of multi-hop routes. The main features of the nodes used for our experimentation, as well as their software specifications, are depicted in Table 2.1. In our experimentation, the nodes used 802.11a.

Table 2.1: Basic Configuration of NITOS nodes.

Model	Icarus nodes
CPU	Intel i7-2600 Proc., 8M Cache, at 3.40GHz
RAM	Kingston 4 GB HYPERX BLU DDR3
Storage	Solid State Drive 60GB
WiFi cards	two Atheros 802.11a/b/g/n (MIMO)
OS	3.2.0-31-generic Ubuntu precise
Driver	compat-wireless version 3.6.6-1-snpc

NS-3 is a valid simulation framework that is able to simulate network topologies under accurately controlled conditions. The experimenter has the flexibility to decide about the preferred OSI layer protocols. We exploited this feature to run our experiments using specific routing protocols. We decided to use the NS-3 simulator, because of its ability to be integrated with the Click framework, in order to be able to test the developed Click configurations in a simulation environment. As a result, we were able to use the same Click configuration in both real and simulated experiments.

2.5.3 Measurement Methodology

The traffic on the real testbed is generated using Iperf [21], a powerful tool for traffic generation and measurement. The experimental setup consists of several pairs of nodes that initiate UDP traffic flows by running Iperf clients at the sources and Iperf servers at the destinations. We also use the Iperf tool to collect throughput performance in each experiment. In order to monitor end-to-end delay performance per packet, we developed a custom timestamp mechanism using the Click framework. For experiments conducted in the NS-3 platform, we used the same mechanism to gather delay measurements, while traffic generation and throughput monitoring were performed using the *OnOffApplication* and *PacketSink* NS-3 application modules. Description of the conducted experiments and discussion about the obtained results follow.

2.5.4 Experiment I

The topology and connectivity map of our first experiment are illustrated in Figure 2.4. A ring topology of 7 nodes is designed, enabling a 3-hop and 4-hop path with the same source (node 1) and destination (node 4) nodes. Each network node is able to communicate through single-hop transmissions with each one of the two nodes that exist before and after it in the ring topology. In the presented topology, an extra node (node 8) also exists, which is able to communicate directly only with node 2.

The experiment consists of two parallel active flows, namely flow A and flow B, where through flow A node 1 transmits to node 4, and through flow B node 2 transmits to node 8. We conduct experiments of varying traffic rate for both of these flows, in order to estimate the capacity region supported by the network across the different protocols. Results obtained through experimentation in NITOS testbed and NS-3 simulator are very close, and for this reason we present in the following figures only the realistic testbed results. In Figure 2.5(a), we illustrate the capacity regions of the examined schemes. For example, the throughput vector $[6, 15]$Mbps that indicates throughputs 6Mbps

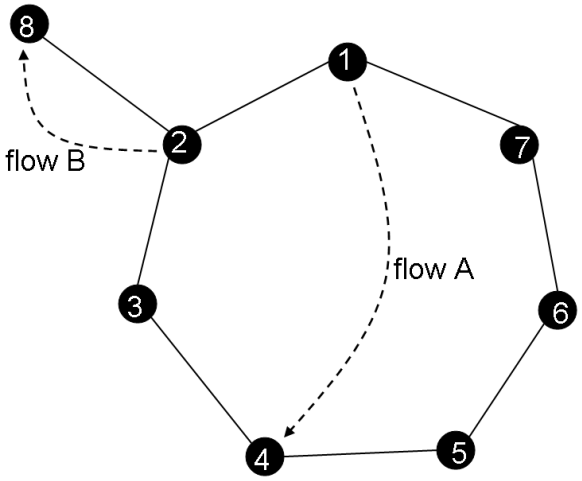


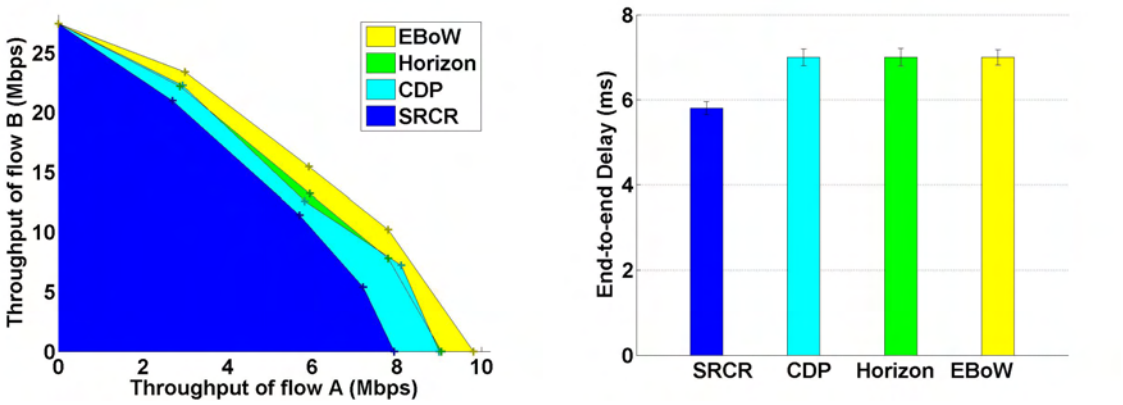
Figure 2.4: Topology of Experiment I.

and 15Mbps for the flows A and B respectively, exists inside the capacity region of EBoW, while it is outside the capacity region of the SRCR scheme.

SRCR as a shortest path routing algorithm indicates the shortest route for each flow. Due to the symmetric nature of the designed topology, nearly equal ETT weights are reported for each link. As a result, SRCR obviously selects the shortest 3-hop path 1-2-3-4 that features the lowest aggregate ETT weight, in comparison with the 4-hop path 1-7-6-5-4. As SRCR does not feature any load balancing mechanism, it is not able to detect simultaneous ongoing transmissions, as these of flow B in this scenario. Under the SRCR approach, node 2 acts as a bottleneck that significantly reduces the throughput for both flows.

On the other hand, the rest of the schemes under consideration, better exploit the existence of two paths between the source and destination nodes of flow A. Although the 4-hop path is longer in terms of ETT weight, it features less traffic load in comparison with the 3-hop path due to the simultaneous transmissions of node 2. As a result, the 4-hop path is able to provide higher throughput performance. In addition, we notice that EBoW approach features an extended capacity region compared to the regions of the other two distance vector routing protocols. The main reason for this phenomenon comes from the inherent scheduling policy of the EBoW scheme, through which nodes that feature non positive differential backlogs are scheduled not to forward packets, while relay nodes that follow are provided with more transmission opportunities (see more details in Section 2.4).

Another important factor that has to be considered is end-to-end delay performance. We notice that end-to-end delay reported for flow B is equal for all schemes, as they all use the same 1-hop route for packet forwarding. The end-to-end delay performance for flow A yielded in each approach, is depicted in Figure 2.5(b). As clearly shown in this figure, end-to-end delay performance for flow A is reported quite similar for all load balancing schemes. Another observation is that the SRCR scheme outperforms all the other schemes in terms of end-to-end delay. The lower end-to-end delay measurements reported for SRCR are expected, as this approach is based on the shortest path



(a) Capacity Region

(b) End-to-end Delay for flow A

Figure 2.5: Results of Experiment I.

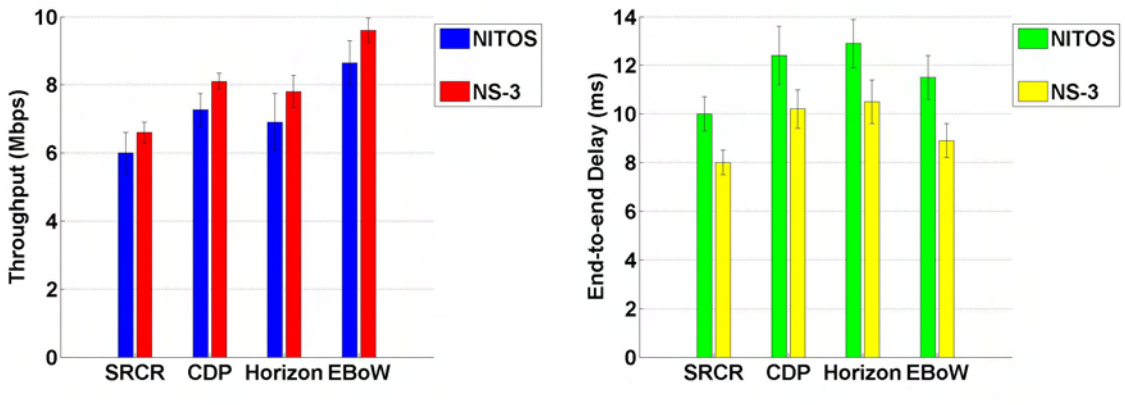
principle. However, end-to-end delay performance of SRCR is not significantly lower, which is due to the collisions that occur frequently in the heavy loaded 3-hop path and negatively affect delay performance.

2.5.5 Experiment II

In the second experiment we extend our validation in a random setup that includes 20 nodes and 3 randomly selected 4-hop flows. Figure 2.6(a) shows the average throughput achieved for the 3 flows under each approach, on top of the two different platforms. As we can see, the proposed scheme features significant throughput improvement compared with the other schemes, across both platforms. Under EBoW, transmitter nodes are able to utilize multiple paths and moreover refrain from forwarding packets to nodes that feature high backlogs and thus result in network capacity improvement. In addition, EBoW performs better even in cases where the source-destination node pair is connected through a unique path, as it provides more transmission opportunities to nodes that feature higher backlogs.

Figure 2.6(b) demonstrates end-to-end delay measurements obtained during experimentation in the extended topology, across the different environments. According to these results, we notice that SRCR provides the lowest delay values among the compared approaches. An observation of notable importance is that the EBoW scheme outperforms the rest load balancing schemes. This result is obtained under experimentation in more generic topologies compared with the simple ring topology used in the previous experiment. More complex topologies provide larger paths in terms of aggregate ETT, which are avoided by the proposed scheme based on the non-negative differential distance forwarding requirement, but not by the rest load balancing schemes.

Results obtained between the different platforms are quite close but have some characteristics that provide for further discussion. First of all, we notice that real testbed experiments yield lower throughput and higher delay performance in comparison with simulation results. This comes from



(a) Throughput (b) End-to-end Delay

Figure 2.6: Results of Experiment II.

the fact simulation environments are not able to accurately estimate performance of realistic networks. However, experimentation in each platform aids in arriving at relative conclusions, regarding the superiority of our protocol. At this point we also remark the higher deviation values observed during testbed experimentation, which result because of the volatile nature of the realistic testbed environment.

2.6 Chapter Conclusions & Future work

In this chapter we propose an implemented Enhanced-BP inspired scheme that exploits multi-path flow forwarding and outperforms the state of the art routing protocols in terms of throughput, while it keeps low packet delay close to the delay of the shortest path routing protocols. The new scheme features significant throughput benefits comparing to other routing protocols with load balancing efforts. We intent to extend the current work towards two directions: The first one is to compare the proposed protocol with centralized Enhanced-BP based implementation approaches, and to see the advantages and disadvantages of the distributed vs. the centralized version. The second one is to combine the proposed scheme with more sophisticated scheduling Enhanced-BP inspired policies that will be implemented in a distributed manner and will allow the scheduling of transmissions in the neighborhood based on the load difference of the contenting hops. The scheduling will be based on prioritization schemes of 802.11 such as those proposed in 802.11e (different AIFS or different back-off values based on traffic queues).

Chapter 3

Optimal algorithms for Multirate Multicast over wireless ad hoc networks

MULTIRATE multicast improves user quality but complicates network optimization. This chapter introduces a novel control scheme to dynamically optimize multirate multicast. We present Maximum Multicast Throughput (MMT), an adaptive policy which combines differential backlog scheduling and intelligent packet dropping, both based on local information. MMT is shown to maximize network throughput by adapting to changing conditions such as channel quality, network congestion, and device capabilities. Then, we study the problem of per-receiver network utility maximization. To maximize sum utility we propose the Maximum Multicast Utility (MMU) policy, an extension of MMT with receiver-end flow control. Under the operation of both policies backlog sizes are deterministically bounded, which provides delay guarantees on delivered packets. An important feature of the proposed scheme is that it does not require source cooperation or centralized calculations. To illustrate its practicality, we present a prototype implementation in the NITOS wireless testbed. Experimental results verify the optimality of the scheme and its low complexity.

3.1 Introduction

The increasing demand for multimedia applications, such as real-time conferencing, multiview video and video streaming, pushes data networks to their operational limits and motivates efficient resource allocation schemes. Multicast is a candidate method for delivering multimedia streams to multiple users across a network. To optimize individual user experience, it is desired to employ *multi-rate multicast* transmissions and use layered multimedia coding schemes to adapt users' perceived quality to allowable data rates, see [22, 23]. Since different receivers may require different data rates, we study the problem of *per-receiver Network Utility Maximization* (NUM) in multi-rate multicast, where each receiver is assigned a potentially different utility function.

Controlling multicast streams is challenging; the optimal network resource allocation and stream rate adaptation depends on the network state, which includes channel quality, network congestion,

user demand and device capabilities. Current approaches collect network state information at the source and calculate the achievable stream rate per receiver. Such a process can be overwhelming for the source, since a multicast stream may have thousands of receivers. In this work we develop a solution to per-receiver NUM in multi-rate multicast without source cooperation.

Our solution combines scheduling with intelligent packet dropping at intermediate nodes. Packets corresponding to all stream layers are initially injected into the network without any calculations. Progressively, some packets are dropped according to a dropping scheme which bases its decisions on local information. We show that the original stream is stripped of unnecessary packets so that each receiver obtains the exact amount of information that corresponds to maximum throughput. Moreover, we combine the above mechanism with receiver-end flow control to produce a scheme that maximizes utility without source cooperation.

The proposed policies have the following attractive features. *First*, they dynamically track the optimal solution without explicitly exchanging information about time-varying system parameters such as stream rate, link quality and network congestion. *Second*, they are based on neighbor backlog information which is found locally. Thus the policies are amenable to distributed implementation for wireless and heterogeneous network technologies. *Third*, they do not require source cooperation, i.e. the sources transmit stream packets without calculating the achievable receiver rates–this simplifies multi-rate multicast for networks with a large number of receivers. *Last*, they yield deterministic bounds for the queue backlogs, which provides delay guarantees and facilitates implementation on systems with finite buffers. Our contribution is summarized in the following points:

- We present the MMT policy, which achieves near optimal throughput for multi-rate multicast. MMT uses BP-type scheduling and threshold-based packet dropping.
- We develop the MMU policy, which additionally includes a utility-based flow controller at the receivers. MMU is shown to solve the per-receiver NUM problem.
- We propose MMU-Wireless (MMU-W), a heuristic modification for operation on IEEE 802.11-based wireless devices. We implement MMU-W in a wireless testbed and perform experiments. The results demonstrate the optimality and the low complexity of our scheme.

3.1.1 Related Work

The problem of NUM has been extensively studied for the case of unicast sessions [24]. For multicast sessions, [25] provides a fair-utility offline solution. Optimizing resource allocation and maximizing network utility by solving offline optimization problems is less desirable since variability in the network renders this approach ineffective. Every change requires re-solving the problem and enforcing new rules. A preferable alternative is a dynamic policy that achieves the long-term goals by making adaptive real-time decisions [26].

In [27], an approach to dynamic control for multicast sessions balances the stream across a selection of multicast trees. While this approach provides maximum throughput, it requires the arrivals

to be feasible, which can only be achieved with proper flow control. Many existing flow control approaches have the users estimate their maximally allowable throughput and convey this information to the source, which then creates a virtual multicast session for each stream layer [28, 29]. Recently, [30] proposed a dynamic policy that solves the stochastic NUM problem under the assumption of infinite demand, using the concept of the *shadow* BP routing. Virtual packets travel in the reverse direction in order to discover congestion and help route the actual data packets.

Our approach differs because it is based on local dropping and it does not require end-to-end signaling. An in-network flow control approach is proposed in [31], where a credit-based flow controller is shown to achieve *max-min* fairness, i.e. it solves the NUM problem for a specific choice of utility functions. We generalize in-network flow control for multirate multicast per-receiver NUM.

3.2 Network Model

Let $G=(\mathcal{N},\mathcal{L})$ be the graph, serving a set $\mathcal{C}$ of multicast sessions. Session $c\in C$ consists of a source node $s_c\in\mathcal{N}$[1] and a set of receivers $\Delta^{(c)}$, and is delivered over a given *tree* $G^{(c)}=(\mathcal{N}^{(c)},\mathcal{L}^{(c)})\subset G$. We consider a wireline network in which all links in E can be used simultaneously (we discuss wireless networks in section 3.5). Time is slotted and in slot t, $A^{(c)}(t)$ packets arrive at the source of session c. We assume that $A^{(c)}(t)$ are i.i.d. over time slots with mean $\lambda^{(c)}$ and take finite values, i.e. $A^{(c)}(t)\leq A_{\max}$.

Each node maintains one *transmission queue* for every outgoing link l and session c, let $Q_l^{(c)}(t)$ denote its backlog at the beginning of slot t. Define $p(l)\in\mathcal{L}^{(c)}$ to be the incoming (parent) link to that node[2] and let $\mathcal{L}_{\text{out}}(c)\subset\mathcal{L}^{(c)}$ be the set of outgoing links of the source node c. Queue $Q_l^{(c)}(t)$ evolves across slots according to

$$Q_l^{(c)}(t+1)\leq\left[\left(Q_l^{(c)}(t)-\mu_l^{(c)}(t)\right)^+-d_l^{(c)}(t)\right]^++A^{(c)}(t)1_{[l\in\mathcal{L}_{\text{out}}(c)]}+\mu_{p(l)}^{(c)}(t),$$
$$\forall l\in\mathcal{L}^{(c)},c\in\mathcal{C},\qquad(3.1)$$

where $\mu_l^{(c)}(t)$ is the allocated transmission rate and $d_l^{(c)}(t)$ is the number of packets that are dropped from $Q_l^{(c)}(t)$. Let $\mu_l^{\max}$ denote the capacity of link l. The capacity constraint $\sum_{c\in C}\mu_l^{(c)}(t)\leq\mu_l^{\max}$ must be satisfied in every slot. Also, we impose $d_l^{(c)}(t)\in[0,d_{\max}]$, where $d_{\max}$ is a system-defined parameter. Throughout the chapter, we assume $d_{\max}\geq A_{\max}+\mu_{\max}$, where $\mu_{\max}\triangleq\max_{l\in\mathcal{L}}\mu_l^{\max}$ is the maximum link capacity. The value $A_{\max}+\mu_{\max}$ is an upper bound to the incoming data rate to a node, and our choice of $d_{\max}$ ensures that the packet dropping rate is large enough so that all transmission queues can always be stabilized.

Let $\widetilde{d}_l^{(c)}(t)=\min[Q_l^{(c)}(t),d_l^{(c)}(t)]$ be the actual packets dropped from $Q_l^{(c)}(t)$ in slot t, which can be smaller than $d_l^{(c)}(t)$ if there are not enough packets to be dropped. To provide direct control over

[1] To simplify the notation we do not allow different sessions to have the same source. This limitation can be waived without affecting the results.

[2] Only one such incoming link $p(l)$ exists since $G^{(c)}$ is a tree. Note that the value of $p(l)$ depends on the multicast session under consideration, and we abuse the notation to simplify exposition.

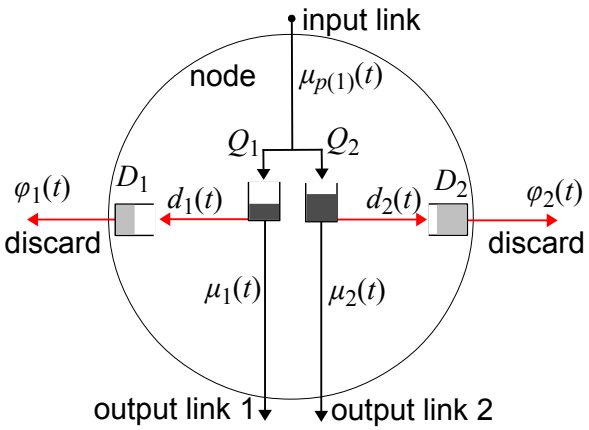


Figure 3.1: The proposed queue structure on an node with one incoming and two outgoing links (we show one session and we omit the session notation). Each link $l = 1,2$ is associated with a transmission queue $Q_l(t)$ and a dropping queue $D_l(t)$.

the amount of dropped packets, we set up a *drop queue* $D_l^{(c)}(t)$ associated with each transmission queue $Q_l^{(c)}(t)$. Before leaving the system, the dropped packets are "moved" to the drop queue from which they are later discarded according to the control policy. The drop queue $D_l^{(c)}(t)$ evolves across slots according to

$$D_l^{(c)}(t+1) = \left[D_l^{(c)}(t) - \varphi_l^{(c)}(t)\right]^+ + \widetilde{d}_l^{(c)}(t), \quad \forall l \in \mathcal{L}^{(c)}, c \in \mathcal{C}, \tag{3.2}$$

where $\varphi_l^{(c)}(t) \in [0, d_{\max}]$ is a decision variable that determines the number of packets that are eventually removed from the network in slot t. Note that the packets in drop queues $D_l^{(c)}(t)$ are not going to be transmitted, and therefore it practically suffices to keep track of the values of $D_l^{(c)}(t)$ only as counters.

Focusing on a network node, our queueing mechanism works as follows. All arriving packets are replicated to each transmission queue $Q_l^{(c)}(t)$, for example see Fig. 3.1. In a slot t, $\mu_l^{(c)}(t)$ determines the number of session c packets transmitted on link l, $d_l^{(c)}(t)$ decides the number of packets that are internally moved from $Q_l^{(c)}(t)$ to the drop queue $D_l^{(c)}(t)$, and $\varphi_l^{(c)}(t)$ is the number of packets that are discarded from queue $D_l^{(c)}(t)$ and permanently removed from the network. A control policy chooses the values of the decision variables, $\mu_l^{(c)}(t)$, $d_l^{(c)}(t)$ and $\varphi_l^{(c)}(t)$ at each slot.

3.3 Throughput Maximization

In order to introduce the concepts and the notation, we begin by considering the problem of maximizing the sum throughput of all receivers in multiple multicast sessions. In Section 3.4, we study the more general problem of utility maximization.

3.3.1 Flow-Level Characterization

Before we develop the dynamic network control policy, it is useful to provide a flow-level characterization of the optimal throughput vector and the optimal packet dropping rates, presented as solutions to linear optimization problems. These flow-level solutions are useful for proving the optimality of our control policies. However, the proposed policies solve these problems in a distributed and dynamic manner without the need to explicitly compute these solutions.

We define $f_l^{(c)}$ to be the average session c data flow rate over link l and $q_l^{(c)}$ the average packet dropping rate at queue $Q_l^{(c)}$. These flow variables must satisfy the flow conservation and link capacity constraints:

$$\lambda^{(c)} = f_l^{(c)} + q_l^{(c)}, \quad l \in \mathcal{L}_{\text{out}}(c),\ \forall c \tag{3.3}$$

$$f_{p(l)}^{(c)} = f_l^{(c)} + q_l^{(c)}, \quad l \in \mathcal{L}^{(c)} \setminus \mathcal{L}_{\text{out}}(c),\ \forall c \tag{3.4}$$

$$\sum_{c \in C} f_l^{(c)} \leq \mu_l^{\max},\ f_l^{(c)} = 0 \text{ if } l \notin G^{(c)}, \quad l \in E, \tag{3.5}$$

The packet dropping rate vector $(q_l^{(c)})$ is said to be feasible if there exist flow variables $f_l^{(c)}$ that satisfy (3.3)-(3.5).

Let $r_\delta^{(c)}$ denote the throughput of receiver $\delta \in \Delta^{(c)}$. Let Λ be the set of feasible throughput vectors $(r_\delta^{(c)})$. We have

$$\Lambda = \left\{ (r_\delta^{(c)}) \middle| \begin{array}{c} \text{(3.3)-(3.5) hold,} r_\delta^{(c)} = f_{l_\delta}^{(c)} \\ f_l^{(c)},\ q_l^{(c)} \geq 0 \end{array} \right\}, \tag{3.6}$$

where l_δ is the incoming link of the receiver δ in session c. In (3.6), $r_\delta^{(c)} = f_{l_\delta}^{(c)}$ states that the throughput of a receiver is equal to its incoming flow rate.

The problem of maximizing the sum throughput of all receivers in the network is

$$\text{maximize} \sum_{c,\delta} r_\delta^{(c)}, \quad \text{subject to } (r_\delta^{(c)}) \in \Lambda. \tag{3.7}$$

where the index c, δ in the summations hereinafter is a shorthand form of $c \in C$ and $\delta \in \Delta^{(c)}$. It is useful to consider an equivalent optimization problem that minimizes packet dropping rates. Let $\mathcal{P}_\delta^{(c)}$ denote the set of links that form the path from the source node s_c to a receiver δ. Summing (3.4) over $l \in \mathcal{P}_\delta^{(c)}$ and using $r_\delta^{(c)} = f_{l_\delta}^{(c)}$, we have

$$r_\delta^{(c)} = \lambda^{(c)} - \sum_{l \in \mathcal{P}_\delta^{(c)}} q_l^{(c)}, \tag{3.8}$$

which states that the throughput $r_\delta^{(c)}$ of receiver δ is equal to the exogenous data arrival rate less the sum of packet dropping rates along the path $\mathcal{P}_\delta^{(c)}$ to δ. Summing (3.8) over all receivers $\delta \in \Delta^{(c)}$ in a

session, the total session c throughput is

$$\sum_{\delta \in \Delta^{(c)}} r_{\delta}^{(c)} = \left|\Delta^{(c)}\right| \lambda^{(c)} - \sum_{l \in \mathcal{L}^{(c)}} m_l^{(c)} q_l^{(c)}, \tag{3.9}$$

where $m_l^{(c)}$ is the number of session c receivers connected to their source via link l.[3] From (3.9) we see that maximizing the total throughput of session c is equivalent to minimizing the weighted packet dropping rate $\sum_{l \in \mathcal{L}^{(c)}} m_l^{(c)} q_l^{(c)}$. Consequently, the throughput maximization problem (3.7) is equivalent to the minimization problem

$$\text{minimize} \sum_{c,l} m_l^{(c)} q_l^{(c)}, \quad \text{subject to } (q_l^{(c)}) \text{ feasible.} \tag{3.10}$$

where the index l in the summations hereinafter is a shorthand form of $l \in \mathcal{L}^{(c)}$. Next, we design a control policy that stabilizes all queues in the network and achieves optimal packet dropping rates; from the equivalence of (3.7) and (3.10), our policy achieves the maximum total throughput as well.

3.3.2 Intuition for Packet-Level Control

To measure the degree of congestion in the network, we construct a strictly increasing function of the queue backlogs $Q_l^{(c)}(t)$ and $D_l^{(c)}(t)$, i.e., we define the weighted quadratic Lyapunov function

$$L(t) = \frac{1}{2} \sum_{c,l} m_l^{(c)} ([Q_l^{(c)}(t)]^2 + [D_l^{(c)}(t)]^2).$$

The quadratic terms are weighted by $m_l^{(c)}$ because the importance of a queue is proportional to the number of receivers connected to that queue. Let $H(t) = (Q_l^{(c)}(t); D_l^{(c)}(t))$ be the queue backlog vector in slot t. Define the Lyapunov drift

$$\Delta(t) = \mathbb{E}\left[L(t+1) - L(t) \mid H(t)\right] \tag{3.11}$$

as the expected difference of the congestion measure $L(t)$ over a slot. A control policy that minimizes the Lyapunov drift in every slot suffices to stabilize the network and keep the queue backlogs bounded [26].

Recall from (3.10), that we also seek to minimize the weighted time-average packet dropping rate

$$\sum_{c,l} m_l^{(c)} q_l^{(c)} = \sum_{c,l} m_l^{(c)} \overline{d}_l^{(c)} \triangleq \sum_{c,l} m_l^{(c)} \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}[\widetilde{d}_l^{(c)}(t)]. \tag{3.12}$$

If a drop queue $D_l^{(c)}(t)$ is stable, then from queueing theory its arrival rate must be less than or equal

[3] We have $m_l^{(c)} = 0$ if $l \notin \mathcal{L}^{(c)}$.

to its time-average service rate, i.e., from (3.2) we have

$$\lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}[\widetilde{d}_l^{(c)}(t)] \leq \lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}[\varphi_l^{(c)}(t)]. \tag{3.13}$$

Our approach forces this bound to be tight, and hence minimizing (3.12) can be achieved by minimizing its upper bound in (3.13), provided all $D_l^{(c)}(t)$ queues are stable. In fact, it suffices to minimize in every slot the sum $\sum_{c,l} m_l^{(c)}\mathbb{E}_H[\varphi_l^{(c)}(t)]$, where $\mathbb{E}_H[\cdot]$ is a compact notation for the conditional expectation $\mathbb{E}[\cdot \mid H(t)]$.

Minimizing both the Lyapunov drift $\Delta(t)$ and the sum $\sum_{c,l} m_l^{(c)}\mathbb{E}_H[\varphi_l^{(c)}(t)]$ induces a conflict, because the network becomes more congested when less packets are dropped. It is therefore natural to consider minimizing a weighted sum of these two metrics,

$$\Delta(t) + V\sum_{c,l} m_l^{(c)}\,\mathbb{E}_H[\varphi_l^{(c)}(t)], \tag{3.14}$$

where $V > 0$ is predefined parameter that reflects the relative importance of minimum packet dropping to queue stability. As we will see, V also controls a tradeoff between the performance gap of our policy from optimality and the required finite buffer size in the transmission queues $Q_l^{(c)}(t)$.

