UNIVERSITÀ DI PISA

Scuola di Dottorato in Ingegneria "Leonardo da Vinci"

Corso di Dottorato di Ricerca in
Ingegneria dell'Informazione

Ph.D. Thesis

Delay-aware Link Scheduling and Routing in Wireless Mesh Networks

Alessandro Lori

2011

UNIVERSITÀ DI PISA

Scuola di Dottorato in Ingegneria "Leonardo da Vinci"

Corso di Dottorato di Ricerca in
Ingegneria dell'Informazione

Ph.D. Thesis

Delay-aware Link Scheduling and Routing in Wireless Mesh Networks

Author:

Alessandro Lori

Supervisor:

Prof.Gigliola Vaglini

2011
SSD ING-INF/05

Sommario

L'allocazione di risorse assume un ruolo centrale nella gestione delle reti di calcolatori per via degli ingenti investimenti finanziari che tale settore richiede. In questa tesi affrontiamo problemi di allocazione di risorse in reti di calcolatori in presenza di flussi real-time, sfruttando la ricerca operativa e gli strumenti che questa scienza mette a disposizione per modellarli, risolverli e analizzarli.

Il primo contesto analizzato è quello delle reti Wireless Mesh, nelle quali affrontiamo il problema della pianificazione dei collegamenti tra nodi della rete in presenza di flussi tempo-reale. Sfruttando recenti risultati ottenuti con l'utilizzo di Network Calculus, formuliamo tale problema come programmazione non-lineare mista intera. Risolvendo tale problema, mostriamo come l'ammissibilità di una soluzione dipenda dal tipo di aggregazione adottata per i flussi. Abbiamo inoltre affrontato il problema congiuntamente alla decisione sull'instradamento dei flussi, fornendo delle euristiche per ridurne la complessità di risoluzione.

Come ulteriore contributo, nel contesto delle reti wireless 802.11 proponiamo un approccio a divisione temporale all'interno del protocollo MAC CSMA. Raggruppando stazioni wireless ed assegnando a tali gruppi uno specifico intervallo temporale in cui trasmettere all'interno di un piano periodico siamo in grado di ridurre i ritardi sperimentati dai flussi trasmessi e di aumentare notevolemente la capacità fino ad un 100% per reti di grandi dimensioni.

Concludiamo trattando l'allocazione di risorse in reti cablate in presenza di flussi che richiedono garanzia su di un ritardo massimo. Il problema è formulato e risolto usando differenti modelli di latenza. Su una rete di studio i risultati ottenuti mostrano che usando l'ottimizzazione globale si possono ottenere soluzioni ammissibili quando modelli di uso comune falliscono, sorprendentemente anche per bassi utilizzi della rete.

Abstract

Resource allocation is a critical task in computer networks because of their capital-intensive nature. In this thesis we apply operations research tools and technologies to model, solve and analyze resource allocation problems in computer networks with real-time traffic.

We first study Wireless Mesh Networks, addressing the problem of link scheduling with end-to-end delay constraints. Exploiting results obtained with the Network Calculus framework, we formulate the problem as an integer non-linear optimization problem. We show that the feasibility of a link schedule does depend on the aggregation framework. We also address the problem of jointly solving the routing and link scheduling problem optimally, taking into account end-to-end delay guarantees. We provide guidelines and heuristics.

As a second contribution, we propose a time division approach in CSMA MAC protocols in the context of 802.11 WLANs. By grouping wireless clients and scheduling time slots to these groups, not only the delay of packet transmission can be decreased, but also the goodput of multiple WLANs can be largely increased.

Finally, we address a resource allocation problem in wired networks for guaranteed-delay traffic engineering. We formulate and solve the problem under different latency models. Global optimization let feasible schedules to be computed with instances where local resource allocation schemes would fail. We show that this is the case even with a case-study network, and at surprisingly low average loads.

Acknowledgements

I would like to thank Professor Gigliola Vaglini, and Professor Luciano Lenzini for teaching me how to be a researcher. My colleagues and co-authors for all the time spent in interesting and passionate discussions that led to this thesis.

Professor Prasant Mohapatra, An (Jack) Chan and all my colleagues at the Network Research Group, University of California, Davis, for the warm welcome while I was visiting.

Professor Fabio Schoen and my colleagues at the Global Optimization Lab, Università degli Studi di Firenze, for all the valuable suggestions and feedbacks on my work.

My friends. Gianni for being my unintentional mentor. My family for all the support given during my studies.

Valentina for always standing right next to me and giving me support when needed.