3.3.3 The Proposed Policy

Our policy arises from the minimization of (3.14). In Appendix A, performing standard calculation of the Lyapunov drift, we obtain a bound

$$\begin{aligned}(3.14) \leq\ & \underbrace{B_1 + \sum_{c,l\in\mathcal{L}_{\text{out}}(c)} m_l^{(c)} Q_l^{(c)}(t)\,\lambda^{(c)}}_{\text{constant}} \\ & \underbrace{-\sum_{c,l} m_l^{(c)}\mathbb{E}_H[d_l^{(c)}(t)]\big(Q_l^{(c)}(t) - D_l^{(c)}(t)\big)}_{\text{dropping}} \\ & \underbrace{-\sum_{c,l} m_l^{(c)}\mathbb{E}_H[\varphi_l^{(c)}]\big(D_l^{(c)}(t) - V\big)}_{\text{discarding}} \\ & \underbrace{-\sum_{c,l}\mathbb{E}_H[\mu_l^{(c)}(t)]W_l^{(c)}(t)}_{\text{scheduling}}\end{aligned} \tag{3.15}$$

where $B_1 > 0$ is a finite constant given in Appendix A and

$$W_l^{(c)}(t) \triangleq m_l^{(c)}Q_l^{(c)}(t) - \sum_{l':p(l')=l} m_{l'}^{(c)}Q_{l'}^{(c)}(t) \tag{3.16}$$

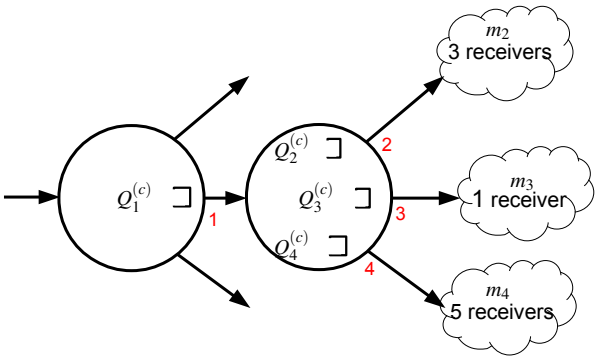


Figure 3.2: Illustration of the differential backlog calculation in MMT policy; $W_1^{(c)} = 9Q_1^{(c)} - 3Q_2^{(c)} - Q_3^{(c)} - 5Q_4^{(c)}$.

Maximum Multicast Throughput (MMT) Policy

Packet Dropping: Each transmission queue $Q_l^{(c)}(t)$ moves $\min\{d_l^{(c)}(t), Q_l^{(c)}(t)\}$ packets to its drop queue $D_l^{(c)}(t)$ at the end of slot t, where

$$d_l^{(c)}(t) = \begin{cases} d_{\max} & \text{if } Q_l^{(c)}(t) > D_l^{(c)}(t) \\ 0 & \text{otherwise.} \end{cases} \tag{3.17}$$

The drop queue $D_l^{(c)}(t)$ removes $\min\{\varphi_l^{(c)}(t), D_l^{(c)}(t)\}$ packets from the network according to

$$\varphi_l^{(c)}(t) = \begin{cases} d_{\max} & \text{if } D_l^{(c)}(t) > V \\ 0 & \text{otherwise.} \end{cases} \tag{3.18}$$

Scheduling: Let C_l be the set of multicast sessions that use link l. Define $W_l^*(t) = \max_{c \in C_l} W_l^{(c)}(t)$ and let c_l^* be a maximizer session (ties are broken arbitrarily). We allocate the link rate

$$\mu_l^{(c_l^*)}(t) = \begin{cases} \mu_l^{\max} & \text{if } W_l^*(t) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.19}$$

Let $\mu_l^{(c)}(t) = 0$ for all the other sessions $c \in C_l \setminus \{c_l^*\}$.

---

is the weighted differential backlog. Fig. 3.2 gives an example calculation of $W_l^{(c)}(t)$. We propose a throughput-optimal policy that is designed to minimize the RHS of (3.15) at each slot, named MMT.

Observe that (3.17) minimizes the dropping term of (3.15), (3.18) minimizes the discarding term and (3.19) minimizes the scheduling term. Since the first two terms in (3.15) are constant, we conclude that MMT minimizes the RHS of (3.15).

We note that the policy operates in a distributed manner using only locally available information. For the computation of $W_l^{(c)}(t)$, we require knowledge of the neighbor backlogs. As shown in prior

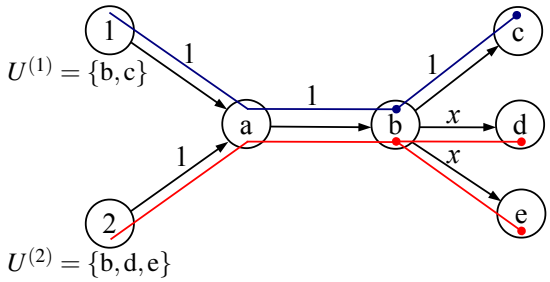


Figure 3.3: An example of multirate multicast with two sessions. Session 1 uses the link set $\mathcal{L}^{(1)} = \{(1,a),(a,b),(b,c)\}$ and session 2 uses the link set $\mathcal{L}^{(2)} = \{(2,a),(a,b),(b,d),(b,e)\}$. The set of receivers are denoted with $\Delta^{(1)}, \Delta^{(2)}$. Numbers on links indicate capacities.

work, this is not restrictive for practical applications, e.g. see [32]. Also, delayed backlog information is sufficient for throughput optimality, see [10, §4.7].

3.3.4 Performance Evaluation of MMT

Due to the dropping mechanism in (3.17)-(3.18), $Q_l^{(c)}(t)$ and $D_l^{(c)}(t)$ are deterministically bounded. Applying the approach of [33] we have the following result.

Lemma 1. All queues $Q_l^{(c)}(t)$ and $D_l^{(c)}(t)$ are deterministically bounded by

$$Q_l^{(c)}(t) \leq V + 2d_{\max},\ D_l^{(c)}(t) \leq V + d_{\max},\ \forall l, c, t.$$

Hence, a buffer size of $V + 2d_{\max}$ is sufficient to avoid unexpected queue overflow at $Q_l^{(c)}(t)$. The MMT policy achieves near-optimal total throughput as the following theorem asserts.

Theorem 3.1 (Optimality of MMT)**.** The MMT policy yields the total throughput satisfying

$$\sum_{c,\delta} \bar{r}_\delta^{(c)} \geq \sum_{c,\delta} r_\delta^{(c)*} - \frac{B_1}{V}.$$

Where

$$\bar{r}_\delta^{(c)} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\left[\widetilde{\mu}_{l_\delta}^{(c)}(t)\right]$$

is the throughput of receiver δ in multicast session c and $(r_\delta^{(c)*})$ is a solution to (3.7).

The performance gap B_1/V can be made arbitrarily small by choosing a sufficiently large $V > 0$. Appendix B provides the proof of Theorem 3.1.

3.3.5 Simulation of MMT

We illustrate how MMT adapts to changing conditions via simulations. Consider the multicast scenario of Fig. 3.3. Two multicast sessions share link (a,b). The set of multicast receivers are

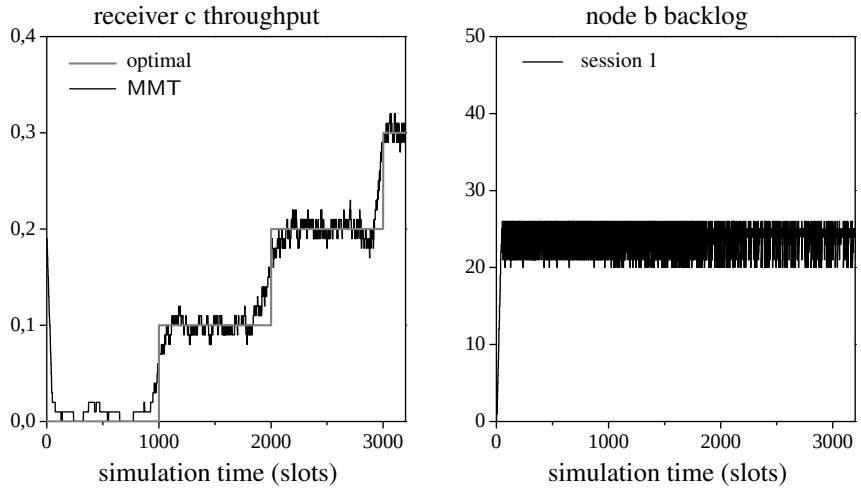


Figure 3.4: Performance when varying quality for links $(\mathrm{b},\mathrm{d}),(\mathrm{b},\mathrm{e})$ in the topology of Fig. 3.3. The left Figure compares MMT to the optimal average throughput of receiver c. The right Figure shows the backlog of node b with packets for transmission to receiver c.

$\Delta^{(1)} = \{\mathrm{b},\mathrm{c}\}$ for session 1 and $\Delta^{(2)} = \{\mathrm{b},\mathrm{d},\mathrm{e}\}$ for session 2. Links (b,d) and (b,e) have capacity x, while the rest links have unit capacities, i.e. 1packet/slot. Both sessions have unit arrival rate. We seek to maximize total throughput.

Observe that throughput maximization depends crucially on the value of x. For example, if $x = 1$, then maximum throughput is achieved by allocating all the resources of link (a,b) to session 2, since session 2 has three receivers and session 1 has two. If on the other hand $x = 0$, then maximum throughput is achieved by allocating all the resources of link (a,b) to session 1. In general, for $x \in [0,1]$, throughput is maximized if the allocation on link (a,b) is x to session 2 and $1-x$ to session 1. Note, that the packet dropping decision of node $\{\mathrm{a}\}$ depends on the quality of links (b,d) and (b,e), information which is not directly available at $\{\mathrm{a}\}$.

In the simulation we vary the value x. Initially $x = 1$ and gradually x reduces in steps of 0.1. Fig. 3.4 (left) shows the receiver c throughput. According to the above discussion, the optimal average throughput is equal to $1-x$, showcased in the Figure with gray line. The simulations showed that the average throughput of MMT is equal to the optimal. Hence, we show the instantaneous throughput averaged in moving windows of 100 slots. At each interval, the throughput converges quickly to the optimal, which shows how MMT adapts to changing conditions.

In Fig. 3.4 (right), we showcase the backlog at node b with packets destined to node c, for the same sample path. In this simulation we have used $V = 25$ and $d_{\max} = 5$ and by Lemma 1, the backlog is upper bounded by 35 packets. In the simulations, the backlog never exceeds 25 packets despite the link quality variations and the randomness of the arrivals.

3.4 Utility maximization

Next we consider the per-receiver NUM problem. Solving this general problem allows to use different utility functions to achieve several objectives such as maximum throughput (studied separately in the previous section), $\alpha-$fairness which includes proportional fairness and max-min fairness as special cases, user priority, and satisfying user-specific quality requirements.

3.4.1 Per-Receiver NUM Problem Formulation

In multicast session c, a receiver δ has a utility function $g_\delta^{(c)}$, which is assumed to be concave, increasing and continuously differentiable with bounded derivatives.[4] Consider the per-receiver NUM problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{c,\delta} g_\delta^{(c)}(r_\delta^{(c)}) \\ \text{subject to} \quad & (r_\delta^{(c)}) \in \Lambda. \end{aligned} \tag{3.20}$$

Define the auxiliary function

$$h_\delta^{(c)}(x) \triangleq g_\delta^{(c)}(x) - \theta x,$$

where $\theta > 0$ is a parameter decided later. Then, maximizing the total utility $\sum_{c,\delta} g_\delta^{(c)}(r_\delta^{(c)})$ is equivalent to maximizing

$$\begin{aligned} &\sum_{c,\delta} \left(h_\delta^{(c)}(r_\delta^{(c)}) + \theta\, r_\delta^{(c)}\right) \\ &\quad = \sum_{c,\delta} h_\delta^{(c)}(r_\delta^{(c)}) + \theta \sum_{c,\delta} \left(\lambda^{(c)} - \sum_{l \in \mathcal{P}_\delta^{(c)}} q_l^{(c)}\right) \\ &\quad = \sum_{c,\delta} h_\delta^{(c)}(r_\delta^{(c)}) - \theta \sum_{c,l} m_l^{(c)} q_l^{(c)} + \theta \sum_{c,\delta} \lambda^{(c)}, \end{aligned} \tag{3.21}$$

where the last sum is an (unknown) constant. In what follows, we modify our mechanism so that by controlling functions $\mu_l^{(c)}(t), d_l^{(c)}(t), \varphi_l^{(c)}(t)$ and a new virtual queue which will introduce next, the system is driven to the solution of (3.21).

3.4.2 Receiver virtual queue $Z_\delta^{(c)}(t)$

At each multicast receiver δ, we set up the virtual queue $Z_\delta^{(c)}(t)$, which tracks the deficit/surplus of session c packets received at that user and evolves as

$$Z_\delta^{(c)}(t+1) = [Z_\delta^{(c)}(t) - \nu_\delta^{(c)}(t)]^+ + \widetilde{\mu}_{l_\delta}^{(c)}(t),$$

[4] We assume $[g_\delta^{(c)}]'(x) \le [g_\delta^{(c)}]'(0) < \infty$. Utility functions that have unbounded derivatives as $x \to 0$, such as $\log(x)$, can be approximated by those with bounded derivatives. For example, we can approximate $\log(x)$ by $\log(x+\xi)$ for some small $\xi > 0$.

where $l_\delta \in E_\delta^{(c)}$ is the incoming link of node δ, and

$$\widetilde{\mu}_{l_\delta}^{(c)}(t) = \min\left[Q_{l_\delta}^{(c)}(t), \mu_{l_\delta}^{(c)}(t)\right]$$

is the actual number of packets delivered to that user. The departures $\nu_\delta^{(c)}(t)$ are controlled by the policy and chosen in the interval $[0, \nu_{\max}]$, we choose $\nu_{\max}$ below. The functionality of this virtual queue is to track the urgency of a receiver to obtain more packets: if $Z_\delta^{(c)}(t)$ is small, receiver δ must urgently obtain packets for the maximum utility to be preserved.

We also define the *virtual pressure* for each receiver δ which is regulated by the virtual queue:

$$Y_\delta^{(c)}(t) \triangleq \begin{cases} we^{w(Z_\delta^{(c)}(t)-\zeta)}, & \text{if } Z_\delta^{(c)}(t) \geq \zeta, \\ -we^{w(\zeta - Z_\delta^{(c)}(t))}, & \text{otherwise,} \end{cases} \tag{3.22}$$

where w, ζ are positive parameters whose value will be chosen later. Note, that in normal backpressure, the pressure of a destination node is zero, while in our policy $Y_\delta^{(c)}(t)$ can take positive or even negative values. The sign of $Y_\delta^{(c)}(t)$ indicates the urgency of the particular receiver to obtain more or less packets according to the requested objective. Indeed, the behavior of $Y_\delta^{(c)}(t)$ is controlled by $\nu_\delta^{(c)}(t)$, which as we will see shortly, is chosen according to the utility function.

3.4.3 The Proposed Policy

Let $H(t) = (Q_l^{(c)}(t); D_l^{(c)}(t); Z_\delta^{(c)}(t))$ be the joint queue backlog vector in slot t. Define the Lyapunov function

$$L(t) = \frac{1}{2}\sum_{c,l} m_l^{(c)}\left([Q_l^{(c)}(t)]^2 + [D_l^{(c)}(t)]^2\right) + \frac{1}{2}\sum_{c,\delta}\left(e^{w(Z_\delta^{(c)}(t)-\zeta)} + e^{w(\zeta - Z_\delta^{(c)}(t))}\right).$$

Note, that the Lyapunov function is composed of two terms, the quadratic term is identical to the Lyapunov function used in throughput maximization section, while the exponential term is identical to the one used for receiver-based flow control for unicast sessions in [34].

Recall the definition of Lyapunov drift $\Delta(t)$ from (3.11). In order to solve the problem in (3.21) we define the weighted objective:

$$\Delta(t) + V\left[\theta\sum_{c,l} m_l^{(c)}\mathbb{E}_H\left[\varphi_l^{(c)}(t)\right] - \sum_{c,\delta}\mathbb{E}_H\left[h_\delta^{(c)}(\nu_\delta^{(c)}(t))\right]\right]. \tag{3.23}$$

Using standard drift derivation techniques we obtain the following bound in the Appendix C

$$
\begin{aligned}
(3.23) \leq & \underbrace{B_2 + \sum_{c,l \in \mathcal{L}_{\text{out}}(c)} m_l^{(c)} Q_l^{(c)}(t)\, \lambda^{(c)} + \frac{\varepsilon}{2} \sum_{uc} Y_\delta^{(c)}(t)}_{\text{(constant)}} \\
& \underbrace{- \sum_{c,l} m_l^{(c)} \mathbb{E}_H[d_l^{(c)}(t)](Q_l^{(c)}(t) - D_l^{(c)}(t))}_{\text{(dropping)}} \\
& \underbrace{- \sum_{c,l} m_l^{(c)} \mathbb{E}_H[\varphi_l^{(c)}(t)](D_l^{(c)}(t) - V\theta)}_{\text{(discarding)}} \\
& \underbrace{- \sum_{uc} \mathbb{E}_H \left\{ V h_\delta^{(c)}(\nu_\delta^{(c)}(t)) + Y_\delta^{(c)}(t) \nu_\delta^{(c)}(t) \right\}}_{\text{(flow control)}} - \underbrace{\Psi(t)}_{\text{(scheduling)}}
\end{aligned} \tag{3.24}
$$

where $\varepsilon > 0$ is a parameter, B_2 is a large constant defined in the Appendix C and

$$
\Psi(t) \triangleq - \sum_{c,\delta} \mathbb{E}_H[\mu_{l_\delta}^{(c)}(t)]\, Y_\delta^{(c)}(t) + \sum_{c,l} m_l^{(c)} Q_l^{(c)}(t)\, \mathbb{E}_H\left[\mu_l^{(c)}(t) - \mu_{p(l)}^{(c)}(t)\right].
$$

Let $1_{[l,\delta]}^{(c)}$ be the indicator function on the event that the tail node of l on $G^{(c)}$ is a receiver $\delta \in \Delta^{(c)}$. Then, define the weighted differential backlog as

$$
W_l^{(c)}(t) = m_l^{(c)} Q_l^{(c)}(t) - \sum_{l' :\, p(l') = l} m_{l'}^{(c)} Q_{l'}^{(c)}(t) - 1_{[l,\delta]}^{(c)} Y_\delta^{(c)}(t). \tag{3.25}
$$

Observe that the virtual pressure $Y_\delta^{(c)}(t)$ is applied only if the tail node of l is a receiver for this session. By rearranging terms, we have

$$
\Psi(t) = \sum_{c,l} \mathbb{E}_H[\mu_l^{(c)}(t)]\, W_l^{(c)}(t).
$$

We design our Maximum Multicast Utility (MMU) policy to minimize the RHS of (3.24). To achieve this, we add a receiver-end flow controller, similar to the one used in[34].

3.4.4 Performance Evaluation of MMU

Lemma 2. Under the MMU policy, all queues $Q_l^{(c)}(t)$, $D_l^{(c)}(t)$, and $Z_\delta^{(c)}(t)$ are deterministically bounded by

$$
\begin{aligned}
& Q_l^{(c)}(t) \leq V\theta + 2d_{\max},\ D_l^{(c)}(t) \leq V\theta + d_{\max},\ \forall c, l, t, \\
& Z_\delta^{(c)}(t) \leq \zeta + \frac{1}{w} \log\left(\frac{V\theta + 2d_{\max}}{w} \right) + \mu_{\max},\ \forall c, \delta, t.
\end{aligned}
$$

Maximum Multicast Utility (MMU) Policy

Parameter Selection: Choose positive parameters V, $d_{\max}$, $\nu_{\max}$, w, ζ, and θ as summarized in the Table 3.1. For a discussion on these parameter choices see [33, §V-C]. Initialize the queues with $Q_l^{(c)}(0) = 0$, $Z_\delta^{(c)}(0) = \zeta + \frac{1}{w}\log\left(\frac{V\theta}{w}\right)$ and $D_l^{(c)}(0) = V\theta$.
Packet Dropping: Same as in MMT policy.
Receiver-End Flow Control: Choose $\nu_\delta^{(c)}(t)$ to be the solution to

$$\text{maximize} \quad Vh_\delta^{(c)}(x) + Y_\delta^{(c)}(t)\,x \tag{3.26}$$
$$\text{subject to} \quad 0 \le x \le \nu_{\max}, \tag{3.27}$$

where $Y_\delta^{(c)}(t)$ is given in (3.22).
Scheduling: Same as in MMT policy, except that we use (3.25) as the value of $W_l^{(c)}(t)$, instead of (3.16).

Table 3.1: Parameter Selection for MMT, MMU.

Parameter	Explanation	Suggested values
$m_l^{(c)}$	number of session c users connected to source c through link l	problem defined
$\mu_l^{\max}$	capacity of link l	problem defined
$\mu_{\max}$	maximum link capacity	$\max_l \mu_l^{\max}$
$d_{\max}$	drop batch size	$d_{\max} \ge A_{\max} + \mu_{\max}$
ε	utility gap parameter	> 0
$\nu_{\max}$	maximum value for $\nu_u^{(c)}(t)$	$\mu_{\max} + \varepsilon/2$
$\delta_{\max}$	bound on $\lvert\nu_\delta^{(c)}(t) - \mu_{l_\delta}^{(c)}(t)\rvert$	$\max[\nu_{\max}, \mu_{\max}]$
w	multiplier in (3.22)	$\frac{\varepsilon}{\delta_{\max}^2} e^{-\varepsilon/\delta_{\max}}$
ζ	central value for $Z_\delta^{(c)}(t)$	$\ge \nu_{\max}$
$g_\delta^{(c)}(x)$	user utility function	objective specific
θ	upper bound on $[g_\delta^{(c)}]'(x)$, $x \ge \varepsilon$	$\max_{u,c}[g_\delta^{(c)}]'(0)$
$h_\delta^{(c)}(x)$	auxiliary function	$g_\delta^{(c)}(x) - \theta x$
V	utility gap/backlog size tradeoff	$V\theta + 2d_{\max} \ge w$

Theorem 3.2 (Optimality of MMU)**.** The MMU policy achieves the long-term utility satisfying

$$\sum_{c,\delta} g_\delta^{(c)}(\overline{r}_\delta^{(c)}) \ge \sum_{c,\delta} g_\delta^{(c)}(r_\delta^{(c)*}) - \frac{B_2}{V} - \frac{3\varepsilon}{2}\sum_{c,\delta}\left([g_\delta^{(c)}]'(0) + \theta\right), \tag{3.28}$$

where $(r_\delta^{(c)*})$ is the utility-optimal throughput vector.

3.4.5 Achieving Throughput Requirements

We show how to use the MMU policy to deliver a video stream to users with strict throughput requirements. Consider the optimization problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{c,\delta} g_\delta^{(c)}(r_\delta^{(c)}) \\ \text{subject to} \quad & (r_\delta^{(c)}) \in \Lambda, \\ & (r_\delta^{(c)}) \geq (\xi_\delta^{(c)}), \end{aligned} \tag{3.29}$$

where the inequality is element-wise and $\xi_\delta^{(c)}$ denotes the throughput requirement of session c receiver δ. We assume problem (3.29) admits a feasible solution. In order to solve (3.29) using MMU, we use the penalty method, see [35, §4.7]. Define the penalty function

$$\pi\big[(x_\delta^{(c)})\big] \triangleq K \sum_{c,u\in U^{(c)}} \left(\xi_\delta^{(c)} - x_\delta^{(c)}\right)^+,$$

where $(x_\delta^{(c)})$ is a vector with one element for every receiver-session pair. If all requirements are satisfied (i.e. $r_\delta^{(c)} \geq \xi_\delta^{(c)},\ \forall\delta$) then $\pi\big[(r_\delta^{(c)})\big] = 0$. If some requirement is violated, then $\pi\big[(r_\delta^{(c)})\big]$ increases proportionally to K and to the norm-1 distance of $(r_\delta^{(c)})$ from the feasible set. Also note that π is convex and thus $-\pi$ is concave. Next, consider a convex optimization problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{c,\delta} \left[g_\delta^{(c)}(r_\delta^{(c)}) - K\left(\xi_\delta^{(c)} - r_\delta^{(c)}\right)^+ \right] \\ \text{subject to} \quad & (r_\delta^{(c)}) \in \Lambda, \end{aligned} \tag{3.30}$$

By letting $K \to \infty$, the solution of (3.30) converges to the solution of (3.29) [35]. A practical approach is to pick a "large" finite value for K.

3.4.6 Simulations: Prioritizing Base Layer Packets

In multimedia streaming with layer coding, the stream reconstruction requires the reception of specific data packets, belonging to the *base layer*. Then, the reception of additional *enhancement layer* packets improves the quality of the stream. Therefore, a reasonable strategy is to maximize the number of enhancement layer packets subject to the correct reception of base layer packets at each receiver. We show next how to tune MMU to have such a behavior.

We revisit the example of Fig. 3.3 and set $x = 1$ so that all links have unit capacities. Next, we tag the packets belonging to the base layer video to distinguish them from the enhancement layer packets. The video stream of each session is modeled by a superposition of two Poisson processes with $\lambda_{\text{base}} = 0.2$ and $\lambda_{\text{enh}} = 0.8$.

Using the penalty approach explained in the previous subsection, it is possible to achieve throughput maximization subject to rate 0.2 at each receiver. We choose $g_\delta^{(1)}(x) = g_\delta^{(2)}(x) = x$, $\xi_\delta^{(1)} = 0.2$ and

Table 3.2: MMU simulation results for prioritizing base layer packets.

	Session 1		Session 2		
receivers	b	c	b	d	e
stream rate	0.996		0.998		
$\xi_\delta^{(c)}$	0.2	0.2	0.2	0.2	0.2
$\bar{r}_\delta^{(c)}$	0.1948	0.1948	0.805	0.805	0.805
base layer packets breakdown					
stream rate	0.1997		0.199		
received rate	0.1944	0.1944	0.199	0.199	0.199
delivery ratio	97.35%	97.35%	100%	100%	100%
enhancement layer packets breakdown					
stream rate	0.7963		0.799		
received rate	0.0003	0.0003	0.606	0.606	0.606
delivery ratio	0.037%	0.037%	75.84%	75.84%	75.84%

$\xi_\delta^{(2)} = 0.2$ for all $\delta \in \Delta^{(1)}, \Delta^{(2)}$ respectively. However, additionally to achieving a specific throughput rate requirement, we require the reception of specific packets. To cope with this added constraint, we impose a strict priority rule at all transmission queues $Q_l^{(c)}$: *enhancement layer packets are only served if there are no base layer packets left in the queue.*

The resulting experiments for this scenario are shown in Table 3.2. The combination of MMU with the priority rule provides delivery ratio of base layer packets very close to the ideal 100%. The small loss is attributed to randomness of the arrivals. Moreover, when the base layer packet delivery ratio is less than 100%, the enhancement layer counterpart is very small. Conclusively, our policy achieves the high-level goal to combine guaranteed delivery with optimal performance.

3.5 Experimentation Results

To demonstrate the practicality of the MMU policy, we develop a prototype implementation in NITOS testbed [6]. The main hardware and software specifications of the NITOS nodes are depicted in Table 2.1. During our experimentation, the nodes used 802.11a.

3.5.1 Implementation Framework

The implementation is based on the Click Modular router framework [7]. Click facilitates experimentation and evaluation of scheduling and flow control algorithms in real systems. It runs as a user-level daemon at each node and via the libpcap library it provides full control on packet transmission. Our implemented framework includes mechanisms for estimating channel quality, forming a queue structure, exchanging queue backlog information, and splitting time into virtual slots.

Estimating Channel Quality. To evaluate channel quality, we adopted the ETT estimation algorithm of Roofnet [18]. Nodes periodically broadcast probes which are used to estimate the successful

transmission probability. With this process every node periodically obtains a table with the qualities for each channel rate/neighbor pair. Using this table, the $\mu_l^{\max}$ parameters are determined. This mechanism is known to incur negligible throughput overhead [1], [18].

Queue Structure. We implement the transmission queues $Q_l^{(c)}$ on each node and we create a counter for each $D_l^{(c)}$, $Z_l^{(c)}$ virtual queue. The counter $Z_l^{(c)}$ may take non-integer values. Each of these internal queues/counters is created upon the arrival of the first packet of a new session. This allows session generation "on the fly". The queues are removed after a period of inactivity.

Exchanging Queue Backlog Information. To compute $W_l^{(c)}$ from (3.25), each node broadcasts periodically the backlog size of all its transmission queues $Q_l^{(c)}$. If a node δ is a receiver for some session c, it broadcasts $Q_l^{(c)} + Y_\delta^{(c)}$ instead. The broadcast messaging is repeated once every second. Prior experiments suggest that more frequent broadcasts incur visible throughput overhead, while rarer broadcasts may affect the delay performance due to obsolete queue information.

In the proposed schemes, the routing is based on fixed multicast trees. Thus $m_l^{(c)}$ parameters are predefined and known. However, in our implementation, it is possible to use the backlog exchange mechanism to transport information about $m_l^{(c)}$, should these be time-varying.

Virtual Slots. In order to simplify the implementation we use the concept of the virtual slot. Each node keeps an internal timer that expires once every slot. Upon counter expiration the policy selects the next queue to be served and for the duration of the next slot the decision remains fixed. The slot duration is set to 100msecs, equal to 1/10 of the broadcasts period. Small values for the slot duration improve delay and reduce throughput fluctuations but burden the CPU of the device. We leave the investigation of optimum slot duration for future work. We note that the implementation of MMU is not tied to the idea of the virtual slot.

3.5.2 Policy Implementation

We modify our proposed policy so that it can operate on a network with wireless channels. Due to interference, some wireless links cannot be activated simultaneously. A well known link activation policy is the *maxweight* policy, proposed in [8] for stabilizing mobile packet networks. Maxweight activates at each slot the set of links that maximize the sum products $\sum_l \mu_l^{\max} W_l^{(c)}(t)$, effectively preferring links with higher capacity. In our setting, the activation of the transmitting nodes is automatically selected by the IEEE 802.11 protocol. Thus, it remains to choose the activation of a session and a receiving link, subject to the activated nodes. Using intuition from the maxweight policy we propose the following heuristic.

3.5.3 Experiments and Results

We conduct experiments on the specific topology of Figure 3.5. Five NITOS nodes are used: Alice and Bob are connected via Ethernet while Bob is connected to the other three nodes via wireless. The nodes are configured to run the MMU-W policy. The wireless links use fixed physical rates instead of the 802.11 rate adaptation scheme. In particular we set the physical rates to 18Mb/s, 6Mb/s and

Maximum Multicast Utility for Wireless (MMU-W) Policy

Parameter Selection, Packet Dropping, Receiver-End Flow Control, Scheduling on Wired Links: same as in MMU.

Scheduling on Wireless Links: Calculate $W_l^{(c)}(t)$ using (3.25). On a wireless node, choose the link-session pair

$$(l^*, c^*) \in \arg\max_{(l,c)} \mu_l^{\max} W_l^{(c)}(t) 1_{[W_l^{(c)}(t)>0]}$$

ties broken arbitrarily. Then, allocate the rate

$$\mu_{l^*}^{(c^*)}(t) = \begin{cases} \mu_{l^*}^{\max} & \text{if } W_{l^*}^{(c^*)}(t) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mu_l^{(c)}(t) = 0$ for all the other link-session pairs.

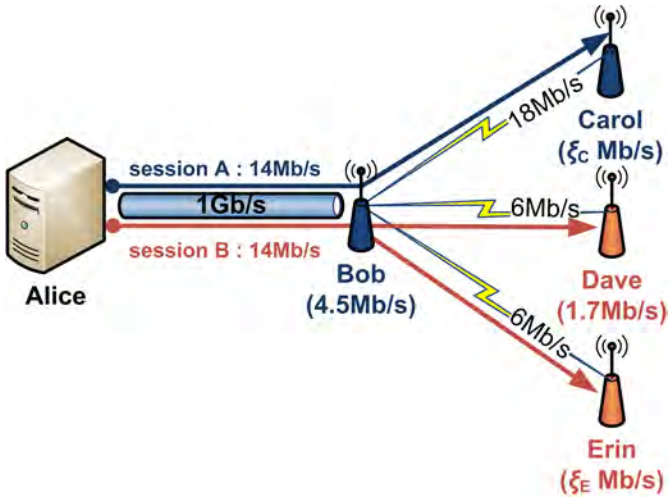


Figure 3.5: Experiment topology with five NITOS nodes. Two sessions A and B are generated at Alice, forwarded to Bob via a wired connection, and then distributed to Carol, Dave, and Erin through wireless. The Figure shows the rate requirement per receiver (in parentheses) and the physical rate per link.

6Mb/s for the links to Carol, Dave, and Erin respectively. The physical rate of the wired connection is 1Gb/s.

We consider two sessions, A and B, each with traffic rate 14Mb/s. The source node for both sessions is Alice and the multicast receivers are {Bob, Carol} for A, and {Dave, Erin} for B, see Fig. 3.5. To generate packets we use two UDP streams created with the Iperf tool [21]. We run the Iperf tool on external nodes to avoid polluting the CPU measurements. The receiver rate requirements are 4.5Mb/s for Bob, ξ_CMb/s for Carol, 1.7Mb/s for Dave and ξ_EMb/s for Erin, where the values ξ_C, ξ_E are chosen differently per experiment. The objective is to satisfy all receiver rate requirements as well as achieve maximum throughput.

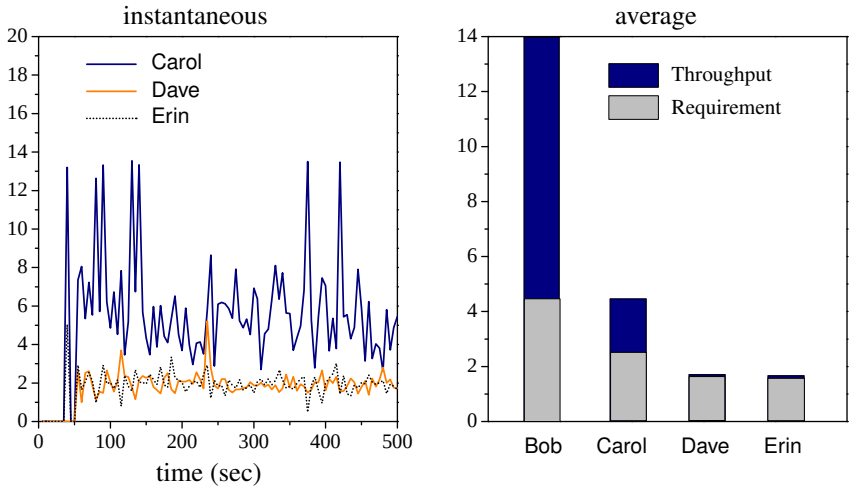


Figure 3.6: Scenario 1: $(\xi_C, \xi_E) = (2.8, 1.7)$. Instantaneous and average throughput (Mb/s) are shown.

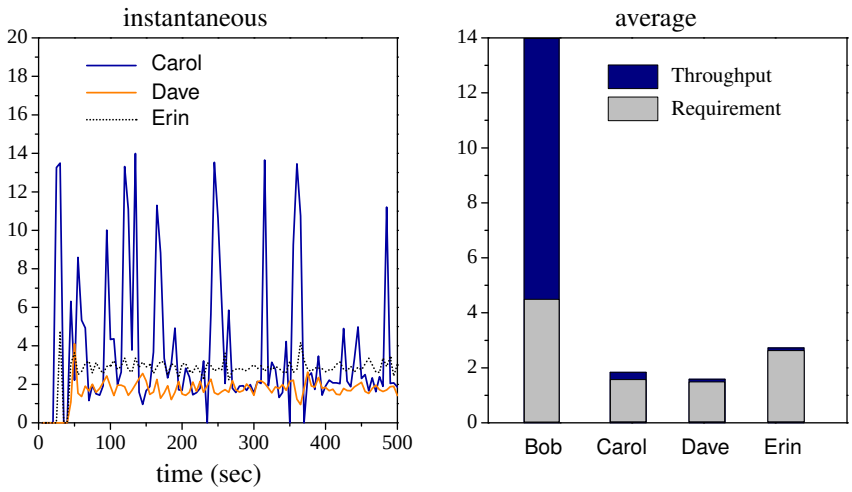


Figure 3.7: Scenario 2: $(\xi_C, \xi_E) = (1.7, 2.8)$. Instantaneous and average throughput (Mb/s) are shown.

3.5.4 Throughput

We show the measured instantaneous and average throughput for two scenarios. The instantaneous throughput is computed as the average over 1sec periods. In the first scenario we choose $(\xi_C, \xi_E) = (2.8, 1.7)$, see Fig. 3.6. The objective is achieved because all receiver requirements are satisfied and the excess wireless resource is allocated to the receiver with the highest capacity, i.e. Carol. We observed that the wireless medium was fully utilized. In the second scenario, we reverse the requirements of Carol and Erin, $(\xi_C, \xi_E) = (1.7, 2.8)$, see Fig. 3.7. The theoretical total throughput is smaller in this case due to Erin's low physical rate and high requirement.

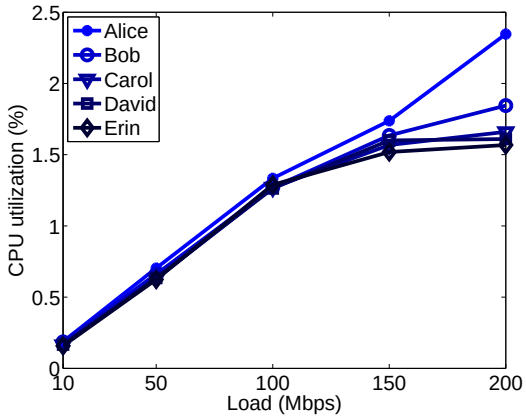


Figure 3.8: The CPU utilization of the five nodes of Fig. 3.5. The horizontal axis shows the traffic rate of the two sessions, A and B.

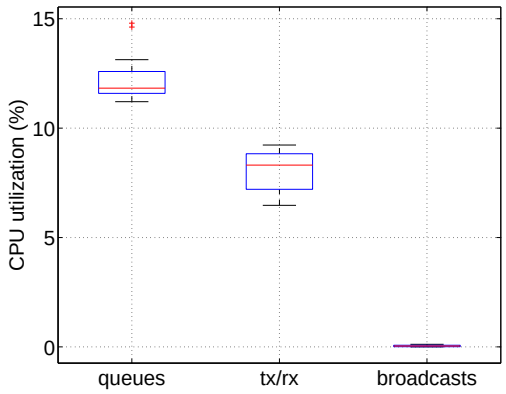


Figure 3.9: The percentage of the CPU occupancy of the main processes, queues maintenance, tx/rx operations and broadcast messaging.

3.5.5 CPU Occupancy

Our framework is implemented on user-level click. We observed the user-level CPU occupancy using the *sigar* library incorporated into the *click* framework, as well as the *valgrind* tool for analytical profiling. The CPU occupancy at every node remained below 3% even when we increased the traffic rate of the two sessions, A and B, up to 200Mb/s. We note, that a kernel-level implementation can improve this figure further. As we see in Fig. 3.8, the CPU usage remains the same for each node in most cases, indicating that our policy does not incur extra burden on the sources. A slight difference between the source and the other nodes is attributed mainly to high load and the fact that at optimality this specific node is charged to drop a lot of packets. However, we note that this difference is not comparable to the utilization variance caused by a congestion control based on source cooperation. Additionally, the CPU utilization was largely independent of data rates used, since it is

slightly increased by almost 2.5% when the traffic rate of the sessions is extremely increased up to 200Mb/s and more than 180Mb/s are dropped.

This implies that packet operations and queue maintenance have a minor contribution to the CPU occupancy. This is also illustrated in Fig. 3.9, where we see that the percentage of the CPU occupancy of *click* that occurs due to the queues maintenance, reception/transmission operations and message broadcasting is low enough (almost 15%). Instead most of the CPU occupancy is due to operations related to the underlying routing/forwarding scheme, which is only used as a demonstrator vehicle here and it is not necessary for our scheme.

3.6 Chapter Conclusions & Future Work

We proposed a distributed control scheme that maximizes utility in multirate multicast. The performance is analyzed and shown to be near-optimal. Several enhancements of the policy are described including a priority rule for base layer packets, and a modification for 802.11 wireless devices. The scheme is implemented in a wireless testbed and its applicability is demonstrated. In future work, we plan to derive the optimal policy for general wireless networks and to experiment further in larger topologies, investigating delay and CPU occupancy.

Chapter 4

Distributed Load Shedding with Minimum Energy

THIS chapter proposes *distributed load shedding* policies for regulating excessive network load. Data packets are inserted into the network to be delivered to intended destinations. The intermediate network nodes may decide to forward or drop some packets depending on temporally available resources. It is possible for some packets to traverse several nodes in the network until they are finally dropped before reaching the destination, which exacerbates energy consumption. We define a multi-objective optimization problem where we aim to minimize the used energy subject to providing maximum sum throughput. For the case of single-path unicast sessions, we show that Energy-efficient Distributed Load Shedding (E-DLS), a simple dropping mechanism combined with pushback routing, solves this load shedding optimization. We implement E-DLS in a testbed and use the experiments to select policy parameter values that strike a good balance between energy and delay performance. We then propose heuristic extensions of E-DLS for the cases of multirate multicast and multipath routing, and showcase via testbed experiments their optimal performance.

4.1 Introduction

Networks employ congestion control in order to regulate the network load, prevent overload, and ultimately improve the Quality of Service delivered to users. In applications such as mobile ad hoc networks and datacenter networks, dynamic congestion controllers have been proposed to solve optimally the network utility maximization problem [24]. When the network resource allocation is decided by the backpressure mechanism [8], a simple queue-based congestion control mechanism can be implemented at every source [36].

More recently, it was shown that congestion control can also be performed without source cooperation, by a combination of routing, packet dropping, and receiver congestion feedback [37]. Data packets enter the network without regulation, while the network reacts by shedding excess load to maximize the utility of the ultimately received traffic. In such a distributed load shedding approach,

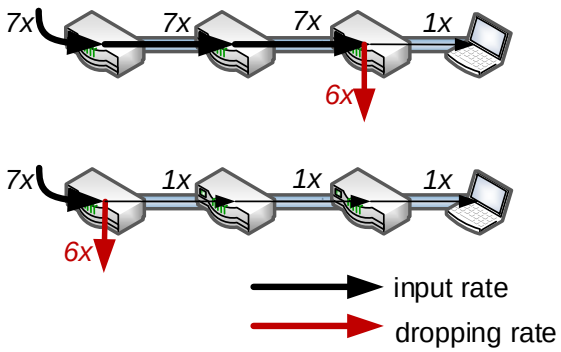


Figure 4.1: An example of an overloaded network where distributed load shedding (top) may consume more energy than source-based congestion control (bottom). We propose a novel load shedding mechanism which minimizes energy usage by adjusting real-time dropping decisions.

the congestion control happens *inside* the network, and there are no specified nodes responsible for performing it. This approach is desirable for many applications since it provides network protection from misbehavior or malicious sources, and simplifies the congestion control decisions in a large network. Load shedding has been successfully applied to data stream systems such as very large databases and information collection and computation systems [38].

Although the load shedding approach yields the same throughput as the source-based congestion control, it may lead to excessive energy consumption. Some packets may traverse several network nodes before they are eventually dropped, and thus the energy used for their transmissions is wasted. In the example of Fig. 4.1, under the threshold dropping mechanism of [37] (shown on top), the excess load is shed at the last node. This requires 15 transmissions per packet delivered on average, as opposed to the source-based congestion control solution (bottom) which only requires 3 transmissions. Since modern network devices turn on/off the network interface cards to save power [39], the consumed energy is roughly proportional to the time the cards need to be active. In an ad hoc network, nodes often spend most of their energy on communication [40]. Hence for this example, the system consumes roughly five times more energy when shedding the load in a distributed way, which is a heavy price for implementing load shedding in a distributed way. In this chapter we propose a novel distributed load shedding scheme which minimizes the energy expenditure subject to achieving maximum sum throughput.

We study a multi-objective load shedding optimization problem, where we aim to minimize the energy usage measured in number of transmissions, subject to maintaining maximum sum throughput. First, we focus on single-path unicast sessions and show that E-DLS, a simple distributed dropping policy combined with pushback routing, approximately solves the above optimization. Similarly to [37], E-DLS performs threshold-based dropping, but in our case the threshold at a node depends on its hop count distance from the destination. We implement E-DLS in a testbed, and demonstrate by experiments how the internal parameters affect the energy usage and the average delay.

E-DLS has a plethora of attractive features. It adapts to changing conditions, such as arrival rate

and channel quality. It can be implemented in a distributed fashion, it is highly scalable since it relieves sources from heavy computations, and it yields deterministic bounds for the queue backlogs facilitating implementation on systems with finite buffers, and providing delay guarantees. Since every network node is capable of shedding excessive load, E-DLS provides a natural shielding against overload.

We extend our study to multirate multicast and multipath routing, and propose heuristic extensions for E-DLS in these cases. These two communication scenarios underline the significance of our approach. In multirate multicast it is known that source-based congestion control is insufficient for maximum performance. In multipath routing, energy efficient solutions cannot be achieved by source-based congestion control unless the backpressure is altered as in [41].

Our contributions can be summarized as follows:

- We formulate a multi-objective load shedding optimization problem whose solution yields minimum number of transmissions subject to maximum sum throughput.
- We prove that a simple online dropping mechanism combined with pushback routing approximates the above offline optimization. Unlike prior work [37], this approach does not use virtual queues and thus straightforward to implement in any real system.
- We implement our proposed policy in a testbed and study how energy-delay tradeoffs depend on policy parameters.
- We propose heuristic extensions for multirate multicast, and multipath routing.

4.1.1 Related Work

In recent years, extensive research has been devoted to the challenge of network congestion control. The idea behind congestion control is to adjust appropriately the flow rates of a network, in order to prevent network overload and avoid the underutilization of the network resources. The common practice is to formulate an optimization problem that maximizes the sum of the source rates, where each source rate is the throughput of a flow specified by a source-destination pair. The optimal network control policy is revealed as the algorithm that solves the optimization problem. Moreover, enhanced works have elaborated the congestion problem and proposed algorithms that focus on the maximization of the sum of utilities (e.g. see [36]), where the utility is a function of the source rate. The latter problem is also named as per-receiver network utility maximization problem.

In contrast to the previously referred works, the authors in [37] consider the limitations of the source-based congestion control and propose a completely different approach. The receiver-based congestion control does not require the knowledge of arrival rates and therefore is robust to time-varying arrival rates that can go far beyond the network's stability region. It works with finite-size buffers and its performance gap from the optimal solution goes to zero as buffer sizes increase. Moreover, it is more sustainable to distributed denial of service attacks, in which an attacker creates a large number of flows with different source addresses to overwhelm prominent websites.

In addition to the work in [37], the authors in [2] extended the receiver-based congestion control with multirate multicast support. They present multiple algorithms, where each of them aims at the solution of a different optimization problem. Their first scheme succeeds in maximizing the total throughput received by all receivers of all streams, while the second scheme takes into account the utility of the receivers and maximizes the sum utility. The advantage of these approaches is that they are based on local dropping and do not require the end-to-end signaling of the source-based rate control, that would not support the multirate delivery. Source-based rate control algorithms are able to support multirate streaming by balancing the stream across a selection of multicast trees, increasing in this way the complexity and the buffering requirements of the network.

Our approach differs because it is based on the distributed load shedding imposed by the receiver-based congestion control, however, it minimizes the energy usage. An outline of the chapter is as follows. The network model is given in Section 4.2. We formulate the problem and introduces a threshold-based packet dropping policy that minimizes the energy consumption under maximized sum throughput in Section 4.3. Experimentation results that demonstrate the near-optimal performance of our policy is given in Section 4.4, while Sections 4.5 and 4.6 propose and experimentally evaluate heuristic solutions for multirate multicast and multipath routing.

4.2 Network Model

We consider a network with prespecified single-path routing and multiple unicast sessions, which are defined as source-destination pairs. In Sections 4.5-4.6 we extend our model for multicast, and multipath routing.

Our network consists of nodes in the set $\mathcal{N}$ and directional links in the set $\mathcal{L}$, where each link $(n,m) \in \mathcal{L}$ represents a communication channel between nodes n,m. In the analysis we consider a wireline network in which all links in $\mathcal{L}$ can be used simultaneously. Packets belong to a set of sessions $\mathcal{C}$, where session $c \in \mathcal{C}$ packets are generated at the source s_c and follow a specified path $P^{(c)}$ to the destination δ_c. We assume slotted time; in slot t, $A^{(c)}(t)$ session c packets are generated at s_c, where $A^{(c)}(t)$ are random, bounded above by $A_{\max}$, and i.i.d. over time with mean $\lambda^{(c)}$.

With the exception of destinations, each node n maintains a queue for every session c, whose backlog at the beginning of slot t is denoted by $Q_n^{(c)}(t)$. Let $(\cdot)^+ \triangleq \max(\cdot,0)$, we may express the evolution of $Q_n^{(c)}(t)$ as

$$Q_n^{(c)}(t+1) \leq \left[\left(Q_n^{(c)}(t) - \mu_n^{(c)}(t)\right)^+ - d_n^{(c)}(t)\right]^+ + \mu_{p(n)}^{(c)}(t),$$
$$\forall n \in P^{(c)} \cap \mathcal{N} \setminus \{\delta_c\}, c \in \mathcal{C}, \tag{4.1}$$

where $\mu_n^{(c)}(t)$ is the number of packets to be routed from n to the next node on path $P^{(c)}$, $d_n^{(c)}(t)$ is the number of packets to be dropped, and $p(n)$ is the incoming neighbor node (parent) of n. Hence, $\mu_{p(n)}^{(c)}(t)$ is an upper bound to the number of session c packets arriving at n. In the special case where $n = s_c$, we have by convention that $\mu_{p(n)}^{(c)}(t) = A^{(c)}(t)$. Moreover, for the simplicity of our notation,

we assume that $Q_{\delta_c}(t) = 0$.

In slot t, our queueing mechanism at node n works as follows. Newly arrived session c packets are added to backlog $Q_n^{(c)}(t)$. A distributed control policy chooses the values of the decision variables $\mu_n^{(c)}(t)$ and $d_n^{(c)}(t)$ for all sessions. For a decision to be admissible, the capacity constraint over link (n,m) must be satisfied

$$\sum_{c\in\mathcal{C}:(n,m)\in P^{(c)}} \mu_n^{(c)}(t) \leq R_{nm}, \quad \forall (n,m) \in \mathcal{L},$$

where R_{nm} is the capacity of link (n,m), and the summation is over all sessions that may transmit over link (n,m). For technical reasons, we limit the maximum number of permissible drops to $d_n^{(c)}(t) \leq d_{\max}$, where $d_{\max}$ is a system-defined parameter. Throughout the chapter, we assume $d_{\max} \geq A_{\max} + \max_{(n,m)} R_{nm}$. Since $A_{\max} + \max_{(n,m)} R_{nm}$ is an upper bound to the incoming data rate to a node, our choice of $d_{\max}$ ensures that the packet dropping rate is large enough to allow queue stabilization.

The purpose of this chapter is to propose distributed policies for determining the variables $\mu_n^{(c)}(t)$ and $d_n^{(c)}(t)$ at every slot, so that the long-term average performance of the system, as measured by throughput and energy metrics, is optimized.

4.3 Minimizing Energy Consumption under Maximized sum Throughput

4.3.1 Flow-Level Analysis

To design an optimal online policy, it is useful to first characterize the packet-level system performance. We define $q_n^{(c)}$ to be the rate of dropped packets of session c at node n, and $f_n^{(c)}$ the rate of forwarded packets to the outgoing link. The rate of received session c packets by destination δ_c is called *throughput*, and it is denoted by $r^{(c)}$. Since packets follow a single path, we have

$$r^{(c)} = f_{p(\delta_c)}^{(c)}, \quad \forall c \in \mathcal{C}. \tag{4.2}$$

For an arrival rate vector $\boldsymbol{\lambda} = (\lambda^{(c)})$, we say that a throughput vector is feasible if and only if there exist non-negative variables $f_n^{(c)}$, $q_n^{(c)}$ such that conditions (4.3)-(4.4) are satisfied:

$$f_{p(n)}^{(c)} = q_n^{(c)} + f_n^{(c)}, \quad \forall n \in P^{(c)} \cap \mathcal{N} \setminus \{\delta_c\}, c \in \mathcal{C}, \tag{4.3}$$

$$\sum_{c:(n,m)\in P^{(c)}} f_n^{(c)} \leq R_{nm}, \quad \forall (n,m) \in \mathcal{L}, \tag{4.4}$$

where by convention $f_{p(s_c)} = \lambda^{(c)}$. Equations (4.3) capture flow conservation and inequalities (4.4) capture capacity constraints. Let $\Lambda_{\boldsymbol{\lambda}}$ be the set of all feasible throughput vectors $\boldsymbol{r} = (r^{(c)})$. We have $\Lambda_{\boldsymbol{\lambda}} = \{\boldsymbol{r} \mid (4.2) \text{ - } (4.4)\}$. We define T^* to be the maximum sum throughput of all sessions, which is given by the following problem.

Sum Throughput Maximization:

$$T^* \triangleq \max_{\boldsymbol{r} \in \Lambda_{\boldsymbol{\lambda}}} \sum_{c \in \mathcal{C}} r^{(c)}. \tag{4.5}$$

Summing up (4.2) over all sessions and using (4.3) we see that the sum throughput is equal to the total arrived traffic less the sum of dropping rates

$$\sum_c r^{(c)} = \sum_c \lambda^{(c)} - \sum_{c,n} q_n^{(c)}, \tag{4.6}$$

where the index in the summations hereinafter is a shorthand form of $c \in C$ and $n \in P^{(c)} \cap \mathcal{N} \setminus \{\delta_c\}$. Therefore, for any fixed $\boldsymbol{\lambda}$, problem (4.5) is equivalent to

Sum Dropping Minimization:

$$Q^* \triangleq \min_{\boldsymbol{f},\boldsymbol{q} \geq \mathbf{0}} \sum_{c,n} q_n^{(c)}, \text{ subject to (4.3) - (4.4)},$$

where $\boldsymbol{f} = (f_n^{(c)})$ and $\boldsymbol{q} = (q_n^{(c)})$ are the flow and the dropping rates respectively.

4.3.2 Problem Statement

As illustrated in Fig. 4.1 using flows, problem (4.5) admits multiple optimal solutions, where each solution has potentially different performance with respect to the required number of transmissions and hence energy used. Motivated by this, we focus on *minimizing the average number of transmissions required to achieve maximum sum throughput*. Since the most energy-expensive process for a device is often the transmission/reception process, we effectively minimize the average consumed energy.

Minimizing the average number of transmissions can be achieved by minimizing the total flow across all links $\sum_{c,n} f_n^{(c)}$. Therefore our multi-objective can be expressed in flow-level by the following optimization.

Energy Minimization s.t. Maximum Sum Throughput:

$$E^* = \min_{\boldsymbol{f},\boldsymbol{q} \geq \mathbf{0}} \sum_{c,n} f_n^{(c)}, \text{ s.t. } \sum_c r^{(c)} = T^* \text{ and (4.2) - (4.4)}, \tag{4.7}$$

where E^* is the minimum energy measured in terms of rate of transmissions and T^* is the maximum value of (4.5).

Next, we exploit the assumption of single-path routing to simplify the problem and pose it in terms of dropping rates. Denote with $\mathcal{U}^{(c)}(n) \subset P^{(c)}$ the set of upstream nodes of n on path $P^{(c)}$, including n. Then, using the flow conservation constraints (4.3) we have

$$\sum_{c,n} f_n^{(c)} = \sum_{c,n} \left(\lambda^{(c)} - \sum_{k \in \mathcal{U}(n)} q_k^{(c)} \right). \tag{4.8}$$

Let $h_n^{(c)}$ be the hopcount from node n to the destination δ_c, e.g. $h_{p(\delta_c)}^{(c)} = 1$. By enumerating links in

two different ways we have the following identity

$$\sum_{n}\sum_{k\in\mathcal{U}(n)} q_k^{(c)} = \sum_{n} h_n^{(c)} q_n^{(c)}, \quad \forall c \in \mathcal{C},$$

which combined with (4.8) leads to

$$\sum_{c,n} f_n^{(c)} = \sum_{c} N^{(c)} \lambda^{(c)} - \sum_{c,n} h_n^{(c)} q_n^{(c)}, \tag{4.9}$$

where $N^{(c)} \triangleq |P^{(c)} \cap \mathcal{N}| - 1$ is the number of nodes on the path $P^{(c)}$ except for the destination. Since the arrival term above is constant and using (4.6), we conclude that in the specific case of multiple unicasts with single path routing, (4.7) is equivalent to:

Maximum Weighted Dropping s.t. Minimum Sum Dropping:

$$\max_{\boldsymbol{f},\boldsymbol{q}\geq\mathbf{0}} \sum_{c,n} h_n^{(c)} q_n^{(c)}, \text{ s.t. } \sum_{c,n} q_n^{(c)} = Q^* \text{ and (4.3) - (4.4).} \tag{4.10}$$

We further relax the minimum sum dropping constraint of (4.10) to obtain

$$\max_{\boldsymbol{f},\boldsymbol{q}\geq\mathbf{0}} \left(\sum_{c,n} h_n^{(c)} q_n^{(c)} - K \left(\sum_{c,n} q_n^{(c)} - Q^* \right) \right), \text{s.t. (4.3) - (4.4),} \tag{4.11}$$

where $K > \max_{(c,n)} h_n^{(c)}$ is a positive parameter. Consider the positive weights $\alpha_n^{(c)}(K) \triangleq K - h_n^{(c)}$ defined for nodes on paths $P^{(c)}$. We obtain the equivalent problem

Weighted Dropping Minimization:

$$\min_{\boldsymbol{f},\boldsymbol{q}\geq\mathbf{0}} \sum_{c,n} \alpha_n^{(c)}(K) q_n^{(c)}, \text{ s.t. (4.3) - (4.4).} \tag{4.12}$$

By letting $K \to \infty$, any solution of (4.12) converges to some solution of (4.10) (see the *penalty method* in [35, §4.2]). This also provides an optimal solution for our desired problem (4.7), i.e., it minimizes the energy used to achieve maximum sum throughput. Summarizing our flow-level analysis, we conclude that the "energy minimization subject to maximum sum throughput" problem of interest can be turned into a weighted dropping minimization problem for the case of single path unicast. In the following section we use the queue dropping analysis to design an online policy.

4.3.3 Packet-Level Control

We develop an online control policy that chooses instantaneous routing $\mu_n^{(c)}(t)$ and dropping $d_n^{(c)}(t)$ to yield a good time average performance. Below we make precise what we mean by "good time average performance".

We denote $\tilde{\mu}_n^{(c)}(t)$ the actual number of packets transmitted from node n along the path $P^{(c)}$ in slot

t, under a chosen policy. Note that $\tilde{\mu}_n^{(c)}(t)$ depends on backlog availability, and thus it is random; we have $\tilde{\mu}_n^{(c)}(t) \leq \mu_n^{(c)}(t)$. In the following we study the time averages of $\tilde{\mu}_n^{(c)}(t),\ \forall n \in P^{(c)} \setminus \delta_c,\ c \in \mathcal{C}$, and we assume that under our policy they converge. We define the average sum throughput of our scheme as

$$\hat{T} \triangleq \lim_{\tau \to \infty} \frac{1}{\tau} \sum_{t=0}^{\tau-1} \sum_{c} \mathbb{E}[\tilde{\mu}_{p(\delta_c)}^{(c)}(t)]. \tag{4.13}$$

In (4.13) we take the average of the expected received packets at all receivers $\sum_c \mathbb{E}[\tilde{\mu}_{p(\delta_c)}^{(c)}(t)]$ over a time period of τ slots and then let τ grow large. In a similar fashion, the average number of total transmitted packets in the network is

$$\hat{E} \triangleq \lim_{\tau \to \infty} \frac{1}{\tau} \sum_{t=0}^{\tau-1} \sum_{c,n} \mathbb{E}[\tilde{\mu}_n^{(c)}(t)]. \tag{4.14}$$

We wish to develop a control policy that drives these limits to be equal to the flow-level variables T^*, E^* from (4.7).

We will use the drift-plus-penalty method from the theory of Stochastic Optimization[26]. This is a technique used for stabilizing a queueing network while also minimizing the time average penalty function. In every slot t, the queue backlogs are observed and control actions are taken to greedily minimize a bound on the drift-plus-penalty expression

$$\Delta(t) + V\pi(t).$$

The drift $\Delta(t) \triangleq \mathbb{E}[L(t+1) - L(t)|\mathbf{Q}(t)]$ is determined using a quadratic Lyapunov function $L(t) = \frac{1}{2}\sum_{c,n}[Q_n^{(c)}(t)]^2$, which is a scalar measure of the queue backlog vector $\mathbf{Q}(t) = (Q_n^{(c)})$. Because of the squares, minimizing the drift requires reducing the large backlogs. For the penalty function, driven from (4.12) we choose

$$\pi(t) = \sum_{c,n} \alpha_n^{(c)}(K)\mathbb{E}[d_n^{(c)}(t)|\mathbf{Q}(t)].$$

Observe that the two goals are conflicting. We can choose a large $d_n^{(c)}(t)$ to drop many packets and reduce the backlog $Q_n^{(c)}$ hence reducing $\Delta(t)$, but this increases the penalty $\pi(t)$, and vice versa. Last, the positive parameter V controls the preference between reducing backlogs or reducing dropping penalties. We investigate the role of V in detail in Section 4.4.