Contents

1 Introduction 1
1.1 Overview of the Thesis 3

2 Statement of contributions 5
2.1 Delay-aware link scheduling and routing in WMNs 5
2.2 Time division carrier sensing multiple access 7
2.3 Towards resource-optimal routing plans for real-time traffic 7

3 Delay-aware link scheduling and routing 9
3.1 Related work 9
3.2 Network model 11
3.3 Problem formulation 15
3.3.1 Per-flow and per-path queuing 15
3.3.2 Per-node queuing 18
3.3.3 Coping with downlink traffic 24
3.4 Schedulability comparison 25
3.5 Online admission control 30
3.5.1 Heuristics for the per-flow and per-path case 31
3.5.2 Per-node case 34
3.6 Joint routing and scheduling 36
3.6.1 The DARS model 37
3.6.2 Heuristic solutions 39
3.7 Conclusions 48

4 A time division carrier-sensing scheme for IEEE 802.11 Wireless LANs 51
4.1 Related work 52
4.2 Time division carrier sensing multiple access 53
4.2.1 Basic idea of TD-CSMA 53
4.2.2 Isolating co-channel interference 54
4.2.3 Frame structure in TD-CSMA 57

4.3 TD-CSMA resource optimization 58
4.3.1 Constrained cluster assignment 59
4.3.2 TD-CSMA cluster scheduling 61
4.4 Performance evaluation 64
4.4.1 Quality of services provided by TD-CSMA 65
4.4.2 Scalability of TD-CSMA 67
4.4.3 Robustness of TD-CSMA 69
4.5 Conclusions 69

5 Towards resource-optimal routing plans for real-time traffic 71
5.1 Related Work 72
5.2 System model 73
5.2.1 Scheduling and latency 75
5.2.2 Path computation algorithms 76
5.3 Optimal resource allocation 76
5.4 Numerical results 79
5.5 Conclusions 84

6 Conclusions 85

References 87

List of Figures

3.1 Paths in a sink tree 12
3.2 Different buffering frameworks 13
3.3 Conflicts in a TDMA network 14
3.4 Relevant quantities in link scheduling 14
3.5 Comparison between optimizing on $V_{\max}$ and minimizing the maximum TDMA delay 19
3.6 Relevant quantities for paths P0 and P3 21
3.7 Sample binary tree 26
3.8 Constant overall load (homogeneous flows) 26
3.9 Regions of the (σ, ρ) plane where a given aggregation model leads to smaller $V_{\max}$ (homogeneous flows case) 27
3.10 Homogeneous flows in a balanced binary tree for $\sigma = 0$ (above) and $\sigma = 1000$ (below) 28
3.11 Heterogeneous randomly generated flows 29
3.12 Flows with different deadlines in a balanced binary tree 30
3.13 Box plot of the resolution times for different WMN topologies under per-flow and per-node frameworks 31
3.14 Solution scheme for the reduced MinMVP 34
3.15 Accuracy of the reduced MinMVP problem 35
3.16 Percentage of solved instances against the standard deviation of the rate distribution 35
3.17 Box plot of the relative deviation between the optimal solution of the per-node case and the one computed using the engineered heuristic and (3.22) 36
3.18 Sample mesh 39
3.19 Separate heuristic approach 41
3.20 Accuracy comparison of the heuristic schemes 43
3.21 Solution time for the LCS+r-DARS, using the Lagrangian heuristic 43

3.22 Relative gap between the cascading and the joint approach (the latter solved through the Lagrangian heuristic) on a 6x6 grid WMN 44
3.23 Maximum violation as a function of the rate for a 5x5 grid topology 45
3.24 Maximum violation as a function of the rate for a 6x6 grid topology 45
3.25 Maximum violation as a function of the burst size 46
3.26 The test-case 5x5 WMN 46
3.27 Vmax as a function of the rate for various gateway placements - homogeneous traffic 47
3.28 Allocated capacity as a function of the rate for various gateway placements - homogeneous traffic 47
3.29 Distribution of Vmax over 30 random instances with different placements of the gateway node 48
3.30 Vmax and allocated capacity for a single gateway and two-gateway scenario 48

4.1 Different time slots (TS) are assigned to disjoint groups of clients in two nearby WLANs 54
4.2 Derivation of the cluster conflict graphs 55
4.3 The structure of the time frames in TD-CSMA 58
4.4 A 2-by-2 4-WLAN network 63
4.5 Solving time for a 2-by-2 topology 65
4.6 End-to-end delay vs. number of clients per cell 66
4.7 TD-CSMA vs. CSMA networks: total goodput of network 66
4.8 TD-CSMA vs. CSMA networks: packet dropping rate of the network 67
4.9 TD-CSMA vs. CSMA networks: end-to-end delay of the network 67
4.10 The scalability of TD-CSMA 68
4.11 The robustness of TD-CSMA 68

5.1 Sample network for numerical analysis 80
5.2 Number of oversubscribed links 80
5.3 Average utilization for the links 81
5.4 Link utilization - CM-SPF, with WRP latency, ERA 83
5.5 Link utilization - CM-SPF, with WRP latency, GRA 83

1

Introduction

Computer networks require capital intensive infrastructures to serve as many users as possible. The fundamental issue arising in such an environment is the assignment of the limited resources, which involves facing heterogeneous problems, ranging from short term decisions - for instance call admission control problems - to long term facility provisioning and expansion planning. The successful solution of these problems has played and will always play an important role in the development of computer networks and their widespread use.