Performing a standard derivation [10, 26], it can be shown that the drift-plus-penalty expression is bounded by the following quantity (B is a positive constant related to network capacities and char-

E-DLS **Energy-efficient Distributed Load Shedding**

Pushback Routing: Let $\mathcal{C}_{nm} \subseteq \mathcal{C}$ be the set of sessions that compete for service on each link $(n,m) \in \mathcal{L}$. Choose $c^*_{nm} \in arg\max_{c \in \mathcal{C}_{nm}}(Q_n^{(c)}(t) - Q_m^{(c)}(t))$ (ties broken arbitrarily). Over link (n,m) route

$$\mu_n^{(c^*_{nm})}(t) = \begin{cases} R_{nm} & \text{if } Q_n^{(c^*_{nm})}(t) - Q_m^{(c^*_{nm})}(t) > 0, \\ 0 & \text{otherwise} \end{cases} \tag{4.16}$$

and $\mu_n^{(c)}(t) = 0$ for all the other sessions $c \in \mathcal{C}_{nm} \setminus \{c^*_{nm}\}$, where R_{nm} is the capacity of the link.
Packet Dropping: For each session $c \in \mathcal{C}$ and node $n \in P^{(c)} \cap \mathcal{N} \setminus \{\delta_c\}$ choose

$$d_n^{(c)}(t) = \begin{cases} d_{\max} & \text{if } Q_n^{(c)}(t) > \alpha_n^{(c)}(K)V, \\ 0 & \text{otherwise.} \end{cases} \tag{4.17}$$

Above, $\alpha_n^{(c)}(K) = K - h_n^{(c)}$, where $h_n^{(c)}$ is the hopcount from n to the destination δ_c.
Parameter Selection: $d_{\max}, V, K$ are given in Table 4.1.

acteristics of the arrival process)

$$\begin{aligned} \Delta(t) + V\pi(t) \leq B + \sum_c Q_{s_c}^{(c)}(t)\lambda^{(c)} \\ \underbrace{- \sum_{c,n} \left(Q_n^{(c)}(t) - \alpha_n^{(c)}(K)V \right) \mathbb{E}[d_n^{(c)}(t)|\mathbf{Q}(t)]}_{\text{dropping}} \\ \underbrace{- \sum_{c,(n,m)\in P^{(c)}} \left(Q_n^{(c)}(t) - Q_m^{(c)}(t) \right) \mathbb{E}[\mu_n^{(c)}(t)|\mathbf{Q}(t)]}_{\text{routing}}, \end{aligned} \tag{4.15}$$

since

$$\begin{aligned} \sum_{c,n} \left(\mu_{p(n)}^{(c)}(t) - \mu_n^{(c)}(t) \right) Q_n^{(c)}(t) = \\ \sum_c \left(Q_{s_c}^{(c)}(t) A^{(c)}(t) - \sum_{(n,m)\in P^{(c)}} \left(Q_n^{(c)}(t) - Q_m^{(c)}(t) \right) \mu_n^{(c)}(t) \right). \end{aligned}$$

Observe that the Right-Hand Side (RHS) of (4.15) has two controllable terms noted with under brackets, one term controlled by the dropping variable $d_n^{(c)}(t)$ and one term controlled by the routing variable $\mu_n^{(c)}(t)$. We propose E-DLS, an online policy which observes the current queue state $\mathbf{Q}(t)$ and minimizes the RHS of (4.15) at every slot. The policy is presented below.

E-DLS operates in a distributed manner, using only locally available information. For the computation of (4.16) we require knowledge of the neighbor backlogs. It is straightforward to collect this information using messaging that incurs minor throughput overhead [2]. Backlog information

Table 4.1: Parameter Selection for E-DLS.

Param.	Explanation	Suggested values
$\mu_n^{(c)}(t)$	routing control variable	$[0, R_{nm}]$
$d_n^{(c)}(t)$	dropping control variable	$[0, d_{\max}]$
R_{nm}	capacity of link (n,m)	problem defined
$h_n^{(c)}$	hop count from destination along $P^{(c)}$	problem defined
$d_{\max}$	dropping batch size	$\geq A_{\max} + \max_{(n,m)} R_{nm}$
K	weight of throughput maximization	$\geq \max_{(n,c)} h_n^{(c)} + 1$
V	energy/delay tradeoff	> 0

obtained in this way is often delayed, however prior findings show that this does not hurt throughput optimality [10, §4.7]. The computation of (4.17) is done locally at each node.

To save energy, E-DLS drops all unnecessary packets near the sources, hence with respect to vector $\boldsymbol{q}$ the obtained solution resembles the one from source-based congestion control. However, E-DLS additionally protects the network from denial of service attacks, or situations where a source is malfunctioning. By performing congestion control everywhere, the network is inherently shielded from overload. Moreover, as shown in Section 4.5, the load shedding solution for multicast sessions differs fundamentally from the source-based congestion control.

4.3.4 Performance Evaluation of E-DLS

The performance of E-DLS is characterized by the following theorem.

Theorem 4.1 (Performance of E-DLS)**.** Consider a sequence of systems operating under E-DLS parameterized by K, V. Suppose that each system starts empty. For each system, denote the sum throughput achieved by E-DLS with $\hat{T}(K)$ (defined in (4.13)) and the consumed energy with $\hat{E}(K,V)$ (defined in (4.14)).

(1) **Bounded backlogs.** For any K, V we have

$$Q_n^{(c)}(t) \leq KV + d_{\max}, \quad \text{for all } c, n, t. \tag{4.18}$$

(2) **Near optimal throughput.** The sum throughput achieved satisfies

$$\lim_{K\to\infty} \hat{T}(K) = T^*, \tag{4.19}$$

where T^* is the maximum sum throughput from (4.5).

(3) **Near optimal number of transmissions.** The total energy consumed (measured in number of transmissions) satisfies

$$\lim_{K\to\infty} \lim_{V\to\infty} \hat{E}(K,V) = E^*,$$

where E^* is the minimum value of (4.7).

Proof of Theorem 4.1. The proof is in Appendix D. □

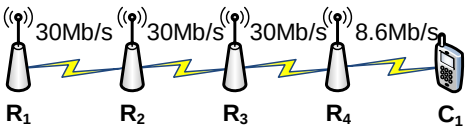
Theorem 4.1 shows that E-DLS approximately solves (4.7), i.e., achieves maximum sum throughput using the minimum required number of transmissions. As we see in the theorem, to accurately approximate the desired solution both K and V need to be chosen arbitrarily large. In practice we will choose finite values for K and V so that the solution is approximated in a satisfactory level but KV is kept as small as possible, which provides delay guarantees. We study this tradeoff in the next section.

4.4 Experimentation Results

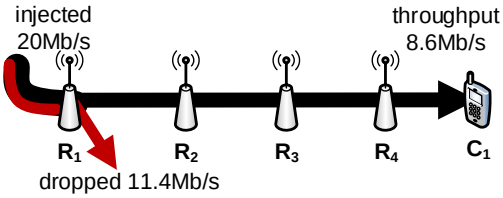
In this section we focus our study on a simple five-node network shown in Fig. 4.2(a). The labels show the names of the nodes and the capacities of the links. We conduct two sets of experiments, where in the first we inject 20Mb/s at node R_1 (4-hop topology shown) and in the second we inject the same load at node R_3 (2-hop topology, not shown). The data are always destined to client C_1. The experiments are run in the NITOS, where network nodes implement our proposed policy E-DLS. The details about our implementation are the same with these of the previous policies implementation, described in Subsection 3.5.1. The energy consumption measurement methodology is based on custom-built hardware that is developed and described in [42].

We note that all wireless links operate in 802.11a using non interfered frequency channels, since each node is equipped with two wireless interfaces, one used for the preceding and one for the following link on the path. As follows, the network model is the same as this of a wireline network, since the links are wireless but non-interfered and almost static (no mobility). To showcase our improvements, we compare our approach to an implemented version of the ORA policy from [37], which performs distributed load shedding for maximum sum throughput without any energy considerations. Fig. 4.2(b)-4.2(c) show the resulting queue dropping rates from the 4-hop set of experiments and clearly demonstrate the benefits of our proposed scheme. By shedding 11.4Mb/s early on, our scheme reduces the total transmissions from 68.6Mb/s to 34.4Mb/s, roughly reducing the energy spent in transmissions by half.

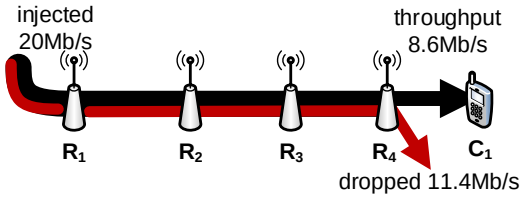
In the remaining section we study tradeoffs between energy, throughput and delay, that naturally arise in stochastic optimization [43]. Fig. 4.3 shows the measured energy, throughput, and delay performance of E-DLS compared to ORA, for both 2-hop (top figures) and 4-hop (bottom figures) cases, where we vary K, KV. For ORA we vary V and plot the corresponding point assuming $K = 1$. From Figures 4.3(a)-4.3(d) we observe that the throughput of E-DLS improves when K, or KV increase, as expected. The maximum throughput is 8.6Mb/s and it is achieved in all cases when $KV > 1500$. From Figures 4.3(b)-4.3(e) we derive a similar conclusion for energy performance: larger values for K and KV reduce energy. The minimum energy for the 2-hop topology measured in rate of transmissions and power consumption is 17.2Mb/s and 43.8W respectively, while for the 4-hop topology is 34.4Mb/s and 87.7W. ORA has a gap from the optimal energy, while E-DLS achieves it as K, KV increase. For small values of KV, E-DLS consumes less energy because most of the packets are dropped early and throughput is small. Increasing KV, first we achieve maximum throughput, and



(a) Experimentation setup

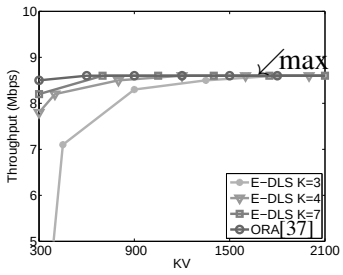


(b) E-DLS

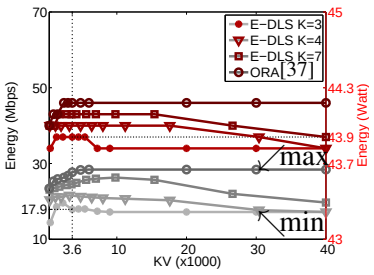


(c) ORA [37]

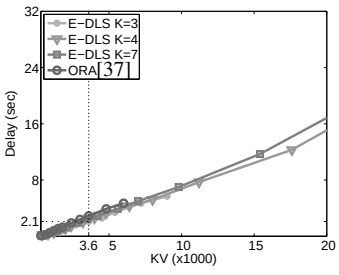
Figure 4.2: Single-path experimentation setup (a) and experimental results (b)-(c).



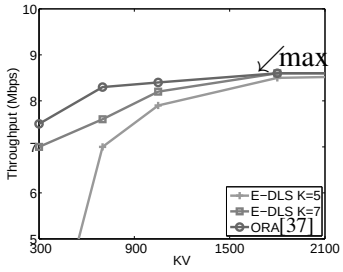
(a) Throughput over 2-hop path.



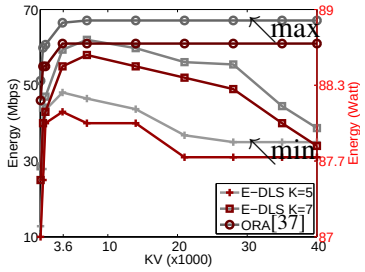
(b) Energy over 2-hop path.



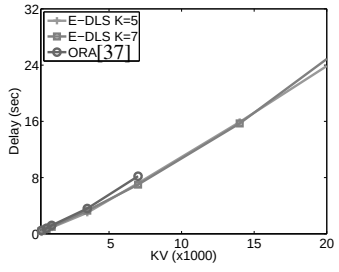
(c) Delay over 2-hop path.



(d) Throughput over 4-hop path.



(e) Energy over 4-hop path.



(f) Delay over 4-hop path.

Figure 4.3: Single-path experimentation results.

then energy consumption gradually reduces to minimum. Last from Figures 4.3(c)-4.3(f), we observe that the average delay increases linearly to KV.

Combining the observations from Fig. 4.3, the best choice is to keep K as small as possible and increase V. In particular, we choose $K = \max_{(n,c)} h_n^{(c)} + 1$, which is the smallest integer that satisfies the condition given in the explanation under (4.11). Then V must be chosen accordingly to yield a good tradeoff between energy and average delay. For example, in the 2-hop topology, a good choice

is $K = 3$ and $V = 1200$, which yields energy 17.9Mb/s and average delay 2.1sec, as it is illustrated in Figures 4.3(b) and 4.3(c) for $KV = 3600$. The parameter choices for E-DLS are summarized in Table 4.1. Moreover, the total power consumption in this case is reduced to 43.9W, that is 300mW less than the corresponding power consumption of ORA. As future work, we plan to research methods for reducing the backlog sizes, thus improving the experienced delay. Two promising methods are (i) the initialization of backlogs with low-priority virtual packets, and (ii) the use of learning methods from [44].

4.5 Multirate Multicast

In single-path unicast, the solution of our distributed load shedding optimization (4.7) with respect to the dropping vector $\boldsymbol{q}$ resembles the solution of source-based congestion control, although E-DLS solves (4.7) using a distributed scheme that operates in the entire network. In this section we consider multicast sessions delivered to receivers over multicast trees. Our objective remains to minimize energy subject to delivering maximum sum throughput to all receivers in the multicast group. We allow receivers to obtain the data stream at different rates, i.e., we consider multirate multicast. In multirate multicast, congestion control at the source is not efficient, [45], [2]. To maximize sum throughput, packets should be dropped at multiple nodes. In particular, each node should drop that many packets such that the remaining packet transmission rate is equal to the maximum reception rate by all receivers served through this node.

Multicast session c has a set of receivers $\Delta^{(c)}$ and utilizes a multicast tree with set of links $\mathcal{L}^{(c)}$. To capture multicast, a node with multiple outgoing links has multiple queues. For every link $l \in \mathcal{L}^{(c)}$, there is a queue backlog $Q_l^{(c)}(t)$ at the incident node holding packets for transmission over that link. Upon arriving to a node, packets are replicated to all queues $Q_l^{(c)}(t)$, where l are outgoing links from that node. The dropping rates at each queue may be different, denoted with $q_l^{(c)}$. Similarly, the data flow rate to each link is denoted with $f_l^{(c)}$. Following the same notation with before, the rate of received session c packets by all destinations $\Delta^{(c)}$ is expressed as $r^{(c)}$. The sum throughput received at all receivers of multicast sessions can be expressed in the same fashion with (4.6). The throughput of each session c receiver is the arrived traffic $\lambda^{(c)}$ less the dropping rates of the links connecting the source to this receiver. As follows, the sum throughput is the total injected rate times multicast group size less a weighted sum of the dropping rates, where the weight of each dropping rate is the number of the receivers that are affected by this. In particular,

$$\sum_c r^{(c)} = \underbrace{\sum_c \left|\Delta^{(c)}\right| \lambda^{(c)}}_{\substack{\text{total throughput} \\ \text{without dropping}}} - \underbrace{\sum_{c,l} m_l^{(c)} q_l^{(c)}}_{\substack{\text{total dropping rate} \\ \text{times affected receivers}}} , \tag{4.20}$$

where the index in the summations hereinafter is a shorthand form of $c \in \mathcal{C}$ and $l \in \mathcal{L}^{(c)}$, and $m_l^{(c)}$ is the number of receivers connected to source s_c via link l. When packets are dropped on this link, the throughput of all such receivers is reduced, hence the multiplier in (4.20). Moreover, the total number

multicast E-DLS

Pushback routing: Similarly to E-DLS, choose $c_l^* \in arg\max_{c\in\mathcal{C}_l}(Q_l^{(c)}(t) - \sum_{l':p(l')=l} Q_{l'}^{(c)}(t))$ (ties broken arbitrarily) and route

$$\mu_l^{(c_l^*)}(t) = \begin{cases} R_l & \text{if } Q_l^{(c)}(t) - \sum_{l':p(l')=l} Q_{l'}^{(c)}(t) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

$\mu_l^{(c)}(t) = 0$ for all the other sessions $c \in \mathcal{C}_l \setminus \{c_l^*\}$.
Packet Dropping per link: For each session $c \in \mathcal{C}$ and link $l \in \mathcal{L}^{(c)}$ choose

$$d_l^{(c)}(t) = \begin{cases} d_{\max} & \text{if } Q_l^{(c)}(t) > \beta_l^{(c)}(K)V \\ 0 & \text{otherwise.} \end{cases}$$

Above, $\beta_l^{(c)}(K) = m_l^{(c)}K - H_l^{(c)}$, where $m_l^{(c)}$ is the number of session c receivers downstream from link l and $H_l^{(c)}$ is the number of links in the subtree connected to s_c via l, including l.
Parameter Selection: $K \geq \lceil(\max_{c,l} H_l^{(c)} + 1)/m_l^{(c)}\rceil$ and $d_{\max}, V$ are given in Table 4.1.

of transmissions can be expressed in the same fashion with (4.9) as

$$\sum_{c,l} f_l^{(c)} = \underbrace{\sum_c \left|\mathcal{L}^{(c)}\right| \lambda^{(c)}}_{\substack{\text{total trans/sion rate} \\ \text{without dropping}}} - \underbrace{\sum_{c,l} H_l^{(c)} q_l^{(c)}}_{\substack{\text{total dropping rate} \\ \text{times affected links}}}, \tag{4.21}$$

where $H_l^{(c)}$ is the number of links in the subtree connected to the source via l (including l). When packets are dropped on l, the packet transmission rate of all the links in the remaining subtree is also reduced. Following the same approach of Section 4.3.2, by replacing the total packet transmission rate and throughput of (4.7) with the RHS of (4.20) and (4.21), we may express our objective of minimizing the total number of transmissions subject to maximum sum throughput as follows.

Weighted Dropping Minimization for Multicast:

$$\min_{\boldsymbol{f},\boldsymbol{q}\geq\mathbf{0}} \sum_{c,l} \beta_l^{(c)}(K) q_l^{(c)} \tag{4.22}$$

$$\text{s.t. } f_{p(l)}^{(c)} = q_l^{(c)} + f_l^{(c)}, \quad \forall l \in \mathcal{L}^{(c)}, c \in \mathcal{C}, \tag{4.23}$$

$$\sum_{c:l\in\mathcal{L}^{(c)}} f_l^{(c)} \leq R_l, \qquad \forall l \in \mathcal{L}^{(c)}, \tag{4.24}$$

where $p(l)$ is the preceding link of l in the multicast tree of session c, and by convention we set $f_{p(l)}^{(c)} = \lambda^{(c)}$ for l outgoing from source. Also R_l is the capacity of link l and we define $\beta_l^{(c)}(K) \triangleq m_l^{(c)}K - H_l^{(c)}$ for all c, l. Above, equations (4.23) are flow conservation constraints per link, and inequalities (4.24) are capacity constraints. We propose *multicast* E-DLS, that dynamically solves (4.22).

The online control policy of *multicast* E-DLS is derived from the same analysis of Section 4.3.3. The penalty function is now driven from (4.22), so we choose

$$\pi(t) = \sum_{c,l} \beta_l^{(c)}(K)\mathbb{E}[d_l^{(c)}(t)|\mathbf{Q}(t)],$$

where $d_l^{(c)}(t)$ is the number of packets to be dropped from $Q_l^{(c)}(t)$. Denoting the packet transmission rate over link l as $\mu_l^{(c)}(t)$ and assuming that $\mu_{p(l)}^{(c)}(t)$ are the source generated packets when l is an outgoing link of s_c, then each queue evolution is expressed as

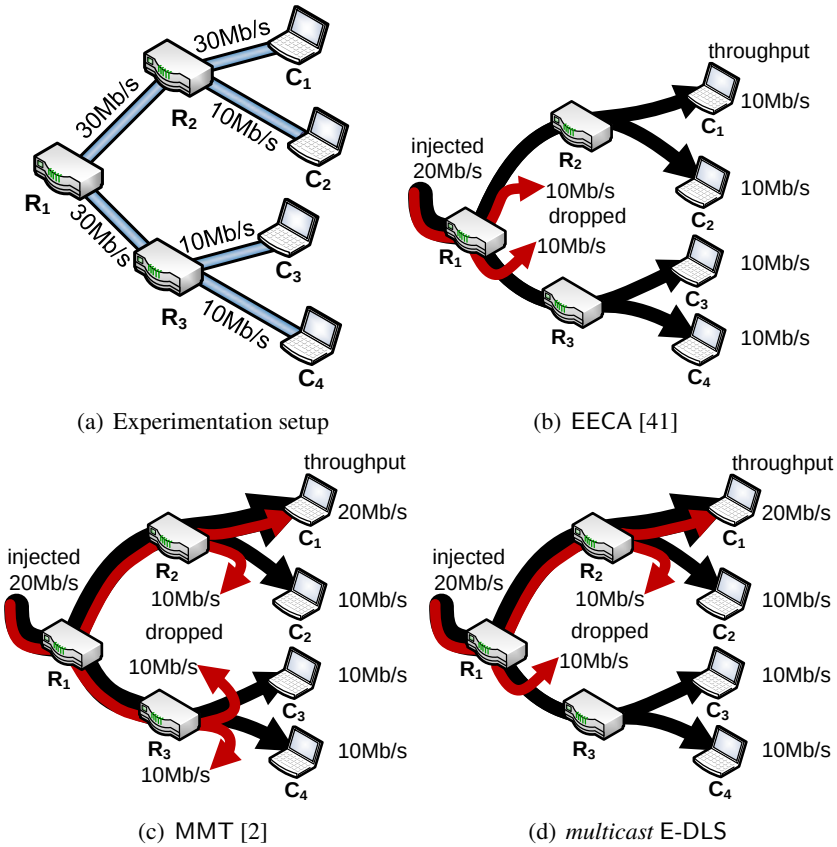
$$Q_l^{(c)}(t+1) \leq \left[\left(Q_l^{(c)}(t) - \mu_l^{(c)}(t)\right)^+ - d_l^{(c)}(t)\right]^+ + \mu_{p(l)}^{(c)}(t),$$
$$\forall l \in \mathcal{L}^{(c)}, c \in \mathcal{C}$$

and the drift-plus-penalty expressions is bounded by

$$\begin{aligned}\Delta(t) + V\pi(t) \leq B_2 &+ \sum_{c,l=(s_c,m)\in\mathcal{L}^{(c)}} Q_l^{(c)}(t)\lambda^{(c)} \\ &\underbrace{-\sum_{c,l}\left(Q_l^{(c)}(t) - \beta_l^{(c)}(K)V\right)\mathbb{E}[d_l^{(c)}(t)|\mathbf{Q}(t)]}_{\text{dropping}} \\ &\underbrace{-\sum_{c,l}\left(Q_l^{(c)}(t) - \sum_{l':p(l')=l} Q_{l'}^{(c)}(t)\right)\mathbb{E}[\mu_l^{(c)}(t)|\mathbf{Q}(t)]}_{\text{routing}},\end{aligned}$$

where *multicast* E-DLS obviously minimizes the RHS at every slot.

We experiment in NITOS with *multicast* E-DLS and we compare its performance to MMT [2] and EECA [41]. In Fig. 4.4(a) we consider a simple multicast tree topology that showcases the advantages of our distributed load shedding mechanism. First, observe that EECA (Fig. 4.4(b)), a source-based congestion control scheme designed for minimum energy in unicast, drops all excess load at the source, and hence its sum throughput is 40Mb/s, which is less that the theoretically maximum 50Mb/s. This demonstrates that in multirate multicast, a source-based solution is not efficient. Then, observe that the distributed load shedding policy MMT from [2] (Fig. 4.4(c)), which is the multicast extension of ORA. It achieves maximum sum throughput, but uses unnecessarily more energy than *multicast* E-DLS (Fig. 4.4(d)), since it carries excess traffic of 10Mb/s from R_1 to R_3 which is later dropped. In conclusion, the proposed multicast extension *multicast* E-DLS achieves maximum sum throughput using only the necessary average number of transmissions.



(a) Experimentation setup (b) EECA [41]

(c) MMT [2] (d) *multicast* E-DLS

Figure 4.4: Multicast tree setup (top-left) and experimental results (rest).

4.6 Multipath Routing

In previous sections we examined the case of fixed routing with single paths (unicast) or trees (multicast). In both cases the flow conservation constraints are simplified allowing rewriting our desired optimization as a weighted dropping minimization, which ultimately results in an optimal threshold-based dropping mechanism. For networks that permit multipath routing this approach is not feasible, hence it is not obvious whether a simple threshold-based mechanism can solve our multi-objective optimization problem.

To showcase the complications that multipath routing brings to our problem, we consider the toy network of Fig. 4.5(a) where a session sends 20Mb/s from R_1 to C_1, and it may use both available paths. The maximum throughput in this case is clearly restricted by link (R_3, C_1), and is equal to 10Mb/s. Clearly dropping 10Mb/s at R_1 is preferable here, since carrying more load inside the network will increase energy consumption without improving throughput. Indeed, experimenting with E-DLS results in the outcome of Fig.4.5(b) which achieves maximum throughput and drops unnecessary packets at the source. However, this solution does not minimize energy in this case. To minimize energy, all 10Mb/s should be routed through the path that does not involve R_2, since this path requires one transmission per delivered packet, while the other path requires two transmissions per delivered

multipath E-DLS

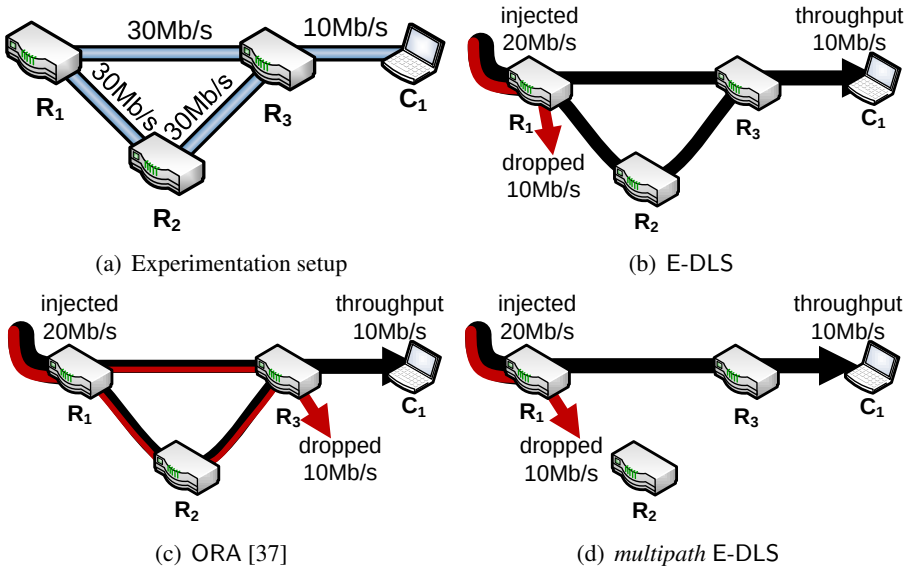
Pushback routing with threshold γ: Over link (n,m) choose $c^*_{nm} \in arg\max_{c\in\mathcal{C}_{nm}}(Q^{(c)}_n(t) - Q^{(c)}_m(t))$ (ties broken arbitrarily) and route

$$\mu^{(c^*_{nm})}_n(t) = \begin{cases} R_{nm} & \text{if } Q^{(c^*_{nm})}_n(t) - Q^{(c^*_{nm})}_m(t) > \gamma, \\ 0 & \text{otherwise.} \end{cases}$$

$\mu^{(c)}_n(t) = 0$ for all the other sessions $c \in \mathcal{C}_{nm} \setminus \{c^*_{nm}\}$.

Packet Dropping: Same as in E-DLS, where $h^{(c)}_n$ is the hop count from n to destination along the shortest path connecting them.

Parameter Selection: $0 < \gamma < \min_c \alpha^{(c)}_{s_c}(K)V$ and $d_{\max}$, V, K are given in Table 4.1.



(a) Experimentation setup (b) E-DLS

(c) ORA [37] (d) *multipath* E-DLS

Figure 4.5: Multipath setup (top-left) and experimental results (rest).

packet and hence consumes more energy. The root cause of our problem in this case is pushback routing, which balances load over all paths and hence consumes more energy in comparison to shortest path routing.

To tackle this case, we consider prior work that proposes EECA to minimize energy by means of source-based congestion control [41]. In particular, EECA uses the framework of [36] for scheduling, routing, and source-based congestion control, and in addition introduces a threshold γ, performing backpressure routing only when the differential backlog exceeds γ. At a high level, this threshold causes the backlogs to increase proportionally to the hop count from the destination, making longer paths less desirable, and gradually leading most of the traffic through shortest paths, reducing the energy consumption.

Using intuition from [41], we propose a new heuristic extension to E-DLS for mutlipath routing,

where we use a threshold γ for pushback routing, similar to EECA. From experiments we observe that our heuristic effectively routes the whole traffic through the shortest path. As we see in Fig. 4.5(d), from the injected 20Mb/s, 10Mb/s is dropped early in the source and the remaining 10Mbp/s is routed through the most energy efficient path. In conclusion, the proposed multipath extension *multipath* E-DLS achieves minimum energy consumption dropping early the unnecessary packets and forwarding the others through the shortest path to their destination.

4.7 Chapter Conclusions & Future Work

We study distributed load shedding in a network, which amounts to initially admitting all extraneous load and gradually shedding part of the load at individual nodes in a distributed fashion. It is observed that such a methodology may lead to excessive energy consumption, since some packets may be forwarded several hops before they are dropped. We formulate a multi-objective load shedding optimization problem where we seek to minimize the energy used to achieve maximum sum throughput. We prove that a simple distributed online policy, using threshold-based dropping and pushback routing, solves this problem for the case of single-path unicast sessions.

Then we propose heuristic extensions for multirate multicast and multipath routing. Experiments in NITOS showcase that our policies achieve our objectives. However, our future work is to prove that the proposed policy succeeds in solving optimally the multirate multicast or mutlipath routing cases as well. Moreover, we plan to further extend the proposed policy towards the maximization of the sum utility, instead of the sum throughput, assuming that each destination is assigned a potentially different utility function. Last but not least, we are willing also to include network modeling for multihop wireless with links operating on the same channel.

Chapter 5

Video-aware Multicast Opportunistic Routing over 802.11 two-hop mesh networks

In this chapter, we propose and characterize the performance of a novel video-aware Opportunistic Routing (OR) algorithm for multicast, using a one-hop and two-hop forwarding scheme in 802.11 mesh networks. We believe that the OR approach exploits the inherent broadcast nature of the wireless medium and adapts very well in the lossy wireless environment. Inferring from the above, we extend a state of the art routing algorithm, namely MORE, that leverages on this approach and offers multicast support, but is not efficiently applicable in video streaming. By employing our scheme, we enable support for video streaming application with hard time-constraints. In addition, we focus on the orchestration of the packet transmissions and the prioritization of the video traffic towards improving the video-perception quality of the end users. In order to evaluate the proposed scheme, we conducted experiments in a realistic medium-scale wireless testbed. Our results show that the proposed scheme increases the average video-perception quality, measured in PSNR, by up to 270% in average, compared to the MORE algorithm.

5.1 Introduction

As the need for Internet access has grown enormously nowadays, wireless connectivity seems to be the most appropriate solution for low-cost and efficient network coverage. Deployment of wireless networks is more affordable compared to the past and the available speeds are now up to 600Mbps (802.11n), rendering the wireless access as the most appropriate option for physical interconnection. Although the wired access seems to be more stable and of high-bandwidth, it is unsuitable in cases of public areas where mobile devices opportunistically arrive or leave. The recent proliferation of these devices (laptops, tablets, smartphones) has been instrumental in bridging even more the development of the wireless access to the attention of networking researchers. There are multiple open issues and

challenges in order to stream efficiently over the wireless medium. For example, the broadcast nature of the wireless medium should be exploited in cases that devices receive the same multicast stream.

Representative examples involve large scale events in public areas (audio concerts, football matches, airports, etc.), where the majority of the requests is for the same multicast and real-time stream. Although the wireless access can backhaul this kind of scenarios, the deployment of many wireless gateways in order to cover the whole area is often impossible [46]. The exploitation of easily placed wireless relays could fill in the gap between the gateways' coverage subareas. Moreover, the collaborative processing and retransmission of overheard information at some end devices, could also make them play the role of wireless relays and create spatial diversity and throughput improvement for all devices. In this case, the routing from the gateway to each end device is the most challenging issue.

ExOR [47] is the first Opportunistic Routing (OR) protocol for wireless mesh networks that takes advantage of the wireless broadcast nature and does not follow the traditional routing approach of choosing the best sequence of relays between the gateway and each device. It creates cooperative diversity, leveraging broadcast transmissions in order to send information through multiple relays concurrently. MORE [48] is an enhanced version of the ExOR protocol, supporting also multicast traffic and utilizing Network Coding (NC) [49] to improve throughput up to three times. Moreover, MORE is a MAC independent protocol as compared to ExOR, running directly on top of 802.11 CSMA/CA instead of the strict scheduler that ExOR deals with. Both protocols are mostly UDP compliant routing protocols, since ExOR needs to be better integrated with TCP and MORE supports multicast streaming that is only implemented over UDP.

In this work we extend and fine tune the work made in MORE in order to meet and satisfy the necessities and requirements of video multicast. It is worth mentioning that *video traffic should be delivered without delay, even if this implies that some information may get lost*. Inferring from the above, we introduce the Video-aware Multicast Opportunistic Routing protocol (ViMOR), focusing on topologies where the end devices are one-hop or two-hop away from a gateway. In particular, our augmentation is threefold: i) support for time-constrained forwarding process, ii) enhancements to the transmission policy regarding the selection of the relays and the orchestration of the transmissions of the gateways and the relays and iii) video-perception quality improvement by classifying and prioritizing the video traffic. In contrast to MORE, ViMOR addresses the demanding video challenges, enjoys high throughput performance and increases the quality of the video perception in all end devices of each multicast group.

The rest of the chapter is organized as follows. In Section 5.2 we introduce related work. Section 5.3 introduces OR concepts and provides the design and the keystones of the proposed scheme. In Section 5.4 we evaluate the performance of the proposed protocol by conducting appropriate experiments in a wireless testbed. We conclude in Section 5.5.

5.2 Related Work

ExOR [47] was proposed by Biswas and Morris, introducing the OR approach. OR belongs to a general class of wireless algorithms that exploit the broadcast nature of the wireless transmission, utilizing the overheard information at multiple nodes to increase wireless throughput. These algorithms could either relay the received signal acting as a multi-antenna system, or combine the bits received at different nodes to correct wireless transmission's errors [50], or optimize the choice of the next relay from the nodes that received a transmission. ExOR belongs to the third category and was the first OR implementation that demonstrated cases, where the more relaxed choice of next-hop achieves significant throughput gains. More specifically, in ExOR the source separates the packets in batches in order to send them collectively. Then, it does not try to send the packets of each batch to a specific next-hop host (expecting for an acknowledgment), but broadcasts the packets for a specific number of retries, and each potential receiver also retransmits them for a specified number of times, until the destination finally receives the whole batch and sends a batch acknowledgment. The scheduling of the transmissions among the source and the potential relays is based on a modified MAC layer, that specifies the intervals when nodes send their packets avoiding contentions/collisions.

MORE [48] is the enhanced version of ExOR, introducing a NC approach that randomly mixes packets before forwarding them. The source and the relays do not forward the identical packets of the batch, but linear combinations with arbitrary multipliers of the native packets. The newly generated packets have the corresponding multipliers encapsulated in a specific header, thus reproducing the native packets in destinations is feasible by executing the inverse process. The scheduling of the transmissions of all involved nodes is arbitrary, based on the 802.11 CSMA/CA, making the protocol MAC-independent and directly applied. However, even under the impact of the resultant contentions/collisions, the throughput performance of MORE is significantly better than that of ExOR. It is also worth to mention that MORE, in the same way that ExOR does, enforces the source and the relays to retransmit until the destination successfully sends an acknowledgment. The main difference with ExOR is that MORE imposes the source to transmit continuously, while each potential relay has a credit value, which is the number of transmissions that it will attempt for each received packet. Finally, the architecture of MORE makes the multicast case a natural extension of the unicast one, in contrast to ExOR that supports only unicast.

Some open issues and weaknesses of the MORE protocol regarding the multicast case have been addressed in some other works [51, 52, 53]. For example, in MORE the source requires an acknowledgment from each destination of the multicast group before proceeding to the next batch, resulting in performance degradation in cases of one or more receivers having poor connections. CodeOR [51] and Pacifier [53] addressed this weakness of MORE and suggested solutions to improve it. For example, CodeOR allows the source to move to the next batch before receiving acknowledgment for the current batch, using a sliding window of multiple batches. Similarly, Pacifier adopts a round-robin mechanism that enables the source to move to the next batch every time that one receiver acknowledges the current batch. After proceeding with a predefined number of batches, the source will repeat the transmission process for each of the previous batches to finally receive acknowledgments from

all multicast destinations. These are very interesting approaches, since they suppress the annoying variation on the batch forwarding duration of MORE. However, they do not succeed in eliminating this phenomenon, since they target again at 100% reliable forwarding. To the best of our knowledge, ViMOR is the first scheme that introduces the total rejection of the acknowledgment mechanism, redesigning appropriately the transmission policy and enabling a time-constrained forwarding process.

OR-PLC [54] is another work that focuses on video traffic, enabling the partial reproduction of a batch, when the full reproduction is not feasible yet. Instead of using the Random Linear Coding (RLC) of MORE and Pacifier, this work introduces a Priority (or progressive) Linear Coding (PLC) to mitigate the error propagation and provide high bandwidth utility. More specifically, with OR-PLC the source generates some network coded packets as a linear combination of only the most important native packets, that correspond to video intra-frames. The intra-frames are encoded by only removing spatial redundancy in the frame, while inter-frames are encoded by removing temporal redundancy in successive frames. The loss of an intra-frame is much more crucial than the loss of an inter-frame, since the intra-frame is also required for decoding all successive frames. PLC enables the earlier retrieval of the intra-frames comparing to RLC, even if some inter-frames get lost, provisioning for at least a low quality video sequence to a poorly connected destination. However, OR-PLC adopts the same acknowledgment mechanism with MORE. ViMOR implements PLC and evaluates its efficiency, when it is activated in parallel with the aforementioned video-aware extensions.

5.3 ViMOR Design

In this work, an innovative and enhanced multicast OR protocol is proposed, extending the MORE philosophy to adapt to the video traffic requirements. Video multicast streaming uses UDP that is unreliable comparing to TCP. We strongly believe that OR integrates well with the video traffic requirements and the high probability of packet loss in wireless networks, since in case of video streaming, delivering on-time is of greater importance than delivering reliably. In case of traditional routing with path selection, the duration of each wireless transmission cannot be easily estimated, since the occasional but not rare variations of channel conditions may cause an unknown number of MAC retransmissions, until the MAC acknowledgment is successfully received. Subsequently, the time of a packet forwarding process through a specific route is unpredictable and may exceed the time constraints of a specific video sequence, since it is equal to the aggregate duration of the individual time-varying transmissions.

On the other hand, in case of OR, the transmissions are broadcasted without MAC retransmissions and acknowledgments, enabling the duration of the packet forwarding process to be upper limited, depending only on the controlled number of transmissions that the source and each relay attempt. Subsequently, OR does not provide reliability in packet delivery, since there are no MAC acknowledgments, but as we already mentioned this is of less importance in case of video streaming. It is worth mentioning that some OR algorithms, like MORE, implement an application layer acknowledgment mechanism to provide reliability in cost of their capability for time constrained streaming,

coping with similar inconvenience with the traditional routing.

Based on the aforementioned analysis, we propose a new OR protocol based on the design of MORE, named Video-aware Multicast Opportunistic Routing (ViMOR), and we summarize its main differences as compared to the MORE protocol:

- **Rejection of the acknowledgment mechanism** since the video traffic should be delivered on-time and not necessarily reliably.
- **Redesign of the transmission policy** concerning the scheduling and the number of transmissions that the source and the relays perform.
- **Classification and prioritization of the video packets** according to their content, adopting an enhanced NC policy.

Due to the first two differenciations, ViMOR achieves high throughput video streaming, satisfying the video requirement for maximum time duration of packet forwarding process and giving more transmission opportunities to the most error-susceptible wireless links. The third one improves even more the video streaming performance by enhancing the quality of the delivered video, increasing the probability of successful delivery of the most important video packets.

As depicted in Figure 5.1, ViMOR focuses on multicast scenarios where **all destinations are at most two-hop away from the source**. The rationale behind this decision is twofold: i) the performance of video wireless streaming over paths of three or more hops is degraded due to the fluctuations that increase as the paths get longer, and ii) the source is not able to apply the transmission policy if it serves more than two-hop away destinations, since it requires the link evaluations that should be on-line and updated. At this point, it is useful to mention that MORE supports broader topologies, however, based on off-line link evaluations that were collected in the past [52]. It is infeasible for one central point to gather on-line measurements in these broader topologies with more than two-hop away destinations. This feature of MORE's design is not desirable, since studies have shown that link metrics are sensitive and should be frequently updated [55].

On the other hand, a mechanism inspired by the ETX estimation algorithm of Roofnet [18] is able to provide on-line link evaluations for the aforementioned topologies of our focus. More specifically, this mechanism enforces nodes to periodically send broadcast packets, estimate the number of the corresponding received packets from each neighbor and report these numbers among them. Through this process, each node calculates the transmission error probabilities of its adjacent links, while a periodical report informs its neighbors about these evaluations. At the end, every node (including the source node that applies the transmission policy) knows the quality of its adjacent links and its neighbors' adjacent links. It is worth mentioning that the flooding mechanism and the statistics from previous packets going to the reverse direction, which are used by a Roofnet node to evaluate the links that are more than two-hop away, are impracticable for every multicast and single-source algorithm, like MORE and ViMOR.

Before proceeding, we introduce some notations further explaining the key points of the NC policy adopted by both ViMOR and MORE. They are summarized in Table 5.1, together with all the

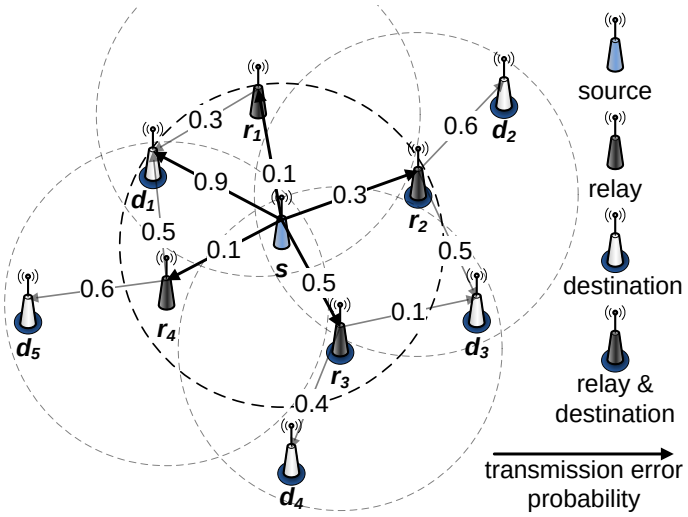


Figure 5.1: A topology where all destinations are at most two-hop away from source.

Table 5.1: Variables description for ViMOR.

Variable	Description
s, $\mathcal{R}$, $\mathcal{D}$	source and sets of relays and destinations respectively
N	total number of nodes
k	number of native packets included in a batch
b	packet size (payload and headers)
ρ	utilized basic physical transmission rate
f, g	video frame ratio and number of GOP frames respectively
l	packets needed for a GOP transmission
τ	time given for the forwarding of one batch (slot)
c	total number of transmissions in a slot (total credit)
c_1, c_2	credits of source and each relay respectively
E	average prob/ty of unsuccessful packet delivery among all destinations $d \in \mathcal{D}$
$\mathcal{O}$	packet classes with different priority
k_o	number of class $o \in \mathcal{O}$ packets in a batch
o_h, o_l	high and low priority classes including intra-frames and all frames respectively
α	the intra-frames size proportion of the whole batch size

other notations that will be introduced later. Regarding a single multicast stream, consider a source s that is supported by a set $\mathcal{R}$ of R relays and serves a set $\mathcal{D}$ of D destinations. The network consists of $N = |\mathcal{R} \cup \mathcal{D}| + 1$ nodes. In both routing schemes, source s breaks up the stream to *batches* of k equal-sized packets of size b. Each time the source forwards a batch, it generates and transmits broadcast packets that are linear combinations of the k native batch packets. The coefficients of each linear combination are encapsulated to the corresponding generated packet. Once a relay $r \in \mathcal{R}$ receives a packet, it linearly combines this packet with the previously received ones of the same batch and forwards the generated packet for transmission. When a destination $d \in \mathcal{D}$ receives k linearly independent packets, it is able to decode the batch and retrieve the k native packets. Both source and relays utilize the basic/lowest physical rate ρ for all packet transmissions, in order to extend as much as possible their coverage areas.

The following Subsections 5.3.1, 5.3.2 and 5.3.3 will explain further the outlines of ViMOR differentiation, as compared to MORE.

5.3.1 Rejection of the acknowledgment mechanism

In MORE, an acknowledgment mechanism gives a signal to the source for the expiration of the batch forwarding process and the initiation of a new one. More specifically, during the batch forwarding process, the source and the relays generate and transmit packets continuously and for an unlimited number of times, until source receives an application layer acknowledgment from each of the involved destinations. It is obvious that this mechanism cannot provide any guarantee for maximum time duration of the multicast batch forwarding.

On the other hand, ViMOR overcomes this challenge by enforcing the source and the relays to transmit for a fixed number of times. The source does not wait for an acknowledgment, but keeps a timer and the batch forwarding is limited within a specified time period, called *slot*. The slot duration is estimated by the source, according to the video stream characteristics. Assuming that a video stream features a frame ratio f, a Group Of Pictures (GOP) with g frames should be delivered in a time interval equal to g/f. So, if a GOP needs l packets or l/k batches to be encapsulated, then the forwarding of one batch should be completed during a slot $\tau = (g/f)/(l/k) = gk/fl$. After the expiration of the slot interval, the source proceeds to the next batch.

The slotted mechanism does not provide reliability in batch forwarding, but every batch that is successfully delivered, is always on-time. As we have already mentioned before, this is a desirable feature, since it is a waste of time and energy for source and relays to keep forwarding a batch, that is already obsolete and useless for the destinations.

5.3.2 Redesign of the transmission policy

The second most important difference in ViMOR's approach is the enhanced transmission policy. In MORE, as it is already mentioned in Section 5.2, source generates and transmits packets continuously and for unlimited number of times before proceeding to the following batch. Once a node receives a packet, it generates and transmits a number of new packets equal to its assigned *credit*, which is estimated by taking into account the quality of all network links. Each node that is "charged" with a non-zero credit is a potential relay. In ViMOR, the credit of a node is interpreted in a different way, representing the number of packet transmissions this node will attempt during a batch forwarding, independent of the number of the received packets. The aggregate credit of source and relays is upper bounded by a c integer value that depends on the utilized slot τ, since the number of transmissions that can be performed in a slot interval is obviously limited by $c < \rho\tau/b$. Source initially estimates the value of $c = \lfloor \rho\tau/b \rfloor$ based on the known parameters ρ, τ and b. Later, it estimates again a more accurate value of c, based on the history of the transmissions happened at the past slots. More specifically, it estimates the new value of c as the sum of the 80% of the old value and the 20% of the number of the transmissions happened in the last slot. Outside interference and unmodelled factors in wireless transmissions are the main reasons for the c variation.

Actually, ViMOR adopts a new transmission policy that is presented below and aims at increasing the individual throughput of each one-hop or two-hop away destination host, **maximizing the average**

Algorithm 1 Computing c_1, where $E(x)$ is the probability E for $c_1 = x$ and $c_2 = (c-x)/R$. The algorithm tries to find the maximum of the convex function $E(x)$ by successively narrowing the initial range $[k, c-Rk]$ of value x by golden ratio ϕ. It converges when the length of the range including the solution is bounded by 1.

$y_l \leftarrow k$
$y_r \leftarrow c - Rk$
$\phi \leftarrow (\sqrt{5}-1)/2$
$x_l \leftarrow y_l + (1-\phi)(y_r - y_l)$
$x_r \leftarrow y_l + \phi(y_r - y_l)$
for $|y_l - y_r| > 1$ **do**
  if $E(x_l) > E(x_r)$ **then**
    $y_r \leftarrow x_r$
    $x_r \leftarrow x_l$
    $x_l \leftarrow y_l + (1-\phi)(y_r - y_l)$
  else
    $y_l \leftarrow x_l$
    $x_l \leftarrow x_r$
    $x_r \leftarrow y_l + \phi(y_r - y_l)$
  end if
end for
$c_1 \leftarrow \lceil \arg\max_{x \in \{x_l, x_r\}} E(x) \rceil$

probability of successful batch reception among all destinations. From now on, assume that E is this probability. The challenges appear in i) selecting the most appropriate one-hop relays; and ii) charging source and these relays with suitable credits. Regarding the relays selection, the source can either choose them or utilize a fixed and dedicated set of relays. In case that the set of relays is not fixed and predefined, $\mathcal{R}$ is retrieved by the source building a multicast tree that connects the source to all two-hop away destinations. The tree is similar to that of Pacifier and is a shortest-ETX tree, constructed at the source by taking account of the union of all shortest-ETX paths to the two-hop away destinations. At the end, the set of relays $\mathcal{R}$ consists of all the one-hop connected nodes to the source, belonging in this tree.

To overcome the second challenge, regarding the credit charging, we need to address two orthogonal sub-challenges. The first has to do with providing the source with the highest possible credit, equal to $c_1 \geq k$, in order to satisfy all one-hop away destinations, while the second aims at sharing appropriately the credit c among source and relays, providing also a credit $c_2 \geq k$ to each relay, in a way that satisfies the two-hop away destinations. Source and relays need at least k transmissions to forward k independent packets which is the minimum required for the batch decoding. We also choose to share the same credit among the relays, following the same approach with other works [56] and enabling the estimation of the two variables with low-computational cost. As follows, $c_1 + Rc_2 = c$ for avoidance of slot violation or underutilization, thus $c_1 \in \{k, k+1, ..., c-Rk\}$ [1]. The balance between these two sub-challenges is related to the aforementioned system objective.

[1] If $k > c - Rk$, then we satisfy only the one-hop away destinations giving all credit to $c_1 = c$.

In order to satisfy this objective, the source shares the total credit c in a way that maximizes the aimed probability, however, for a packet and not for a batch. This is an approximation followed also by MORE. The c_1 and c_2 credits are retrieved at the source by applying the "Golden section" search of Algorithm 1, since probability E is a convex function of c_1. Appendix E proofs that E is a convex function of c_1. The source knows the transmission error probabilities of all links due to the ETX estimation mechanism, described in detail before. Moreover, the relays learn the c_2 value by the source through the periodical broadcasts, which are used for the estimation of the transmission error probabilities. The complexity of this algorithm is $O((R+1)D\log c)$ (the estimation of E requires at most $(R+1)D$ calculations for a specified couple of c_1 and c_2 values), while the complexity of the corresponding algorithm of MORE is $O(DN^2)$ for the case of multicast forwarding [48]. It is worth mentioning that R is limited, since in most cases there is no need for more than 4 or 5 relays supporting the two-hop away destinations. Moreover, for large values of c, the algorithm converges rapidly in less iterations than $\log c$, since the maximum value of E is close to 1 and is given for many c_1 values. Subsequently, the complexity of this algorithm is apparently better than this of MORE.

Our experimentation shows that this transmission policy outperforms the behavior of MORE by giving more transmission opportunities over the lowest quality links. Actually in MORE, the source does not stop transmitting and competing with the one-hop relays for the medium access during the whole period of a batch forwarding. This approach results to equal transmission opportunities among the source and its one-hop relays, regardless of the links quality and the corresponding MORE's credit assignment, since the CSMA/CA mechanism of 802.11 statistically distributes equally the channel access among the competing transmitters. In ViMOR, the contentions/collisions are reduced by enforcing relays to apply the *first-decode-then-transmit* policy. When applying this policy, the relays are imposed to start forwarding a batch only after the successful decoding of this batch and the retrieval of the corresponding k native packets, thus the contentions/collisions are reduced. This policy is also applied for a second reason; the relays should not spend transmission opportunities of the source for transmission of packets, which are not linear combinations of all k native packets and thus contain less information.

5.3.3 Classification and prioritization of the video packets

The last differenciation of ViMOR is the implementation of NC using Priority Linear Coding (PLC), which classifies the packets to $\mathcal{O}$ priority classes and replaces the Random Linear Coding (RLC) of MORE. Our scheme focuses on video streaming, which inherently consists of packets of varied significance. For example, the packets that include segments of the intra-encoded frames (I-frames) are more important than the packets that include segments of the inter-decoded ones (P-frames and B-frames). The latter P/B-frames cannot be decoded without having the corresponding I-frame. In ViMOR, we define priority classes of packets, where each class $o \in \mathcal{O}$ contains the k_o most important packets of a batch. More specifically, we utilize a high and a low priority o_h and o_l class respectively. The o_h packets include the segments of the intra-frames, having always $k_{o_h} = \alpha k$, while the o_l class contains all batch packets and $k_{o_l} = k$. We assume that the intra-frames of a batch need a proportion of

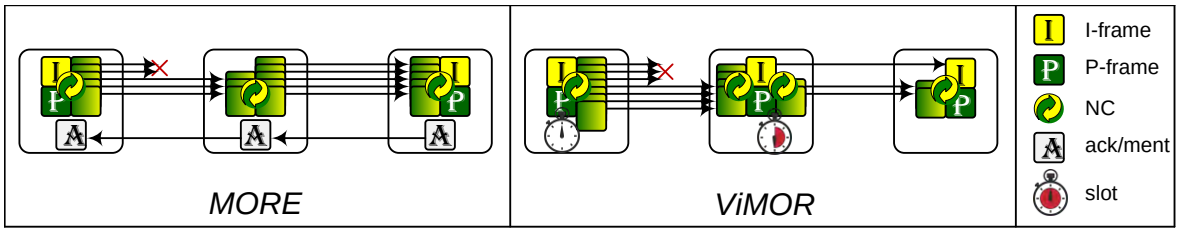


Figure 5.2: The main differences between MORE and ViMOR. In MORE, the source and the relay are competing for the medium during the whole period of the batch forwarding, which ends when the source receives the acknowlegment. The source has the same transmission opportunities with the relay, although it is more susceptible to transmission errors and needs more. ViMOR replaces the acknowlegment with a time counter (gaining the time spent for the acknowlegment forwarding). It gives more transmission opportunities to the source and disables the collisions between the source and the relay by applying the *first-decode-then-transmit* policy. Finally, the relay uses PLC enabling the forwarding of the I-frame even if the P-frame is lost.

the whole batch size less or equal to α. The credit of each relay is shared proportionally to each class. The o_h packets take a proportion α of the whole credit c_2, and the o_l packets take the whole credit. This means that each relay generates and transmits the first αc_2 packets as linear combinations of the most important o_h packets, while the rest ones are linear combinations of all packets. The source does not change its behavior, doing the same as with the RLC mechanism.

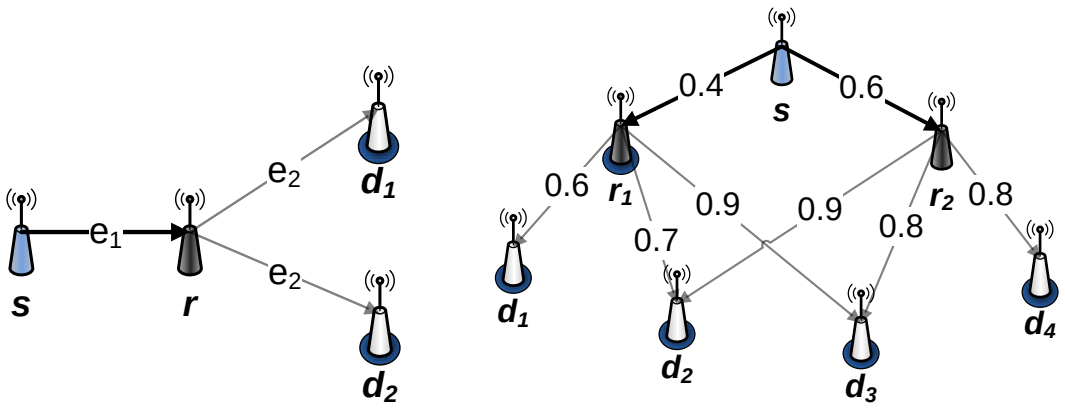
The receiver performs two parallel decoding processes; the first one is fed with the packets generated from the coding of the o_h packets, while the second one is fed with all received packets. The two decoding processes are executed simultaneously, hence enabling the successful decoding of the o_h packets with higher or equal probability. Even if the decoding of the whole batch is infeasible, a receiver may be capable to decode the most important packets of this batch. This enables the reception of a video sequence of tolerable quality, in case that the reception of a high quality video is infeasible.

Figure 5.2 summarizes and depicts the main differences between MORE and ViMOR in a representative example with a source, one relay and one destination.

5.4 Experimentation Results

The implementation of ViMOR routing scheme is based on the Click framework [7], which offers easy to develop, flexible and configurable modular routers. Click modular router is comprised from packet processing modules called elements, that implement simple router functions. In this work, we extend and modify the Click based implementation of the MORE routing algorithm, introducing the aforementioned contributions for video streaming.

The deployment and evaluation of ViMOR took place at the NITOS testbed [6], where we conducted experiments under various topologies with specific features. In our experimentation, we chose 802.11g. Obviously, our architecture can be directly applied to the next wireless access technologies such as 802.11n/ac/ad, without requiring modifications or influencing in any other way the perfor-



(a) $\mathcal{R} = \{r\}, \mathcal{D} = \{d_1, d_2\}$ (b) $\mathcal{R} = \{r_1, r_2\}, \mathcal{D} = \{r_1, d_1, d_2, d_3, d_4\}$

Figure 5.3: Two different topologies with 4-nodes (a) and 7-nodes (b) used in our algorithm evaluation.

mance comparison between MORE and ViMOR. The specifications of the NITOS nodes used for the experiments are depicted in Table 2.1.

The thorough evaluation of ViMOR required the experimentation under different topologies with several connectivity conditions. Since it is impossible to find the desired conditions in a testbed with stationary nodes, we reproduced them with the use of a distributed packet filtering mechanism, that we further explain. More particularly, we selected NITOS nodes that are close to each other, shaping a full mesh connected topology with robust links (transmission error probabilities very close to zero). Then, we applied a packet filter to each one of these nodes, allowing a received packet to pass through with a specific probability, according to the transmitter's identifier. This mechanism enabled the full control of the connectivity map, providing us with the ability to replicate any lossy link. The topologies of our experimental setups are illustrated in Figure 5.3. Each link represents a communication channel for direct transmission from a given node to another one, and is labeled by its corresponding transmission error rate.

There are also three other important issues that we had to cope with in our implementation. Firstly, we modified the code of MORE in order to forward video packets pushed from the upper layer, instead of using dummy packets created by MORE. Secondly, we added the ETX broadcast mechanism of Roofnet, in order to use updated link evaluations during our experimentation, enabling the source to run periodically Algorithm 1 and recompute the credits. Last but not least, we utilized the queue notification scheme of Click in order to reduce the number of empty pull requests, a.k.a. the unreplied requests made by the wireless device for coded packets to be transmitted. We notify the wireless device when there are no packets for transmission, avoiding the CPU-consuming process of the pull request, since otherwise the CPU utilization of Click was almost 100%.

5.4.1 First class of experiments

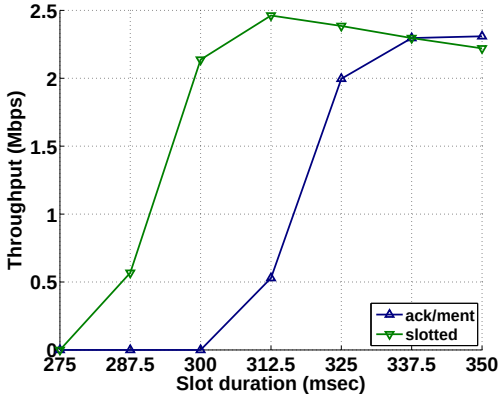
The first class of our experiments is conducted using the topology of Figure 5.3(a), where the source is s, $\mathcal{R} = \{r\}$ and $\mathcal{D} = \{d_1, d_2\}$, while the transmission error probabilities e_1 and e_2 are adjusted appropriately. The performance of both MORE and ViMOR is expected to be highly insensitive to different batch sizes ($k = 8, 16, 32, 64$), as it is presented in [48]. However, as we explain later and conclude in our experimentation, $k = 64$ seems to be the best choice for ViMOR. The main configuration parameters are that RTS/CTS is disabled, as it happens in most real networks, and all nodes use $\rho = 6$ Mbps as physical transmission rate. Finally we configure the packet payload to be equal to 1470 bytes. The packet size is $b = 1556 + k$ bytes, after adding the WiFi, IP and UDP headers/trailers, as well as the MORE header that is also adopted by ViMOR. The MORE header features $22 + k$ bytes length, where the k bytes are used for holding the coefficients that linear coding uses to generate the corresponding packet.

In the following lines, we present the evaluation of ViMOR. ViMOR's proposed contributions have been evaluated individually, conducting the three following separate sets of experiments in order to explore the individual benefits of each contribution. There is also an extra set of experiments for evaluating the CPU utilization of the source, relays and destination nodes.

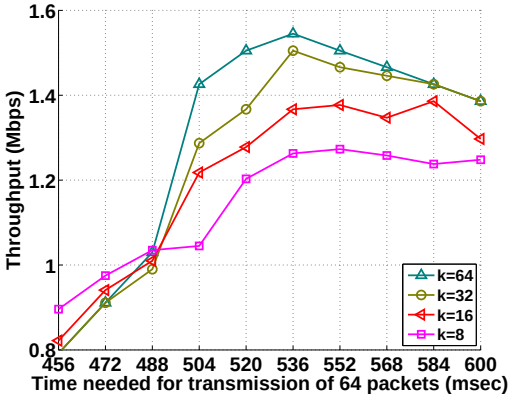
5.4.2 Slotted vs. acknowledgment mechanism

In the first set of experiments, the throughput performance of the proposed video-aware slotted mechanism of ViMOR (details in Subsection 5.3.1) is compared to the one of the acknowledgment mechanism of MORE. We perform the comparison using the first topology under transmission error probabilities close to zero, in particular $e_1 = e_2 \approx 0.001$, and $k = 64$, since this is the best value for k as we will see later. The performance of the slotted mechanism is quite insensitive to the k value in this experiment. When using MORE, the source transmits continuously, while the relay retransmits a specific number of packets for each one received. In our case, this number is equal to one. The source proceeds to the next batch after receiving an aggregate acknowledgment from both destinations. On the other hand, under the slotted mechanism of ViMOR, the source proceeds to the next batch after the expiration of the current slot, even if the destinations have not yet decoded the current batch.

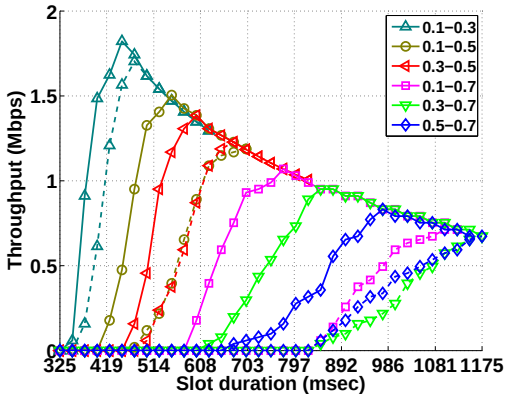
The plots in Figure 5.4(a) depict the average throughput of the on-time decoded packets between the two destinations for the two mechanisms. On-time decoded packets are only these that have been delivered in a time interval less than the slot duration τ. The traffic load sent from the source may be larger than the corresponding throughput, since it also includes packets that either got lost, as it happens in the slotted mechanism, or received too late, that happens in the acknowledgment mechanism. The horizontal axis represents the slot duration in milliseconds, while on the vertical axis we depict the measured throughput in Mbps. It is obvious that for long time slots the performance of the two mechanisms is similar, or the acknowledgment mechanism performs better, due to the underutilization of the wireless medium that the slotted mechanism imposes as the slot duration increases. Both mechanisms achieve to forward frames on-time, while the acknowledgment one succeeds in



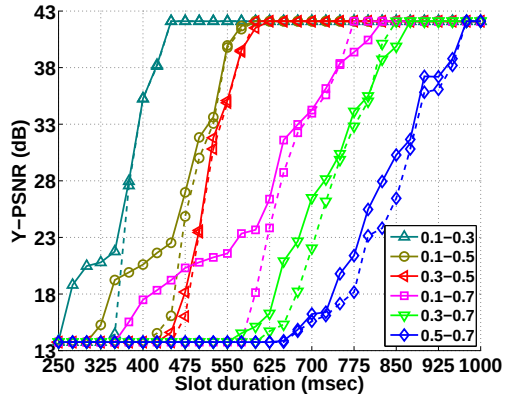
(a) The slotted mechanism of ViMOR compared to the acknowledgment mechanism of MORE for $k = 64$ and error transmission probabilities $e_1 = e_2 \approx 0.001$.



(b) The VIMOR's credit assignment performance for $k = 8, 16, 32, 64$ and error transmission probabilities $e_1 = 0.1$ and $e_2 = 0.5$.



(c) The ViMOR's credit assignment compared to the $50-50\%$ equally distributed credit assignment for $k = 64$ and multiple e_1 and e_2 probabilities. The marker of its line indicates the e_1 - e_2 or e_2 - e_1 values (same results). The solid lines correspond to the ViMOR's assignment and the dashed lines to the $50-50\%$ one.



(d) PLC compared to RLC for $k = 64$, $\alpha = 1/3$ and multiple e_1 and e_2 probabilities. The marker of its line indicates the e_1 - e_2 values. The solid lines correspond to PLC and the dashed lines to RLC.

Figure 5.4: Evaluating ViMOR in the 4-nodes experimentation topology of Figure 5.3(a).

pre-buffering more and more as the slot period increases. However, as the slot period decreases, it is evident that the proposed mechanism achieves a significant performance improvement, delivering video in cases that the acknowledgment mechanism is completely inefficient ($\tau \leq 300$ msecs). This is a remarkable result, since it enables transmission of higher quality video sequences, that feature high frame ratios (high f) or high definition frames (high l) and subsequently require low slot duration $\tau = gk/fl$.

5.4.3 Evaluation of the transmission policy

In the second set of experiments, we evaluate the proposed transmission policy by configuring the nodes connectivity and applying the suggested credit assignment mechanism of Subsection 5.3.2.

Table 5.2: Credits c as a percentage of slot duration τ for physical transmission rate $\rho = 6$ Mbps and $k = 8, 16, 32, 64$.

k	8	16	32	64
c	$\tau \cdot 45.1\%$	$\tau \cdot 44.9\%$	$\tau \cdot 44.5\%$	$\tau \cdot 43.7\%$

Initially, we configure the transmission error probabilities $e_1 = 0.1$ and $e_2 = 0.5$ for selecting the best k. The selection of these error rates is the result of extensive experimentation, where we have observed the largest differentiation in the performance of the proposed policy for multiple values of k. Figure 5.4(b) shows the performance of the proposed credit assignment for $k = 8, 16, 32, 64$. The horizontal axis represents the time needed in milliseconds for delivering a sequence of 64 packets, while on the vertical axis we depict the measured throughput in Mbps. The interval represented in horizontal axis is equal to $\tau \cdot 64/k$, where τ is the slot duration that a batch needs to be delivered. As it is clearly depicted, $k = 64$ is the best choice. Although $k = 64$ imposes the largest overhead in packet transmission, since it uses longer headers, it enables the most accurate estimation of the redundancy packets that a transmitter should use. Therefore, for the rest of the experiments presented, we use $k = 64$.

The next step is to configure the transmission error probabilities e_1 and e_2, using different pairs of probability values. Figure 5.4(c) shows the performance of the proposed credit assignment compared to the performance of a simple and equally distributed credit assignment ($50-50\%$), where $c_1 = c_2 = c/2$ independently of the e_1 and e_2 values. The horizontal axis represents the slot duration τ in milliseconds, while the vertical axis represents the achieved throughput in Mbps. The solid lines depict the throughput performance of the ViMOR policy, and the dashed lines the one of the equally distributed assignment policy. The c value depends on the slot duration τ, as we have already mentioned, and it is presented in Table 5.2. Each pair of same colored solid and dashed plots corresponds to a different couple of probability pairs $e_1 - e_2$. It is worth mentioning that both assignment policies succeed the same results if we swap the values of e_1 and e_2.

We compare the ViMOR policy with the $50-50\%$ one, because in MORE the transmission opportunities among source and one-hop relays are equally shared, due to the 802.11 MAC protocol. Subsequently, although MORE applies a more sophisticated credit assignment policy, the result is the same with applying the $50-50\%$ one. In ViMOR, the first-decode-then-transmit policy applies an indirect scheduling that reduces the contentions/collisions and allows a proportional sharing of the transmission opportunities. It is noticeable that the proposed policy succeeds in delivering higher throughput traffic in all the cases that $e_1 \neq e_2$. As it is expected, the throughput gain of the proposed policy is high in cases that $|e_1 - e_2|$ is large enough. Moreover, it is worth mentioning that the performance of the equally distributed credit assignment depends only on the lowest quality link, since it is the same for all probability pairs that feature the same $\min(e_1, e_2)$.

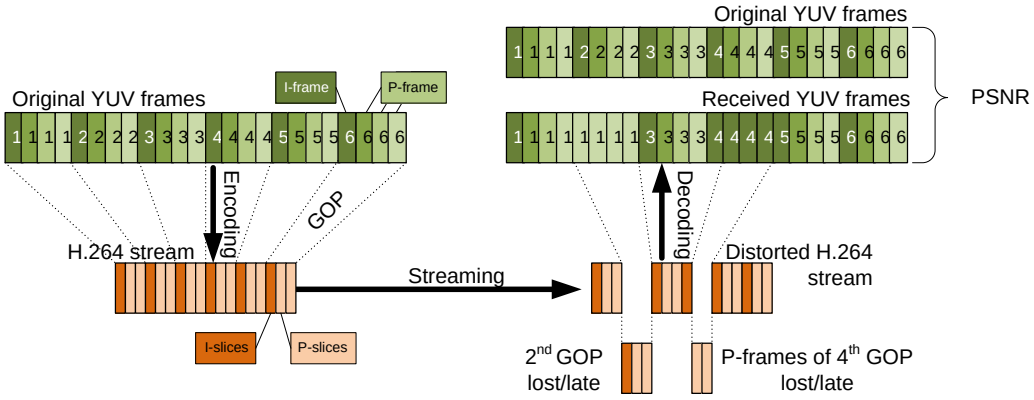


Figure 5.5: PSNR evaluation of the received video stream.

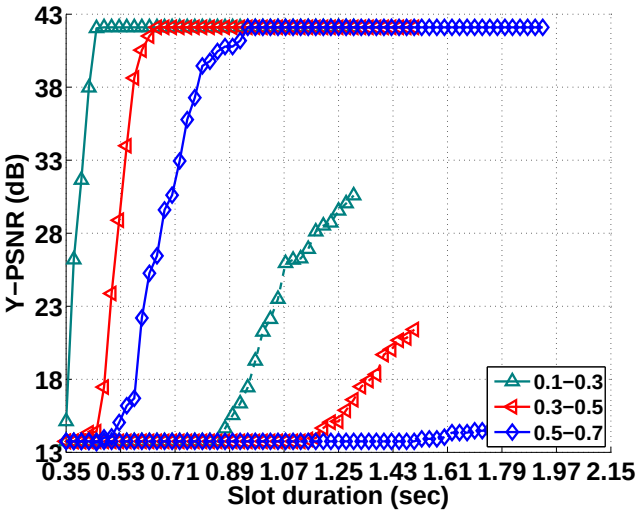


Figure 5.6: The *first-decode-then-transmit* policy compared to the *partial-decode-then-transmit* one, for $k = 64$, $\alpha = 1/3$ and multiple e_1 and e_2 probabilities. The marker of its line indicates the e_1 - e_2 values. The solid lines correspond to the first policy and the dashed lines to the second one.

5.4.4 PLC vs. RLC

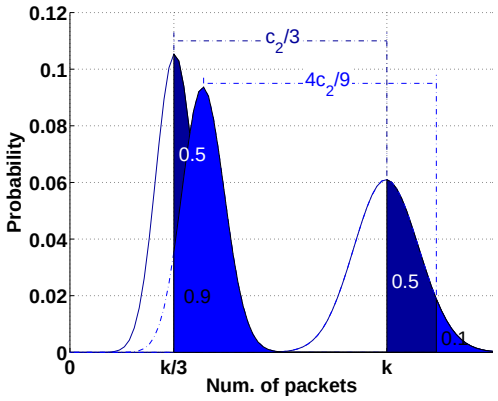
Finally, in our third set of experiments we examine the behavior of our proposed PLC mechanism as compared to the RLC mechanism, with respect to the PSNR metric. We replace the plain data streams with video ones and collect the received videos from each destination under both mechanisms. Each lost or late frame is replaced by the previous video frame, that could be replaced by the frame before the previous one for the same reason, etc. (we always provide the first frame to all destinations during our experimentation). Subsequently, in the extreme case that nothing is received on-time from a destination, the corresponding perceived video corresponds to a sequence of repeated frames that are the same with the first one. Obviously, if an inter-frame is not lost or late but the corresponding intra-frame is, then the inter-frame is useless. Figure 5.5 illustrates how we evaluate the received video stream using the PSNR metric.

We conduct the experiments in almost lossless links by configuring the transmission error prob-

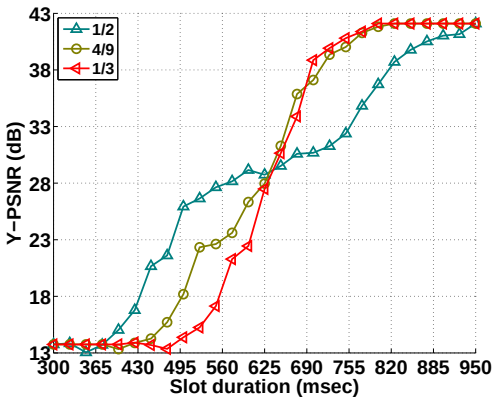
abilities as in the previous experiment. We configure $\alpha = 1/3$ and we use the video sequence of *foreman* with CIF resolution, encoded in H.264 with GOP size $g = 10$ and only I/P-frames (no B-frames). The quality of the H.264 compression (in particular quantization) is such as the average size of a compressed GOP to be almost equal to the batch size ($k/l \simeq 1$), while the size of each I-frame is approximately the $\alpha = 1/3$ proportion of the whole GOP size. For different frame ratios f, our scheme utilizes different time slots equal to $\tau = g/f$. In Figure 5.4(d), we observe how our enhanced PLC mechanism prioritizes the decoding of I-frames, outperforming the simple RLC mechanism. The horizontal axis represents the slot duration τ in milliseconds, while the vertical axis represents the perceived video quality in destinations, measured in PSNR, comparing with the original YUV sequence. We notice that the lowest PSNR value of 13.4 corresponds to the video sequence that results from no batch reception, while the largest PSNR value of 42.1 corresponds to the video sequence that results from no occurrence of lost batch. Moreover, the PSNR gain of PLC is high in cases that $|e_1 - e_2|$ is large enough, as it happens in the previous experiment.

Relaxing the *first-decode-then-transmit* policy After replacing RLC with PLC, our first approach was to relax the *first-decode-then-transmit* policy at the relays, enabling them to start transmitting even if they have decoded only the most important packets of each batch. We named this policy *partial-decode-then-transmit*. Although this policy seems to be more efficient in case that the delivery of the whole batch to the relay is impossible (enabling at least the forwarding of the most important packets), our experimental results show that it is inefficient due to the increased collisions between the source and the relay. As we see in Figure 5.6, the PSNR performance of the *partial-decode-then-transmit* policy is much worse than this of the *first-decode-then-transmit*, for all cases. Based on these results, we believe that it is better to follow the *first-decode-then-transmit* policy together with PLC.

Giving more credit to I-frames We also experimented to share non-proportionally the credit among the priority classes of PLC, giving more than αc_2 credit to the relays for the transmission of the most important packets of the batch. In this way, we increase the probability that the destination successfully decode the most important packets of each batch, however, at the cost of reducing the probability for successful decoding of the whole batch. This practice may offer better perceptual experience to the destinations, prioritizing even more the packets of the I-frames. As we see in Figure 5.7(a), by enabling the relay to transmit more than αc_2 packets as linear combinations of the most important packets, the probability of successful decoding of these packets increases from 0.5 to 0.9, however, in cost of the probability of decoding all packets, which decreases from 0.5 to 0.1. As we see in Figure 5.7(b), by increasing the number of packets that relay produces as combinations of the most important packets, the destinations deliver better video quality for short slots, but worse quality for longer slots. The best choice for a network operator to manage this trade off depends on the video sequence. Future work is to make the relay flexible to this choice, enabling the source to estimate the appropriate proportion of credit for the most important packets transmitted by the relays and propagate them this value.



(a) The probability of successful decoding of the most important and all packets for slot $\tau = 586$ msecs. The two left PDFs show the probabilities the most important packets to be decoded, when the relay transmits $c_2/3$ or $4c_2/9$ packets as linear combinations of these packets, while the right PDFs show the corresponding probabilities of all packets to be decoded.



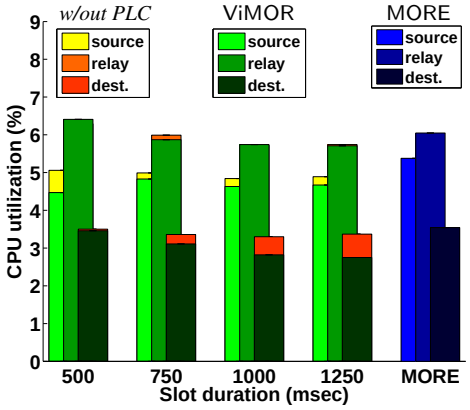
(b) The PSNR performance of ViMOR when the relay transmits $1/3 \cdot c_2$, $4/9 \cdot c_2$ and $1/2 \cdot c_2$ packets as linear combinations of the most important packets.

Figure 5.7: The performance of ViMOR for $k = 64$, $\alpha = 1/3$, and error transmission probabilities $e_1 \approx 0.001$ and $e_2 = 2/3$.

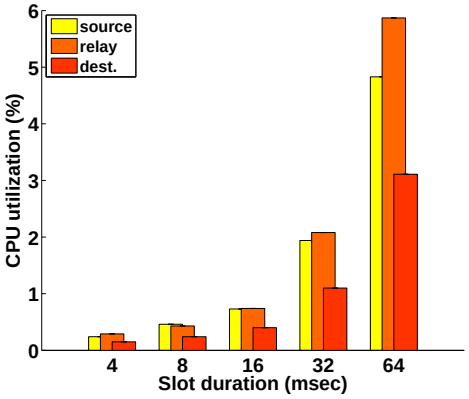
5.4.5 CPU Occupancy

Our last set of experiments in this topology evaluates the performance of ViMOR and MORE in terms of CPU utilization. In Figure 5.8(a), we show the CPU utilization of each node during the whole video streaming for various slot durations. As we see, the CPU utilization is quite similar under both protocols for small slots. However, as the slot increases, ViMOR requires less CPU cycles comparing to MORE. The most CPU demanding process is this of the relays, while the source and destinations require approximately the 75% and 50% of the source's CPU cycles respectively. We also evaluated ViMOR with and without PLC. It is highly remarkable that ViMOR with PLC requires even less CPU cycles, since the extra encoding/deconding process does not increase the CPU usage, as we expected, but decrease it. The rationale behind this observation is that with PLC, the number of computations that the encoding/decoding processes require is less than this of ViMOR-*without PLC*, since some of the encodings/decodings involve a smaller set of initiative packets.

Figure 5.8(b) depicts the CPU utilization of ViMOR for multiple values of $k = 4, 8, 16, 32, 64$. Comparing with Figure 5.4(b), we see a tradeoff between the throughput performance and the CPU utilization of ViMOR. For higher values of k the achieved throughput increases, and this is why we choose $k = 64$. However, in case of devices with less CPU power (e.g. sensors), we remark that the protocol can be significantly less CPU-consuming by decreasing the value of k, in cost of the throughput performance.



(a) The CPU utilization of the source, the relays and the destinations under MORE, ViMOR and ViMOR-*without PLC*. The utilization of all nodes is independent from the slot duration under MORE, while it is decreasing when the slot increases under ViMOR and ViMOR-*without PLC*.



(b) The CPU utilization of ViMOR for $k = 4, 8, 16, 32, 64$ and slot duration $\tau = 750$ msecs.

Figure 5.8: The CPU utilization of the source, the relays and the destinations under MORE, ViMOR and ViMOR-*without PLC*. The last scheme is ViMOR with the relays using RLC instead of PLC.

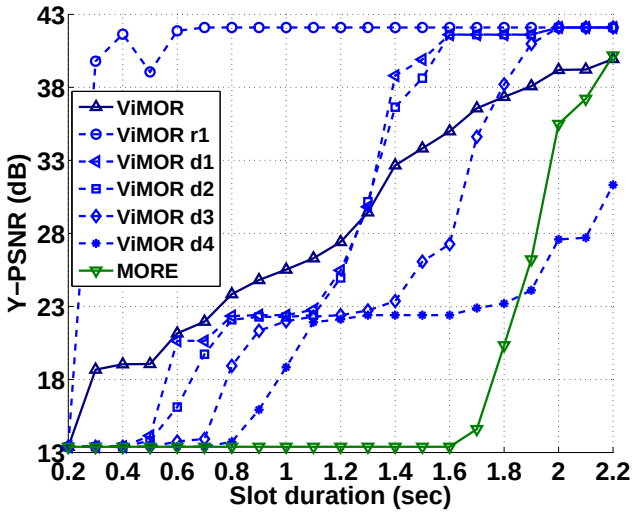


Figure 5.9: Video performance comparison between ViMOR and MORE in the 7-nodes topology of Figure 5.3(b). The dashed lines correspond to the PSNR evaluation of the receipt video of each individual destination under ViMOR.

5.4.6 Second class of experiments

The second class of our experiments aims at comparing the performance of ViMOR to MORE in terms of PSNR, with an overall evaluation of all contributions (slotted mechanism, enhanced transmission policy and PLC). The experiments were conducted in the 7-nodes topology of Figure 5.3(b), where source is s, $\mathcal{R} = \{r_1, r_2\}$ and $\mathcal{D} = \{r_1, d_1, d_2, d_3, d_4\}$. The other configuration variables remain the same as in the previous experiments, since $k = 64$, RTS/CTS is disabled, $\rho = 6$ Mbps, 1470 bytes is the payload size and the video-specifics $\alpha = 1/3$ and $g = 10$.

In Figure 5.9, ViMOR obviously enables the r_1 node to enjoy high quality video for $\tau > 0.6$ sec, while the other two-hop destinations receive a satisfying quality of video after several slots. In particular, all destinations receive a video stream with PSNR greater or equal to 22.4 for $\tau > 1.1$ sec, which corresponds to a video sequence where all I-frames are almost received and P-frames are not. This occurs when the destinations are able to decode only the high priority o_h packets of each forwarded batch, that approximately include the I-frame of the corresponding GOP. The average PSNR value among all destinations, under the ViMOR scheme, is increasing constantly for all slot durations $\tau > 0.3$ sec, while the corresponding PSNR value of the MORE scheme is increasing after slot $\tau > 1.7$ sec. Obviously, ViMOR enables video streaming of acceptable quality, even in a subset of the destinations, with slot durations up to 5.3 times smaller than the corresponding of MORE. Moreover, the PSNR gain is up to 270% for a slot $\tau = 1.6$ sec, while the average gain is 175%.

5.5 Chapter Conclusions & Future Work

In this chapter, we presented ViMOR, the first scheme that efficiently forwards multicast video over wireless networks. To the best of our knowledge, this is the first implementation of a video-aware multicast OR algorithm for 802.11 mesh networks. Of course, there are some other works that propose/simulate routing algorithms considering some of the aspects of our focus, however, none of them is actually implemented and developed on realistic setups. The potential of this research effort is well promising, since the results of our experimentation depict a PSNR gain up to 270%, comparing to the correspondent performance of the existing implementation of MORE. Of course, there are many open issues for further research.

For example, a rate control algorithm that enables the utilization of larger rates than the basic one may allow higher throughput and perceived video quality. However, this comes at the cost of reducing the network coverage area. Moreover, as we already mentioned before, future work could be to make the relays flexible on the choice of the number of transmitted packets, generated from the most important packets of the high priority class. Another point for further research is the effect of an increased number of priority classes, as well as a different way of sharing the credit c among the priority classes. It would also be interesting to design and evaluate algorithms for credit allocation between multiple flows, based on multiple criteria and goals. Last but not least, LTE is an appealing cellular access technology that promotes the use of relays and seems to be an attractive research area for applying most of our concepts in ViMOR. Our future work definitely includes the extension of our scheme to LTE. These all are challenging issues and subjects for our ongoing research.

Chapter 6

Testbed Innovations for Experimenting with Wired and Wireless Software Defined Networks

WIDELY available and remotely accessible testbeds have been used for a direct comparison of innovative protocols and ideas with existing technologies. Therefore, multiple testbeds have been established, aiming at providing experimentation services with both wireless and wired networks. In this context, several frameworks have been developed that enable easy experimentation with the heterogeneous resources that the testbeds provide. However, most of these testbeds aim only in wireless/wired networking experimentation, resulting in unsimilar testbed control and experimentation tools. Several attempts have been made towards bridging this gap in order to allow experimentation with heterogeneous wired and wireless resources. In this article, we present our contributions in extending the state-of-the-art control and management framework for wireless testbeds with support for Software Defined Networking resources. As a proof of concept, we demonstrate two use cases that take advantage of our extensions using novel architectures and present our findings.

6.1 Introduction

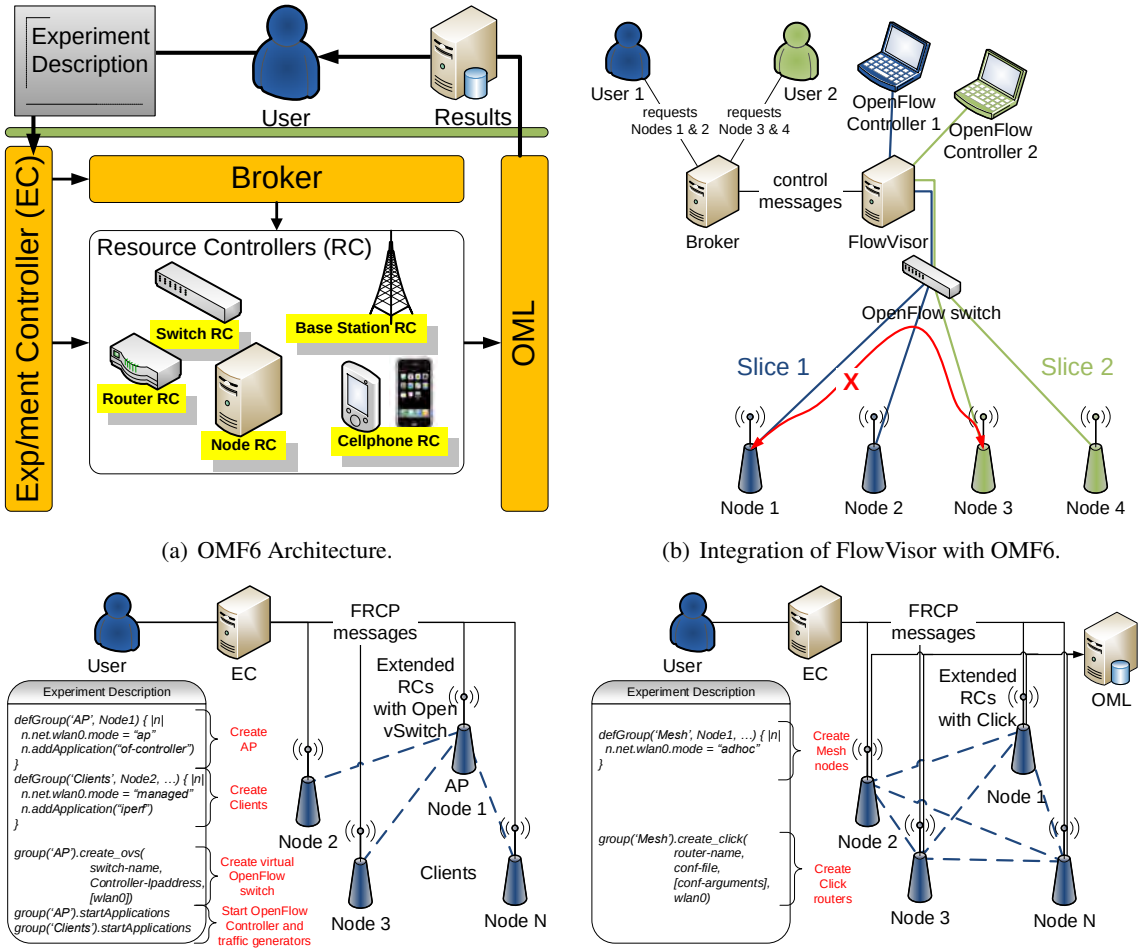
Software Defined Networking (SDN) is an approach in networking that decouples the process of taking forwarding decisions (control plane) from the underlying system that forwards the traffic (data plane). Over the last years, several theoretical underpinnings on SDN have been established, thus inspiring the conception and deployment of many novel network architectures. Most of these architectures feature individual but unified backbone and access network segments, where the former ones rely on high performance wired connections and the latter ones exploit the benefits of the wireless medium. The wired connections provide high reliability and availability, while the wireless ones leverage on easily deployed access networks. Subsequently, numerous initiatives [57, 58, 59] have emerged which fund projects aiming at bridging wireless and wired networking and building en-

hanced end-to-end systems with the use of SDN technologies. These projects are abstractly divided in two classes. The first class focuses on theory-driven protocol and architecture design, which is expected to boost the flexibility as well as the utilization of the existing network infrastructures, while the second class targets at the implementation and evaluation of each proposed design under real experimentation platforms.

The experimental platforms provided by these initiatives are essential for the research community, since consensus is growing that a complete and detailed theoretical study of a network problem has to be followed by an applied extensive validation of the corresponding results. As a consequence, research in joint wired and wireless SDN requires testing platforms equipped with the appropriate resources that enable realistic and large-scale experimentation. For instance, the OpenLab [57] and FIBRE [58] projects developed and enhanced heterogeneous testbeds, such as NITOS (wireless networking), PlanetLab-Europe (wired networking), w-iLab.t (wireless networking), i2Cat (SDN) and University of Bristol (SDN and Optical networking) islands. Similarly, the SmartFIRE [60] project aims in building a unified SDN enabled testbed between Europe and South Korea. These testbeds enable experimentation on both wireless and wired network topologies by making use of virtual or physical OpenFlow [61] switches. OpenFlow is a protocol able to separate the control and forwarding plane of the switches, being the most widely known SDN enabler.

Another key issue in the consistent operation of these testbeds is their efficient and flexible management. A framework that enables the control and management of their resources, as well as the description and execution of experiments with use of these resources, is very essential. The experimentation on both OpenLab and FIBRE environments is based on the state-of-the-art cOntrol and Management Framework (OMF) [62], enabling user-friendly scripting and conduction of experiments. OMF is a widely adopted framework by the Future Internet Research and Experimentation (FIRE) [59] community in Europe and the Global Experimentation in Network Innovation (GENI) [63] in the United States. It is open source and integrates all the necessary functionalities about the life-cycle of experimentation on testbeds. The founders and main developers of OMF initially focused on the administration of wireless resources and then moved to a more generic approach where wired links among wireless nodes are also handled.

This chapter outlines our development efforts in extending OMF to support SDN capabilities. Our goal is to illustrate the feasibility of experimentation in SDN using wireless and wired network topologies, by providing platforms where various scenarios can be tested. The open source nature of the adopted framework enables further extension and modification of the developed contributions. To sum up, in the following Section 6.2 we illustrate the former status in testbed experimentation before our contributions. In Section 6.3, we present the SDN capabilities supported by integrating our extensions in the newest version of the experimentation framework. Implementation of use cases that take advantage of these capabilities is covered in Section 6.4. We conclude in Section 6.5.



(a) OMF6 Architecture.

(b) Integration of FlowVisor with OMF6.

(c) Integration of Open-vSwitch with OMF6.

(d) Integration of Click Modular Router with OMF6.

Figure 6.1: Architecture of the described extensions

6.2 Status in Testbed Experimentation

Large scale experimentation has been enabled through the establishment of several testbed islands over the past years. They feature multiple heterogeneous resources, remotely available and widely utilized, leveraging on cutting-edge networking equipment. However, the efficient operation of a testbed is strongly related to the availability of an easily accessible interface that enables sufficient provisioning and management of its resources. The existence of such an interface is one of the most attractive testbed features for their users, especially when this interface enables the one-click execution of experiments described in an easy-learned and human-readable language. cOntrol and Management Framework (OMF), is fulfilling successfully these prerequisites, thus boosting its wide adoption by several testbed operators worldwide. Several other smaller-scale frameworks exist, applied only in wireless/wired testbeds exclusively.

OFELIA Control Framework (OCF) [64] is a widely used tool enabling experimentation exclu-

Table 6.1: Comparison of the Experimentation Frameworks.

Testbed Frameworks	Experimentation control		Federated Resource Provisioning
	Wireless support	Wired support	
OMF [62]	yes	yes	yes
OCF [64]	no	yes	yes
ProtoGENI [63]	limited	yes	yes
NEPI [59]	yes	yes	no
Teagle [59]	no	no	yes
TEFIS	no	no	yes
FEDERICA	no	no	yes

sively on top of OpenFlow compatible wired networks. Nevertheless, OCF does not support the control of wireless devices that might be present in one single testbed site. Similar to OCF, ProtoGENI [63] is another widely used control framework that focuses on the cluster experimentation, operating virtual machines that are hosted on powerful servers. Nonetheless, it lacks the flexibility of adapting in heterogeneous testbed environments, since its ability to control multiple kinds of resources is rather limited. In the same context, multiple research projects make remarkable efforts in solving multiple open issues on testbed provisioning, experimentation and federation. However, their use from only a limited number of testbeds does not guarantee their off-the-shelf adoption and smooth operation. An example of this case is the NEPI [59] framework. As far as NEPI is concerned, it leverages on the standardized interfaces of OMF, and provides the experimenter an alternative python based interface for handling the testbed resources. Other frameworks target only in providing a federated resource discovery, reservation and provisioning (like TEFIS, FEDERICA, Teagle [59] etc.) and no experiment control. Table 6.1 provides an overview of a comparison between the aforementioned frameworks.

OMF is a generic framework; stemming from the wireless world, its open source and easily extensible nature has been a vehicle for its wide adoption among many testbed operators, regardless of the type of resources they provision. OMF is written in a the Ruby scripting language, with a plethora of libraries available for network specific operations. Its architecture is modular (as illustrated in Figure 6.1(a)) consisting of different components endowed with the operation of the experiment orchestration and the resource control. Using a simple human readable experiment definition, OMF is supporting the whole experiment lifecycle, cooperating also with its accompanying framework, OMF Measurement Library (OML). The experimenter submits a simple script to the OMF Experiment Controller (EC) and the underlying functionality is responsible for setting up the resources, running the defined applications and collecting the results in an organized way. We can easily conclude that due to its flexibility and mild learning curve, OMF has boosted real world experimentation in testbed facilities.

The newest version of OMF, named OMF6, is even more extensible enabling the enhancement of its core functionality supported by the Broker [65] and the adaptation to various testbed specifications. More specifically, OMF6 defines a standardized protocol that may be used for controlling resources

of any type, such as computers or sensors, networking equipment or any other SDN component. This protocol uses a standardized sequence of messages sent by the EC to the Resource Controllers (RCs) and vice versa. The RC is a daemon that behaves as a proxy between the EC and the resource, translating the messages of the EC to executable commands for the resource and vice versa. Testbed operators are able to use this flexible protocol to extend the experimenter's control on new testbed resources or even establish federations of testbeds, thus enhancing the experimentation ecosystem. In the following section we present our work, based on the OMF6 framework, that facilitates wireless and wired SDN experiments in a real testbed environment.

6.3 Bridging Wireless and Wired SDN Experimentation

The research community that carries out the majority of the SDN based efforts has been mainly focusing on the wired However, wireless connections are improved in terms of throughput efficiency and reliability, while they still offer low-cost connectivity for end-devices. The analysis and evaluation of the performance of various end-to-end scenarios, that usually involve a wireless access network and a wired backbone infrastructure, is one of the most attractive research fields [5]. Towards satisfying this growing demand for experimentation, testbed operators have been extending their platforms appropriately, with either SDN-capable, wired or wireless resources.

In this context, we choose to adopt the state-of-the-art OMF6 testbed framework and extend it for handling the respective SDN components. In the following subsections we present our extensions to integrate the *FlowVisor*, *Open-vSwitch* and *Click Modular Router* functionalities in the OMF6 platform.

6.3.1 Testbed OpenFlow slicing

As the demand for testbed experimentation increases, the efficient utilization of the testbed resources is one of the main goals for each testbed operator. Slicing is assisting towards this goal, especially for large-scale facilities with subsets of resources that lie idle otherwise. OMF6 satisfies these demands out-of-the-box, taking advantage of its slicing capabilities. The provisioning of isolated experimental slices is facilitated by the component named *Broker*. Our developed slicing service for OpenFlow switches is offered complementary to the traditional slicing scheme imposed by OMF6. Slicing on such a switch is usually performed using the *FlowVisor* [66] framework (Figure 6.1(b)). FlowVisor behaves as a network hypervisor, which enables the concurrent usage of an OpenFlow switch by multiple experimenters.

In particular, each OpenFlow switch uses a predefined protocol in order to communicate with a remote server. This server is called OpenFlow controller and is endowed with the process of communicating software defined flows to the switch, regarding the switching decisions that it shall take upon arrival of an unknown network frame. FlowVisor is nothing more than a special purpose OpenFlow controller, which acts as a transparent proxy between any OpenFlow switch and multiple experimentation specific OpenFlow controllers. From the perspective of the OpenFlow switch, FlowVisor is its

controller. It isolates parts of the underlying hardware switch and provides access to these subparts to experimentation specific controllers. Slicing might depend on several attributes of the switch, like for example the number of ports used, the physical switch memory or processing power utilized per controller instance. The slicing may also be based on the packet flow characteristics, like the IP/MAC source and destination addresses or the VLAN tagging.

We extended the Broker entity to control the FlowVisor process and allocate completely isolated OpenFlow switch slices upon a user's request. The slices are isolated based on the switch's physical ports, thus preventing each experimenter to interact with the traffic intended for another slice. When a user reserves testbed nodes attached to a physical switch, the Broker transparently creates an OpenFlow slice, consisting of the ports where the reserved nodes are attached. As it is illustrated in Figure 6.1(b), with the existing Broker functionality, each user reserves two nodes that share a wireless connection using an idle or non interfering with other users' wireless frequency. Our extensions take place at the wired OpenFlow enabled backbone connection of the nodes, and upon the node reservation set up the appropriate FlowVisor instance which abstracts the testbed switch that the two depicted nodes connect to.

6.3.2 Virtual OpenFlow switches

The creation of virtual OpenFlow switches relies on the *Open-vSwitch* [67] tool, which is used in multiple commercial products and runs in many large scale production networks. Open-vSwitch is a software tool for creating SDN based networks, using computers instead of dedicated network devices. An example of the wide Open-vSwitch adoption is the PlanetLab-Europe testbed, which upgraded its functionality by supporting OpenFlow capabilities, thus enhancing the testbed's initial usage and upgrading it to one of the largest scale SDN experimentation facilities. With the use of this software, experimenters are able to create an overlay OpenFlow network between the PlanetLab-Europe nodes, that now play the role of virtual OpenFlow switches.

Although Open-vSwitch has been initially developed for managing wired networks by creating network bridges, it can be efficiently used for managing wireless interfaces that are parts of a bridge. If such an interface is placed in an Open-vSwitch bridge, the experimenter has the ability to intercept the traffic that is exchanged over the wireless interface as Ethernet based frames (since the wireless header is removed upon each packet reception by the wireless driver). Although this seems to be a time saving advantage for the researcher, it also poses many questions regarding the controllability of the SDN enabled wireless switch. To this aim, we enable an SNMP agent process on the wireless nodes, which allows us to remotely configure the wireless interfaces in a software defined manner.

Based on these processes for creating wireless OpenFlow switches, we developed the corresponding extensions to the OMF6 framework that allow this functionality. We significantly extended the *Node RCs*, which are in charge of receiving the proper configuration messages and applying the corresponding settings to the underlying physical machines (nodes) of the testbed. All the existing commands of the Open-vSwitch API are supported by our extended RC. Complementary to this, we developed the appropriate exchange messages among the OMF6 entities for instructing the RC to

send the appropriate *snmp-set* commands for configuring the wireless interfaces, and the *snmp-get* commands for retrieving their status. Accordingly, the EC entity, which is in charge of sending the appropriate messages to the RC based in the experiment description submitted by the user, has been extended to support this functionality.

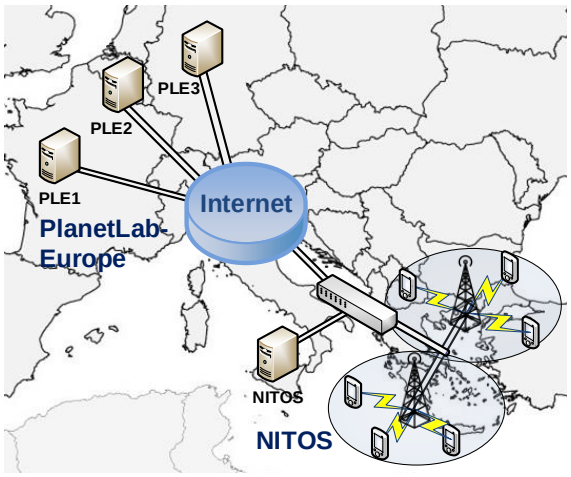
The messages exchanged are based on the *Federated Resource Control Protocol (FRCP)* standardized by the OMF6 research community. With our extensions, the experimenter can now use the testbed framework to transparently create and configure virtual switches, combining even wireless resources, in large scale using a user friendly and human readable experiment description. An example of such a set up is the configuration of an Open-vSwitch instance consisting of the wireless interface of a node, the execution of an OpenFlow contoller to control this switch, and multiple wireless clients to connect and generate traffic accordingly on the wireless network in a single experiment definition. Figure 6.1(c) illustrates the ease of experimentation on this scenario using OMF6.

The aforementioned OMF6 extensions have been developed and evaluated using virtual switches combining several heterogeneous wireless technologies that are available at the NITOS testbed. Namely, our extensions support the concurrent operation and configuration of Atheros and Intel based WiFi interfaces, Intel and Teltonika WiMAX interfaces and Huawei LTE interfaces in a single OpenFlow wireless switch.

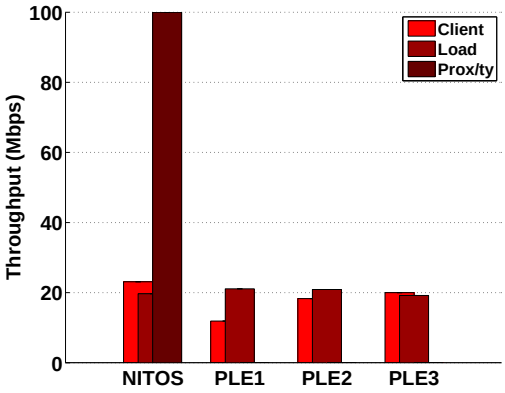
6.3.3 Software Defined Routers

Click (Modular Router) [7] is another long established software tool that its capabilities can be exploited for SDN development. More specifically, Click enables the development of software defined routers with use of Linux operated physical machines. In Roofnet [18], Click developers investigated wireless connectivity issues and proposed a routing algorithm named after it. Their framework is extensible and well documented, enabling the implementation of many routing algorithms with significantly low effort. The alternative option for packet forwarding in a wireless mesh is the 802.11s protocol, that relies on a similar approach called path selection. Nonetheless, Click is much more flexible and extensible than the existing 802.11s implementations.

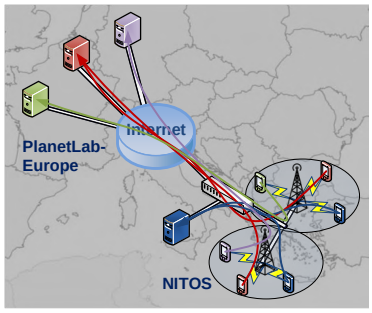
Our extensions to support the Click framework have not been so straightforward as for the other two frameworks. Click is a highly configurable tool with many users being able to develop their own extensions of the supported functionality, using the so-called *elements*. As it is illustrated in Figure 6.1(d), we follow a different approach in order to support as many as possible configurations. The Node RC is only responsible for executing the Click router in the user-space level with the appropriate arguments. The experimenter submits to the EC a configuration file that describes the desired Click settings. With this approach, the experimenters can now define new elements, which did not exist at the time that our developments took place, and use them to orchestrate their experimentation in large scale mesh networks. We have moved one step beyond in the extension of our framework and enabled OML support in the core Click system, responsible for capturing the output of Click execution and injecting the measurements in a database. Using our provided hooks in the Click version 2.0.1, the experimenter can easily support new measurements from the latest released elements.



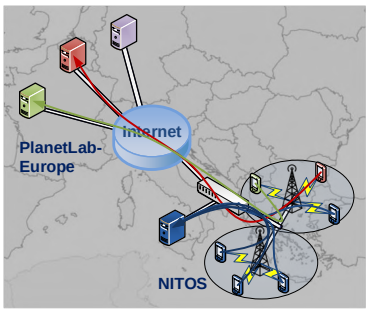
(a) The experiment topology.



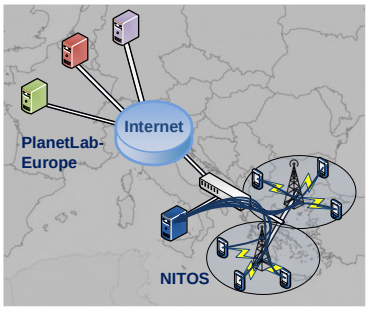
(b) Throughput performance of the three load balancing policies.



(c) Server Request mapping for the Client-based policy



(d) Server Request mapping for the Load-based policy



(e) Server Request mapping for the Proximity-based policy

Figure 6.2: European-wide private network (VPN) based on Open-vSwitch. Each color is mapping a server-client pair.

6.4 Experimental Scenarios with use of Wireless and Wired SDN

In this section we present some experimental results that clearly illustrate the potential of the experimentation in joint wireless and wired SDN. We illustrate the significance of our contributions by running realistic use cases over our enhanced platforms. In particular, we present two separate scenarios, where we try to cope with various research challenges that require deep investigation and experimentation for defining the best approaches.

6.4.1 Building content-based private networks in geographically expanded areas

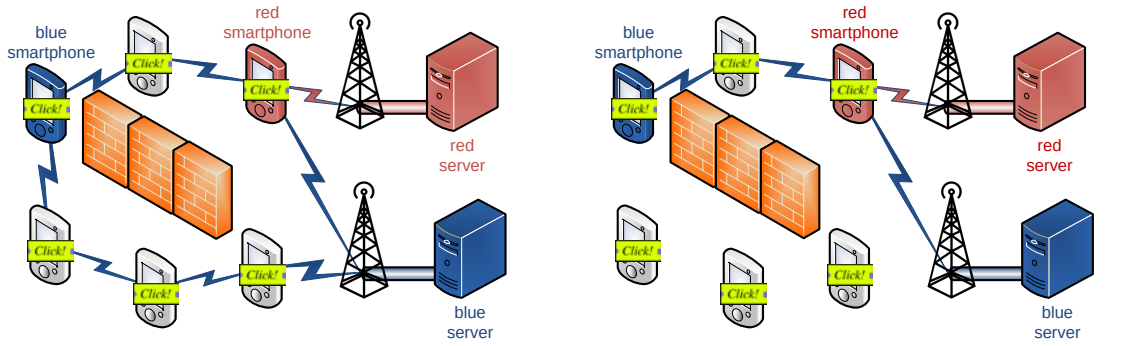
The objective of this scenario is the implementation of large-scale network infrastructures that combine a wireless access network and a wired backbone. The extended use of Virtual Private Networks (VPNs), able to define broadcast-domains with expanded coverage area all over the world, is a strong motivation for improving their capabilities. Their efficiency is boosted significantly when they rely

on schemes that balance the traffic load among multiple traffic flows. Experimenting over such these facilities seems to be very attractive in scenarios involving multiple geographically distributed users who request common contents. In such cases, directing the users' requests to a set of content servers, rather than overloading a single end-point, is the optimal solution. The capacity and performance of these networks can be significantly improved, when alleviating the traditional addressing schemes and imposing a content-based approach. A content-based approach with load balancing capabilities can be efficiently implemented by using SDN techniques, like the OpenFlow technology.

An enhanced large-scale network which supports multiple opportunistic end-point connections and load balancing capabilities, should be based on wireless access-points that are interconnected with a set of content servers through an "intelligent" wired backbone. In [5], authors blend SDN-capable wireless and wired resources in a single pan-European topology. The experimentation on this scheme is based on OMF6, utilizing the Open-vSwitch and SNMP related add-ons. In [5], a VPN is used among wireless access-points/stations, geographically distributed content servers and virtual/physical OpenFlow switches. More specifically, the functionality of the broadcast domain imposed by the VPN is modified, by altering the operation of the Address Resolution Protocol (ARP) and implementing load balancing schemes. The modified broadcast domain connects multiple European-wide sites provided by the OpenLab platform, including multiple content servers from PlanetLab-Europe (PLE1, PLE2 and PLE3) and several wireless nodes from NITOS (including smartphones and the NITOS machine), as it is depicted in Figure 6.2(a).

OpenFlow is used in order to mangle the traditional ARP process and establish the appropriate load balancing schemes. The authors provide their solution as an Information Centric approach, backwards compatible and directly applicable to traditional IP-based networks, where the IP address is not used as a unique identifier of each host but is characterizing a specific content. In this sense, a single host like for example a video streaming server, would feature several IP-addresses, one for each content that it provides. Since the interconnected networks are IP-compatible, the traditional procedure when searching for a host is performed; the requesting client issues an ARP request message and awaits for the ARP reply with the destination MAC address. Subsequently, an ARP request will trigger multiple ARP replies in the same broadcast domain. The intermediate OpenFlow equipment, with the respective OpenFlow controller, intercepts these replies and according to the load balancing scheme imposed, forwards the respective ARP reply. Each load balancing scheme that maps appropriately a set of requesting clients to a group of identical servers, sharing among them the aggregate traffic from these clients, deals with a variety of architectural challenges. The investigated architecture assumes that the content identifier is a URL, as the Content Delivery Networks do, and leave the end-points to be untouched operating as usually. The modifications apply on the underlying OpenFlow network, which intercepts the ARP messaging process in order to implement a variety of load balancing policies.

Thus, the adopted topology is involving two wireless virtual OpenFlow switches, with multiple wireless clients requesting a content from servers deployed in several sites in Europe. Physical wired OpenFlow switches are also employed at the different sites. The experimental topology and the



(a) An enhanced wireless access network with cooperative diversity. EBoW [1] uses efficiently both paths for the forwarding of the traffic generated by the blue smartphone and forwarded to the blue server.

(b) SRCR [18] uses only the shortest path to forward the traffic generated by the blue smartphone, which passes through the overloaded red smartphone.

Figure 6.3: Wireless mesh network based on Click Modular Router. The wireless links indicate which paths are used for the traffic forwarding.

content requesting applications are automatically deployed by using our aforementioned extensions in OMF6. The OpenFlow switches (virtual and hardware) are instructed to apply three different load-balancing policies. One policy, named Client-based policy, forwards each client to a specific server, that is chosen when the client first joins the network (Figure 6.2(c)). The Load-based policy periodically redirects some clients trying to unload the most congested servers (Figure 6.2(d)), while the Proximity-based policy just forwards the ARP reply from the most approximate server to the client (Figure 6.2(e)). In Figure 6.2(b) we present our findings for these three policies. As we can see, the Proximity-based policy allocates all the requests to one server, and thus the maximum throughput is received at one end point (NITOS server). On the other hand, the Load-based and Client-based policies manage to distribute the requests among the different servers (NITOS, PLE1, PLE2, PLE3).

6.4.2 Enhancing the wireless access network with cooperative diversity provided by mesh networks

The focus of this scenario is the enhancement of the access network architecture. Normally, using the standardized protocol stacks that are employed in commercial devices, wireless networks with infrastructure (access-points) do not allow the cooperation among the stations. On the other hand, a wireless mesh architecture implies that there are no access-points and stations but peers and gateways, enabling the cooperation among the peers. The wireless mesh is an emerging architecture that is standardized (802.11s, Roofnet) and enables better performance in wireless networks when the users are crowded or mobile. The efficiency of a wireless mesh depends on the utilized routing protocol.

Shortest path routing is a conceptually simple and widely used approach, since there are several routing protocols that implement it. These protocols aim at achieving minimum end-to-end delay and maximum throughput performance, assuming that the network is not congested. However, the case of network congestion is a frequent phenomenon which should be seriously considered and

appropriately handled. This is the motivation behind multi-path routing research, where alternative paths to the destination are utilized in order to provide resistant and seamless communication. Theory driven research on this area has concluded in a scheme, named Back-Pressure [9], which achieves throughput optimality under some prerequisites. In particular, Back-Pressure operates on a time-slotted and centralized schedule, introducing a specific scheduling and routing policy. Additionally, there is an improved version, named Enhanced Back-Pressure, that can be configured to move packets in the direction of their shortest paths, improving the delay performance. However, both schemes have not been implemented and used in realistic networks, mainly because of the impracticable centralized scheduling policy and the time-slotted assumption.

Multiple research efforts focus on designing wireless and distributed routing algorithms, that implement as many as possible of the Enhanced Back-Pressure aspects and are applied in real-life wireless mesh networks. One of these works is presented in [1], where the authors propose a scheme that incorporates most of the Enhanced Back-Pressure (EBoW) aspects in a WiFi compliant fashion. In particular, they introduce a scheme in which every node attempts to forward packets to less loaded and closer to the destination neighbors. The proposed scheme is using multiple paths to forward the wireless traffic, thus superseding other state-of-the-art multi-path routing algorithms (Horizon [16] and CDP [17]), as well as the well known shortest-path SRCR algorithm [18]. The experimentation on these protocols performance is based on the OMF6 framework and its enhancements for Click Modular Router support. OMF6 controls the nodes and enforces them to behave as Back-Pressure routers, using the Click software tool. Figure 6.3(a) illustrates the experimentation topology shaped on the NITOS platform with use of OMF6.

The integration of Click with OMF6 enables the configuration of this scheme in a large scale topology, similar to the access network of a realistic topology such as this of Figure 6.2(a), which the authors employed to evaluate their scheme. The evaluation of the proposed EBoW scheme, as well as its comparison with the other state-of-the-art wireless routing algorithms, is presented in Chapter 2. The clear advantage of the EBoW scheme can be seen in terms of throughput network performance.

6.5 Chapter Conclusions & Future Work

In this chapter, we describe the implementation of our SDN-specific extensions in the state-of-the-art control and management framework for wireless testbeds. We discuss the necessity and feasibility for experimentation using joint wireless and wired SDN, and the problems in testbed administration that we overcome using our contributions. Through the development and evaluation of realistic large scale scenarios with the use of the enhanced framework, we show the significant benefits of our developments. Our ongoing work includes further SDN extensions for the OMF6 framework, as well as the evaluation of more use cases and scenarios, in a large scale federated environment.

Related Publications

[1] K. Choumas, T. Korakis, I. Koutsopoulos, and L. Tassiulas. Implementation and End-to-end Throughput Evaluation of an IEEE 802.11 Compliant Version of the Enhanced-Backpressure Algorithm. In *Proc. Tridentcom*, 2012.

[2] G.S. Paschos, C. Li, E. Modiano, K. Choumas, and T. Korakis. Multirate Multicast: Optimal Algorithms and Implementation. In *Proc. IEEE INFOCOM*, 2014.

[3] K. Choumas, I. Syrigos, T. Korakis, and L. Tassiulas. Video-aware Multicast Opportunistic Routing over 802.11 two-hop mesh networks. In *Proc. IEEE SECON*, 2014.

[4] K. Choumas, N. Makris, T. Korakis, L. Tassiulas and M. Ott. Testbed Innovations for Experimenting with Wired and Wireless Software Defined Networks. In *CNERT workshop of IEEE ICDCS*, 2015.

[5] K. Choumas, N. Makris, T. Korakis, L. Tassiulas and M. Ott. Exploiting Openflow Resources towards a Content-Centric LAN. In *IEEE EWSDN*, 2013.

Bibliography

[6] NITLab: Network Implementation Testbed Laboratory. `http://nitlab.inf.uth.gr/NITlab/index.php/testbed`.

[7] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M.F. Kaashoek. The Click Modular Router. *ACM Trans. on Computer Systems*, 18(3):263–297, August 2000.

[8] L. Tassiulas and A. Ephremides. Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks. *IEEE Trans. on Automatic Control*, 37(12):1936–1948, December 1992.

[9] M.J. Neely, E. Modiano, and C.E. Rohrs. Dynamic Power Allocation and Routing for Time Varying Wireless Networks. In *Proc. IEEE INFOCOM*, 2003.

[10] L. Georgiadis, M.J. Neely, and L. Tassiulas. Resource Allocation and Cross-layer Control in Wireless Networks. *Found. Trends Netw.*, 2006.

[11] NS-3: Network Simulator. `http://www.nsnam.org`.

[12] R. Laufer, T. Salonidis, H. Lundgren, and P. Le Guyadec. XPRESS: a Cross-Layer Backpressure Architecture for Wireless Multi-Hop Networks. In *Proc. ACM MobiCom*, 2011.

[13] A. Sridharan, S. Moeller, B. Krishnamachari, and M. Hsieh. Implementing Backpressure-based Rate Control in Wireless Networks. In *Proc. IEEE ITA*, 2009.

[14] A. Warrier, S. Janakiraman, S. Ha, and I. Rhee. DiffQ: Practical Differential Backlog Congestion Control for Wireless Networks. In *Proc. IEEE INFOCOM*, 2009.

[15] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali. Routing Without Routes: The Backpressure Collection Protocol. In *Proc. IEEE/ACM IPSN*, 2010.

[16] B. Radunović, C. Gkantsidis, D. Gunawardena, and P. Key. Horizon: Balancing TCP over Multiple Paths in Wireless Mesh Network. In *Proc. ACM MobiCom*, 2008.

[17] A.A. Bhorkar, T. Javidi, and A.C. Snoereny. Achieving Congestion Diversity in Wireless Ad-hoc Networks. In *Proc. IEEE INFOCOM*, 2011.

[18] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and Evaluation of an Unplanned 802.11b Mesh Network. In *Proc. ACM MobiCom*, 2005.

[19] R. Draves, J. Padhye, and B. Zill. Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks. In *Proc. ACM MobiCom*, 2005.

[20] J. Li, C. Blake, D. De Couto, H. Lee, and R. Morris. Capacity of Ad Hoc Wireless Networks. In *Proc. ACM MobiCom*, 2001.

[21] Iperf: The TCP/UDP Bandwidth Measurement Tool. `http://sourceforge.net/projects/iperf/`.

[22] X. Li, S. Paul, and M. Ammar. Layered Video Multicast with Retransmissions (LVMR): Evaluation of Hierarchical Rate Control. In *Proc. IEEE INFOCOM*, 1998.

[23] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven Layered Multicast. In *Proc. ACM SIGCOMM*, 1996.

[24] S. Shakkottai and R. Srikant. Network Optimization and Control. *Found. Trends Netw.*, 2007.

[25] S. Sarkar and L. Tassiulas. Fair Allocation of Utilities in Multirate Multicast Networks: A Framework for Unifying Diverse Fairness Objectives. *IEEE Trans. on Automatic Control*, 47(6):931–944, June 2002.

[26] M.J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan and Claypool Publishers, 2010.

[27] S. Sarkar and L. Tassiulas. A Framework for Routing and Congestion Control for Multicast Information Flows. *IEEE Trans. on Information Theory*, 48(10):2690–2708, October 2002.

[28] K. Kar, S. Sarkar, and L. Tassiulas. A Scalable Low-overhead Rate Control Algorithm for Multirate Multicast Sessions. *IEEE Journal on Sel. Areas in Comm.*, 20(8):1541–1557, October 2002.

[29] S. Deb and R. Srikant. Congestion Control for Fair Resource Allocation in Networks with Multicast Flows. *IEEE/ACM Trans. on Networking*, 12(2):274–285, April 2004.

[30] L. Bui, R. Srikant, and A. Stolyar. Optimal Resource Allocation for Multicast Flows in Multihop Wireless Networks. In *Proc. Control and Decision Conference (CDC)*, 2007.

[31] S. Sarkar and L. Tassiulas. Back Pressure Based Multicast Scheduling for Fair Bandwidth Allocation. *IEEE Trans. on Neural Networks*, 16(5):1279–1290, September 2005.

[32] R. Laufer, T. Salonidis, H. Lundgren, and P. Le Guyadec. A Cross-Layer Backpressure Architecture for Wireless Multihop Networks. *IEEE/ACM Trans. on Networking*, 22(2):363–376, April 2014.

[33] C. Li and E. Modiano. Receiver-Based Flow Control for Networks in Overload. `http://arxiv.org/abs/1207.6354`.

[34] C. Li and E. Modiano. Receiver-Based Flow Control for Networks in Overload. In *Proc. IEEE INFOCOM*, 2013.

[35] Dimitri P.B. *Nonlinear Programming*. Athena Scientific, 1999.

[36] M.J. Neely, E. Modiano, and C. Li. Fairness and Optimal Stochastic Control for Heterogeneous Networks. *IEEE/ACM Trans. on Networking*, 16(2):396–409, April 2008.

[37] C. Li and E. Modiano. Receiver-Based Flow Control for Networks in Overload. *IEEE/ACM Trans. on Networking*, 23(2):616–630, April 2015.

[38] N. Tatbul, U. Çetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker. Load Shedding in a Data Stream Manager. In *Proc. VLDB*, 2003.

[39] AP. Bianzino, C. Chaudet, D. Rossi, and J. Rougier. A Survey of Green Networking Research. *IEEE Commun. Surveys Tuts.*, 2012.

[40] L.M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proc. IEEE INFOCOM*, 2001.

[41] M.J. Neely. Energy Optimal Control for Time Varying Wireless Networks. *IEEE Trans. on Information Theory*, 52(7):2915–2934, July 2006.

[42] S. Keranidis, G. Kazdaridis, V. Passas, T. Korakis, I. Koutsopoulos, and L. Tassiulas. Online Energy Consumption Monitoring of Wireless Testbed Infrastructure Through the NITOS EMF Framework. In *Proc. ACM WiNTECH*, 2013.

[43] M.J. Neely. Super-Fast Delay Tradeoffs for Utility Optimal Fair Scheduling in Wireless Networks. *IEEE Journal on Sel. Areas in Comm.*, 24(8):1489–1501, August 2006.

[44] L. Huang and M.J. Neely. Delay Reduction via Lagrange Multipliers in Stochastic Network Optimization. *IEEE Trans. on Automatic Control*, 56(4):842–857, April 2011.

[45] L. Bui, A. Eryilmaz, R. Srikant, and X. Wu. Joint asynchronous congestion control and distributed scheduling for wireless networks. In *Proc. IEEE INFOCOM*, 2006.

[46] Y. Yuan, H. Yang, S.H.Y. Wong, S. Lu, and W. Arbaugh. ROMER: Resilient opportunistic mesh routing for wireless mesh networks. In *Proc. IEEE Workshop on Wireless Mesh Networks (WiMesh)*, 2005.

[47] S. Biswas and R. Morris. ExOR: opportunistic multi-hop routing for wireless networks. In *Proc. ACM SIGCOMM*, 2005.

[48] S. Chachulski (now Jakubczak), M. Jennings, S. Katti, and D. Katabi. Trading Structure for Randomness in Wireless Opportunistic Routing. In *Proc. ACM SIGCOMM*.

[49] C. Fragouli, D. Katabi, A. Markopoulou, M. Mdard, and H. Rahul. Wireless Network Coding Opportunities & Challenges. In *Proc. MILCOM*, 2007.

[50] S. Katti, D. Katabi, H. Balakrishnan, and M. Medard. Symbol-Level Network Coding for Wireless Mesh Networks. In *Proc. ACM SIGCOMM*, 2008.

[51] Y. Lin, B. Li, and B. Liang. CodeOR: Opportunistic routing in wireless mesh networks with segmented network coding. In *Proc. IEEE ICNP*, 2008.

[52] D. Koutsonikolas, C. Wang, and Y.C. Hu. CCACK: Efficient Network Coding Based Opportunistic Routing Through Cumulative Coded Acknowledgments. In *Proc. IEEE INFOCOM*, 2010.

[53] D. Koutsonikolas, Y.C. Hu, and C. Wang. Pacifier: high-throughput, reliable multicast without "Crying babies" in wireless mesh networks. *IEEE/ACM Trans. on Networking*, 20(5):1375–1388, October 2012.

[54] Z. Li, L. Sun, X. Zhou, and L. Li. Priority Linear Coding Based Opportunistic Routing for Video Streaming in Ad Hoc Networks. In *Proc. GLOBECOM*, 2010.

[55] S.M. Das, H. Pucha, K. Papagiannaki, and Y.C. Hu. Studying wireless routing link metric dynamics. In *Proc. ACM Internet Measurement Conference (IMC)*, 2007.

[56] O. Alay, T. Korakis, Y. Wang, E. Erkip, and S.S. Panwar. Layered wireless video multicast using omni-directional relays. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008.

[57] OpenLab: The OpenLab federation of various European testbeds. `http://www.ict-openlab.eu`.

[58] FIBRE: Future Internet testbeds experimentation between Brazil and Europe. `http://www.fibre-ict.eu`.

[59] Fed4FIRE: Federation of European facilities for FIRE. `http://www.fed4fire.eu`.

[60] SmartFIRE: Enabling SDN Experimentation in Wireless Testbeds exploiting Future Internet Infrastructures in South Korea and Europe. `http://eukorea-fire.eu`.

[61] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. In *Proc. ACM SIGCOMM*, 2008.

[62] T. Rakotoarivelo, M. Ott, G. Jourjon, and I. Seskar. OMF: a control and management framework for networking testbeds. In *Proc. ACM SIGOPS*, 2010.

[63] GENI: Exploring Network of the Future. http://www.geni.net.

[64] Expedient: A Pluggable Platform for GENI Control Frameworks. http://yuba.stanford.edu/~jnaous/expedient.

[65] D. Stavropoulos, A. Dadoukis, T. Rakotoarivelo, M. Ott, T. Korakis and L. Tassiulas. Design, Architecture and Implementation of a Resource Discovery, Reservation and Provisioning Framework for Testbeds. In *Proc. WiNMeE*, 2015.

[66] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. Can the production network be the testbed? In *Proc. OSDI*, 2010.

[67] OvS: Open virtual Switch. http://openvswitch.org.

[68] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Appendix A

Derivation of Bound in (3.15)

From (3.1)-(3.2) and [33, Lemma 7], we obtain

$$\frac{1}{2}\big([Q_l^{(c)}(t+1)]^2-[Q_l^{(c)}(t)]^2\big)\le B_Q-Q_l^{(c)}(t)\Big(\mu_l^{(c)}(t)+d_l^{(c)}(t)-\mu_{p(l)}^{(c)}(t)-A^{(c)}(t)1_{[l\in\mathcal{L}_{\text{out}}(c)]}\Big), \tag{A.1}$$

$$\frac{1}{2}\big([D_l^{(c)}(t+1)]^2-[D_l^{(c)}(t)]^2\big)\le B_D-D_l^{(c)}(t)\big(\varphi_l^{(c)}(t)-d_l^{(c)}(t)\big), \tag{A.2}$$

where

$$B_Q\triangleq(\mu_{\max}+d_{\max})^2+(\mu_{\max}+A_{\max})^2,\ B_D\triangleq 2(d_{\max})^2. \tag{A.3}$$

Multiplying (A.1) and (A.2) by $m_l^{(c)}$, summing over l and c, and taking conditional expectation, we have

$$\begin{aligned}
&\Delta(t)+V\sum_{c,l}m_l^{(c)}\,\mathbb{E}_H[\varphi_l^{(c)}(t)]\\
&\le B_1+\sum_{c,l\in\mathcal{L}_{\text{out}}(c)}m_l^{(c)}Q_l^{(c)}(t)\,\lambda^{(c)}\\
&-\sum_{c,l}m_l^{(c)}Q_l^{(c)}(t)\mathbb{E}_H[\mu_l^{(c)}(t)+d_l^{(c)}(t)]\\
&+\sum_{c,l}m_l^{(c)}Q_l^{(c)}(t)\mathbb{E}_H[\mu_{p(l)}^{(c)}(t)]\\
&-\sum_{c,l}m_l^{(c)}\,D_l^{(c)}(t)\mathbb{E}_H[\varphi_l^{(c)}(t)-d_l^{(c)}(t)]\\
&+V\sum_{c,l}m_l^{(c)}\,\mathbb{E}_H[\varphi_l^{(c)}(t)],
\end{aligned} \tag{A.4}$$

where $B_1=\frac{1}{2}\sum_{c,l}m_l^{(c)}(B_Q+B_D)$ and $V>0$ is a predefined parameter. Rearranging terms and using the equality, for each session c,

$$\sum_l m_l^{(c)}Q_l^{(c)}(t)\,\mathbb{E}_H[\mu_{p(l)}^{(c)}(t)]\quad=\sum_l\mathbb{E}_H[\mu_l^{(c)}(t)]\sum_{l':\,p(l')=l}m_{l'}^{(c)}Q_{l'}^{(c)}(t), \tag{A.5}$$

we obtain (3.15).

Appendix B

Proof of Theorem 3.1

We denote by $\Upsilon(\pi,t)$ the RHS of the inequality (3.15) evaluated under a policy π in slot t, where $\Upsilon(\pi,t)$ can be re-written as

$$\begin{aligned}\Upsilon(\pi,t) &\triangleq B_1 + V\sum_{c,l} m_l^{(c)}\mathbb{E}_H[\varphi_l^{(c)}(t)] \\ &+\sum_{c,l} m_l^{(c)} Q_l^{(c)}(t)\mathbb{E}_H\left[\mu_{p(l)}^{(c)}(t) + \lambda^{(c)}1_{[l\in\mathcal{L}_{\text{out}}(c)]} - \mu_l^{(c)}(t) - d_l^{(c)}(t)\right] \\ &-\sum_{c,l} m_l^{(c)} D_l^{(c)}(t)\mathbb{E}_H[\varphi_l^{(c)}(t) - d_l^{(c)}(t)]. \end{aligned} \tag{B.1}$$

It is clear that the decision variables in (B.1) are chosen according to policy π. Let $DPP(\pi,t)$ be the sum (3.14) evaluated under policy π in slot t. The inequality (3.15) under the MMT policy is

$$DPP(\mathsf{MMT},t) \leq \Upsilon(\mathsf{MMT},t).$$

Now, consider the optimal stationary policy, denoted by *STAT*, that chooses $\mathrm{E}[\mu_l^{(c)}(t)] = f_l^{(c)*}$ and $\mathrm{E}[d_l^{(c)}(t)] = \mathrm{E}[\varphi_l^{(c)}(t)] = q_l^{(c)*}$ for all t, where $q_l^{(c)*}$ and $f_l^{(c)*}$ are the optimal flow variables that solves (3.10). This is a feasible policy. Since the MMT policy minimizes the RHS of (B.1) in every slot, we have

$$DPP(\mathsf{MMT},t) \leq \Upsilon(\mathsf{MMT},t) \leq \Upsilon(STAT,t). \tag{B.2}$$

Using the flow conservation (3.3)-(3.4), we have

$$\Upsilon(STAT,t) = B_1 + V\sum_{c,l} m_l^{(c)} q_l^{(c)*}. \tag{B.3}$$

Plugging (B.3) into (B.2) yields, under the MMT policy,

$$\Delta(t) + V\sum_{c,l} m_l^{(c)}\,\mathbb{E}_H[\varphi_l^{(c)}(t)] \leq B_1 + V\sum_{c,l} m_l^{(c)} q_l^{(c)*}.$$

Taking expectation, summing over slots $\{0,\ldots,t-1\}$, dividing by Vt, and using $\mathbb{E}[L(0)] \geq 0$, we have

$$\frac{\mathbb{E}[L(t)]}{Vt} + \sum_{c,l} m_l^{(c)} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}[\varphi_l^{(c)}(\tau)] \leq \frac{B_1}{V} + \sum_{c,l} m_l^{(c)} q_l^{(c)*}.$$

Lemma 1 shows that $\mathbb{E}[L(t)]$ is finite for all t. Taking $t \to \infty$ yields

$$\sum_{c,l} m_l^{(c)} \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}[\varphi_l^{(c)}(\tau)] \leq \frac{B_1}{V} + \sum_{c,l} m_l^{(c)} q_l^{(c)*}.$$

Since all $D_l^{(c)}(t)$ queues are bounded, we have

$$\hat{d}_l^{(c)} \leq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}[\varphi_l^{(c)}(\tau)].$$

It follows that

$$\sum_{c,l} m_l^{(c)} \hat{d}_l^{(c)} \leq \frac{B_1}{V} + \sum_{c,l} m_l^{(c)} q_l^{(c)*}.$$

In a multicast session c, along the path $\mathcal{P}_\delta^{(c)}$ from the source c to a receiver δ, traffic that arrives at source c in an interval $[0,t]$ must be equal to the sum of total dropped packets and data delivered to receiver u in that interval, plus the total queue backlogs $\sum_{l\in\mathcal{P}_\delta^{(c)}} Q_l^{(c)}(t)$ at time t. In other words,

$$\sum_{\tau=0}^{t-1} A^{(c)}(\tau) = \sum_{\tau=0}^{t-1} \sum_{l\in\mathcal{P}_\delta^{(c)}} \widetilde{d}_l^{(c)}(\tau) + \sum_{\tau=0}^{t-1} \widetilde{\mu}_{l_\delta}^{(c)}(\tau) + \sum_{l\in\mathcal{P}_\delta^{(c)}} Q_l^{(c)}(t).$$

Taking expectation and time average, and using the finiteness of queues $Q_l^{(c)}(t)$, we have as $t \to \infty$

$$\lambda^{(c)} = \overline{r}_\delta^{(c)} + \sum_{l\in\mathcal{P}_\delta^{(c)}} \hat{d}_l^{(c)}, \quad \forall \delta \in \Delta^{(c)}. \tag{B.4}$$

Summing (B.4) over $c \in C$ and $\delta \in \Delta^{(c)}$ yields

$$\sum_c |\Delta^{(c)}| \lambda^{(c)} = \sum_{c,\delta} \overline{r_\delta^{(c)}} + \sum_{c,l} m_l^{(c)} \hat{d}_l^{(c)} \tag{B.5}$$

$$\leq \sum_{c,\delta} \overline{r}_\delta^{(c)} + \frac{B_1}{V} + \sum_{c,l} m_l^{(c)} q_l^{(c)*}. \tag{B.6}$$

From (3.9), the maximum total throughput is

$$\sum_{c,\delta} r_\delta^{(c)*} = \sum_c |U^{(c)}| \lambda^{(c)} - \sum_{c,l} m_l^{(c)} q_l^{(c)*}.$$

As a result,

$$\sum_{c,\delta} \bar{r}_{\delta}^{(c)} \geq \sum_{c,\delta} r_{\delta}^{(c)*} - \frac{B_1}{V}.$$

The proof is complete.

Appendix C

Derivation of Bound in (3.24)

Define the indicator function $1_{\delta c}^{R}(t) = 1$ if $Z_{\delta}^{(c)}(t) \geq Q$, and 0 otherwise. Let $1_{\delta c}^{L}(t) = 1 - 1_{\delta c}^{R}(t)$. Define $\rho_{\delta}^{(c)}(t) = \nu_{\delta}^{(c)}(t) - \mu_{l_\delta}^{(c)}(t)$ and $\delta_{\max} = \max[\nu_{\max}, \mu_{\max}]$. Then we have $|\rho_{\delta}^{(c)}(t)| \leq \delta_{\max}$ and $\nu_{\delta}^{(c)}(t) \leq \nu_{\max}$. For each link $l \in E$, we obtain from [33, Lemma 7] that

$$\frac{1}{2}\big([Q_l^{(c)}(t+1)]^2 - [Q_l^{(c)}(t)]^2\big) \leq B_Q - Q_l^{(c)}(t)\Big(\mu_l^{(c)}(t) + d_l^{(c)}(t) - \mu_{p(l)}^{(c)}(t) - A^{(c)}(t)1_{[l\in L_{\text{out}}(c)]}\Big),$$
$$\frac{1}{2}\big([D_l^{(c)}(t+1)]^2 - [D_l^{(c)}(t)]^2\big) \leq B_D - D_l^{(c)}(t)\big(\varphi_l^{(c)}(t) - d_l^{(c)}(t)\big),$$

where B_Q and B_D are given in (A.3). Using the same analysis in [33, Lemma 5], we get

$$e^{w(Z_\delta^{(c)}(t+1)-Q)} - e^{w(Z_\delta^{(c)}(t)-Q)} \leq e^{w(\nu_{\max}+\mu_{\max})} + w\,(\delta_{\max} + \frac{\varepsilon}{2}) - w\,1_{\delta c}^{R}(t)\,e^{w(Z_\delta^{(c)}(t)-Q)}\big[\rho_\delta^{(c)}(t) - \frac{\varepsilon}{2}\big],$$
$$e^{w(Q-Z_\delta^{(c)}(t+1))} - e^{w(Q-Z_\delta^{(c)}(t))} \leq e^{wQ} + w\,(\delta_{\max} + \frac{\varepsilon}{2}) + w\,1_{\delta c}^{L}(t)\,e^{w(Q-Z_\delta^{(c)}(t))}\big[\rho_\delta^{(c)}(t) + \frac{\varepsilon}{2}\big].$$

Combining the above inequalities yields

$$\begin{aligned}
\Delta(t) \leq\ & B_2 - \sum_{c,l} m_l^{(c)} D_l^{(c)}(t)\,\mathbb{E}_H\Big[\varphi_l^{(c)}(t) - d_l^{(c)}(t)\Big] \\
& - \sum_{c,l} m_l^{(c)} Q_l^{(c)}(t)\,\mathbb{E}_H\Big[\mu_l^{(c)}(t) + d_l^{(c)}(t) - \mu_{p(l)}^{(c)}(t) + A^{(c)}(t)1_{[l\in L_{\text{out}}(c)]}\Big] \\
& - w\sum_{\delta c} 1_{\delta c}^{R}(t)\,e^{w(Z_\delta^{(c)}(t)-Q)}\big[\mathbb{E}_H[\rho_\delta^{(c)}(t)] - \frac{\varepsilon}{2}\big] \\
& + w\sum_{\delta c} 1_{\delta c}^{L}(t)\,e^{w(Q-Z_\delta^{(c)}(t))}\big[\mathbb{E}_H[\rho_\delta^{(c)}(t)] + \frac{\varepsilon}{2}\big],
\end{aligned}$$

where

$$B_2 = \frac{1}{2}\sum_{c,l} m_l^{(c)}(B_Q + B_D) + \frac{1}{2}|U||C|\left[e^{wQ} + e^{w(\nu_{\max}+\mu_{\max})} + w(2\delta_{\max} + \varepsilon)\right]. \tag{C.1}$$

By definition of $Y_\delta^{(c)}(t)$ we have

$$Y_\delta^{(c)}(t) = w\, 1_{\delta c}^{R}(t)\, e^{w(Z_\delta^{(c)}(t)-Q)} - w\, 1_{\delta c}^{L}(t)\, e^{w(Q-Z_\delta^{(c)}(t))}.$$

Using $\rho_\delta^{(c)}(t) = \nu_\delta^{(c)}(t) - \mu_{l_\delta}^{(c)}(t)$ and rearranging terms, we obtain the requested.

Appendix D

Proof of Theorem 4.1

Proof of Theorem 4.1, part (1). To show that the queue lengths are deterministically bounded by (4.18), we use induction. Since the system starts empty, (4.18) clearly holds for $t = 0$. Suppose it also holds for t, we will show it is true for $t+1$. First, assume $Q_n^{(c)}(t) \leq KV$. Then from (4.1) we have

$$\begin{aligned} Q_n^{(c)}(t+1) &\leq Q_n^{(c)}(t) + \mu_{p(n)}^{(c)}(t) \\ &\leq KV + A_{\max} + \max_{(n,m)} R_{nm} = KV + d_{\max}. \end{aligned}$$

Else, if $Q_n^{(c)}(t) > KV$ then by the inductive step it has to be $KV < Q_n^{(c)}(t) \leq KV + d_{\max}$. From (4.17), E-DLS decides to drop $d_{\max}$ packets in this slot. Hence

$$Q_n^{(c)}(t+1) \leq \left(Q_n^{(c)}(t) - d_{\max}\right)^+ + \mu_{p(n)}^{(c)}(t) \leq KV + d_{\max}.$$

This completes the proof for (4.18). □

Proof of Theorem 4.1, part (2). In this part, we will show that the dynamic packet-level decisions converge in the limit to time-averages that support maximum throughput. The total number of session c packets generated up to time τ are split in the following way

$$\underbrace{\sum_{t=0}^{\tau-1} A^{(c)}(t)}_{\text{arrived}} = \underbrace{\sum_{t=0}^{\tau-1} \tilde{\mu}_{p(\delta_c)}^{(c)}(t)}_{\text{received}} + \underbrace{\sum_n \sum_{t=0}^{\tau-1} \tilde{d}_n^{(c)}(t)}_{\text{dropped}} + \underbrace{\sum_n Q_n^{(c)}(\tau)}_{\text{queued}},$$

where we denote with $\tilde{d}_n^{(c)}(t)$ the actual number of packets dropped from node n, since $\tilde{d}_n^{(c)}(t)$ depends on backlog availability and $\tilde{d}_n^{(c)}(t) \leq d_n^{(c)}(t)$. A consequence of the bounded backlogs from part (1) is that $\lim_{\tau\to\infty} \frac{1}{\tau} \sum_{c,n} \mathbb{E}[Q_n^{(c)}(\tau)] = 0$. Summing up over sessions, taking expectations, dividing by τ,

taking limits, and using the Law of Large Numbers for $A^{(c)}(t)$, we get

$$\sum_c \lambda^{(c)} = \hat{T}(K) + \sum_{c,n} \hat{q}_n^{(c)}, \tag{D.1}$$

where $\hat{q}_n^{(c)} \triangleq \lim_{\tau\to\infty} \frac{1}{\tau} \sum_{t=0}^{\tau-1} \mathbb{E}[\tilde{d}_n^{(c)}(t)]$ is the limit of time average dropping for E-DLS.

We introduce RAN, a helpful randomized policy that uses an offline solution to (4.12) $(\boldsymbol{f}^*, \boldsymbol{q}^*)$ (the solution depends on K, V), and then chooses controls so that $\mathbb{E}[\mu_n^{(c)}(t)|\mathbf{Q}(t)] = f_n^{(c)*}$ and $\mathbb{E}[d_n^{(c)}(t)|\mathbf{Q}(t)] = q_n^{(c)*}$. This is an ω-only optimal policy [26]. Let us denote by $Y(t)$ the RHS of (4.15) evaluated under E-DLS and $Y^*(t)$ under RAN. Since E-DLS is designed to minimize the RHS of (4.15) at every slot, we have $Y(t) \leq Y^*(t)$, and combining with (4.15) we have

$$\Delta(t) + V \sum_{c,n} \alpha_n^{(c)}(K) \mathbb{E}[d_n^{(c)}(t)|\mathbf{Q}(t)] \leq Y^*(t). \tag{D.2}$$

Canceling out terms using the flow conservation (4.3) we find that $Y^*(t) = B + V \sum_{c,n} \alpha_n^{(c)}(K) q_n^{(c)*}$. So, summing up the inequality (D.2) over slots $\{0, ..., \tau-1\}$, dividing by $V\tau$, taking expectation with respect to backlogs, and using $L(0) = 0$,

$$\frac{\mathbb{E}L(\tau)}{V\tau} + \frac{1}{\tau} \sum_{t=0}^{\tau-1} \sum_{c,n} \alpha_n^{(c)}(K) \mathbb{E}[d_n^{(c)}(t)] \leq \frac{B}{V} + \sum_{c,n} \alpha_n^{(c)}(K) q_n^{(c)*}$$

Taking limits, and using $\lim_{\tau\to\infty} \mathbb{E}L(\tau)/\tau \to 0$ from (4.18), the above becomes

$$\sum_{c,n} \alpha_n^{(c)}(K) \hat{q}_n^{(c)} \leq \frac{B}{V} + \sum_{c,n} \alpha_n^{(c)}(K) q_n^{(c)*}. \tag{D.3}$$

Also, by the constraint on maximum drops per slot we have $\sum_{c,n} h_n^{(c)} \hat{q}_n^{(c)} \leq CN^2 d_{\max}$. Thus,

$$\begin{aligned}
\hat{T}(K) &\overset{(D.1)}{=} \sum_c \lambda^{(c)} - \frac{1}{K} \sum_{c,n} \alpha_n^{(c)}(K) \hat{q}_n^{(c)} - \frac{1}{K} \sum_{c,n} h_n^{(c)} \hat{q}_n^{(c)} \\
&\overset{(D.3)}{\geq} \sum_c \lambda^{(c)} - \frac{B/V + \sum_{c,n} \alpha_n^{(c)}(K) q_n^{(c)*} + CN^2 d_{\max}}{K} \\
&\overset{(4.6)}{\geq} \sum_c r^{(c)*}(K) - \frac{B/V + CN^2 d_{\max}}{K},
\end{aligned}$$

where $\sum_c r^{(c)*}(K)$ is the max throughput from (4.12). Using proposition 4.2.1 from [35], any solution of (4.12) converges to some solution of (4.10), i.e., $\lim_{K\to\infty} \sum_c r^{(c)*}(K) = T^*$. The proof of (4.19) is completed by taking the limit $K \to \infty$. □

Proof of Theorem 4.1, part (3). From (4.7) we have

$$\begin{aligned} E^* &= \min_{\boldsymbol{f},\boldsymbol{q}\geq \mathbf{0}} \sum_{c,n} f_n^{(c)} \overset{(4.9)}{=} \sum_c N^{(c)} \lambda^{(c)} - \max_{\boldsymbol{f},\boldsymbol{q}\geq \mathbf{0}} \sum_{c,n} h_n^{(c)} q_n^{(c)} \\ &= \sum_c N^{(c)} \lambda^{(c)} - \sum_{c,n} h_n^{(c)} \overline{q}_n^{(c)}, \end{aligned} \tag{D.4}$$

where $\overline{q}_n^{(c)}$ is the solution to (4.7), and $N^{(c)}$ is the number of nodes on path $P^{(c)}$ less the destination. The expected number of transmitted packets up to τ under E-DLS are

$$\sum_{t=0}^{\tau-1} \mathbb{E}[\tilde{\mu}_n^{(c)}(t)] = \sum_{t=0}^{\tau-1} \left(A^{(c)}(t) - \sum_{k\in\mathcal{U}(n)} \mathbb{E}[\tilde{d}_k^{(c)}(t)] + \mathbb{E}[Q_k^{(c)}(t)] \right).$$

Dividing by τ and taking limits we have

$$\hat{f}_n^{(c)} = \lambda^{(c)} - \sum_{k\in\mathcal{U}(n)} \hat{q}_n^{(c)}(K,V), \tag{D.5}$$

where $\hat{f}_n^{(c)} = \lim_{\tau\to\infty} 1/\tau \sum_{t=0}^{\tau-1} \mathbb{E}[\tilde{\mu}_n^{(c)}(t)]$. Now we express the total energy consumed under E-DLS as

$$\begin{aligned} \hat{E}(K,V) &\overset{(4.14)}{=} \sum_{c,n} \hat{f}_n^{(c)} \overset{(D.5)}{=} \sum_{c,n} \left(\lambda^{(c)} - \sum_{k\in\mathcal{U}(n)} \hat{q}_n^{(c)}(K,V) \right) \\ &= \sum_c N^{(c)} \lambda^{(c)} - \sum_{c,n} h_n^{(c)} \hat{q}_n^{(c)}(K,V) \\ &\overset{(D.4)}{=} E^* + \sum_{c,n} h_n^{(c)} \left[\overline{q}_n^{(c)} - \hat{q}_n^{(c)}(K,V) \right]. \end{aligned} \tag{D.6}$$

Let $\hat{q}_n^{(c)}(K,\infty)$ denote the limit performance under E-DLS as we increase $V\to\infty$. From (D.3) we have

$$\sum_{c,n} \alpha_n^{(c)}(K) \hat{q}_n^{(c)}(K,\infty) \leq \sum_{c,n} \alpha_n^{(c)}(K) q_n^{(c)*}(K). \tag{D.7}$$

Since $(\hat{q}_n^{(c)}(K,V)), (\hat{f}_n^{(c)}(K,V))$ is a feasible solution of (4.12), then by (D.7) it follows that $\hat{q}_n^{(c)}(K,\infty)$ is also an optimal one. Therefore, we may revisit the design of the RAN policy and specifically select $q_n^{(c)*}(K) = \hat{q}_n^{(c)}(K,\infty)$ for all n. Finally the proof can be completed by taking the limit $V\to\infty, K\to\infty$ on (D.6) and noticing that $\lim_{K\to\infty} q_n^{(c)*}(K) = \overline{q}_n^{(c)}$. □

Appendix E

Convexity of probability E as a function of c_1

Let $\mathcal{P}^d$ be the set of one-hop and two-hop paths connecting the source s with the destination $d \in \mathcal{D}$. Let e_{xy} be the transmission error probability of the link connecting node x to node y, thus $e_{xy}^z \in (0,1]$ is the probability of z successive transmission errors over this link. We define E_p to be the probability of unsuccessful packet delivery to the destination d through a path $p \in \mathcal{P}^d$, when the source and each relay are charged with a credit c_1 and $c_2 = (c - c_1)/R$ respectively. We show that this quantity is always a convex function of c_1.

For example, the probability of unsuccessful packet delivery through the one-hop path $p' \in \mathcal{P}^d$ is $E_{p'} = e_{sd}^{c_1}$, that is a convex function over all legitimate values of c_1. Furthermore, the corresponding probability of a two-hop path $p'' \in \mathcal{P}^d$, that utilizes a relay $r \in \mathcal{R}$, is a convex function of c_1 as well, equal to $E_{p''} = 1 - (1 - e_{sr}^{c_1})(1 - e_{rd}^{(c-c_1)/R})$. In particular, $E_{p''}$ is convex since its second derivative is always non-negative, as it is depicted in (E.1).

$$\begin{aligned} \partial^2 E_{p''} = & \underbrace{\ln(e_{sr})^2 e_{sr}^{c_1}(1 - e_{rd}^{(c-c_1)/R})}_{non-negative} \\ & + \underbrace{\ln(e_{rd})^2 e_{rd}^{(c-c_1)/R}(1 - e_{sr}^{c_1})}_{non-negative}/R^2 \\ & + 2\underbrace{\ln(e_{sr})\ln(e_{rd}) e_{sr}^{c_1} e_{rd}^{(c-c_1)/R}}_{non-negative}/R \geq 0 \end{aligned} \tag{E.1}$$

The packet is not delivered to d, if each of the paths of $\mathcal{P}^d$ fails to do it. So $E^d = \prod_{p \in \mathcal{P}^d} E_p$ is the probability of unsuccessful packet delivery to d over all paths of $\mathcal{P}^d$. As follows, E^d is a convex function of c_1 as well, since E_p is a positive and convex function for all $p \in \mathcal{P}^d$ [68], as we proved before. Finally, the average probability $E = 1 - \sum_{d \in \mathcal{D}} E^d/D$ is a convex function of c_1 again, which means that at most two c_1 integer values exist that maximize this probability and achieve system objective.