Operations research (OR) is the discipline of applying advanced analytical methods such as mathematical programming to solve complex problems. OR, or simply optimization, has been applied for decades to problems arising in the context of computer networks, resulting in more efficient and effective solutions.

In this thesis we exploit mathematical programming to model important resource allocation problems arising in computer networks. Specifically, we focus our analysis on worst-case end-to-end delays experienced by real-time traffic flows while traversing wireless and wired networks, which can be modeled using some recent results based on the Network Calculus framework ([49, 51]). Unfortunately, the resulting delay functions are non-linear with respect to both integer and continous variables representing resources of a network.

While Linear Programming (LP) and convex optimization have a fairly complete theory, and can be solved numerically very efficiently, Mixed Integer Non-Linear Programming (MINLP) is still a challenging area of research. In fact, it is not until recently that several new methods are becoming available for MINLP problems [33]. Despite their complexity, even for this class of problems arising from real-world applications there exist general purpose solvers that are able to solve them optimally.

Most of our work is focused on the delay-aware link scheduling and routing problems in Wireless Mesh Networks (WMNs). WMNs are an emerging class of networks, usually built on fixed nodes that are inter-connected via wireless links to form a multi-hop network. Their main goal is to provide broadband access to mobile clients who are just on the edge of wired networks. WMNs can be used where cable deployment

is not feasible or is too expensive, such as in remote valleys or rural areas, but also in offices and home environments.

End-users are served by nodes called mesh routers, which are generally assumed to be stationary. Mesh routers are in turn wirelessly interconnected so as to form a network back-haul, where radio resource management challenges come into play. Moreover, some mesh routers are generally provided with access (e.g. through wires) to the Internet and therefore can act as gateways for the entire WMN. Communication between any two mesh routers as well as from any router to gateways is multi-hop.

Many of the WMN issues are thus common to multi-hop wireless networks. However, the fact that mesh routers are fixed makes the back-haul of a WMN inherently different from distributed wireless networks (e.g. ad hoc networks). For example, problems such as energy consumption are no longer an issue. This makes it sensible to opt for a centralized network management, as opposed to the distributed approaches used for ad hoc wireless networks. In this case, nodes act in a coordinated fashion under the supervision of a network entity which determines the management based on global knowledge of the network topology and additional conditions. The radio communication channel employed by WMNs (as by any other wireless network) is broadcasting; i.e. a packet sent out by a mesh router will be received by all mesh routers tuned on the same frequency as the transmitter and within its transmission range, and furthermore it may cause signal interference to some mesh routers that are not intended to be the receivers.

To avoid signal interference, link scheduling is used to guarantee conflict-free operation in the context of Time Division Multiple Access (TDMA, [59]), where time is slotted and synchronized. Through link scheduling, only sets of non-interfering links are activated simultaneously in each slot. WMNs are already and will be supporting real-time traffics, such as voice, video, or traffic control, whose bit rate is often highly variable, and which require firm guarantees on their maximum end-to-end delay.

Cross-layer approaches where link scheduling and routing are jointly addressed have been extensively studied in the past few years due to their application to TDMA MAC protocols [7, 42, 19, 44, 24, 71, 18, 35, 32, 52, 75, 68, 23, 58, 36]. However, no work we are aware of has considered arbitrary end-to-end delay bounds as constraints on link scheduling before. Instead, most of the available works ([7, 42, 19, 44, 24, 71, 18, 35, 32]) compute schedules constrained by the flow rates. While this approach has the obvious benefit of utilizing links efficiently, it is certainly not enough to guarantee that arbitrary prespecified delays are met. Moreover, the comparatively fewer works (e.g. [52]-[23]) that compute link schedules based on delays only take into account the sum of the waiting times due to TDMA scheduling, whereas this is only one component - and not necessarily the largest one - of the end-to-end delay, which also includes queuing. Accordingly, those algorithms often compute delay-infeasible schedules (and largely so), even when delay-feasible solutions exist.

The TDMA scheduling formulation introduced in our work is also applied to 802.11 Wireless LANs to define the Time Division Carrier Sensing Multiple Access (TD-

CSMA) approach. TD-CSMA takes the advantages from both the CSMA and TDMA to reduce the drawbacks of both technologies when real-time flows require the radio resource. The underlying idea is to allocate time slots to groups of wireless clients in a TDMA fashion, leaving to CSMA the task of resolving the remaining interferences within the client group.

Finally, the same network calculus concepts introduced to define the queuing models in WMNs can be exploited in the context of resource allocation for real-time traffic in wired networks. Once again, by means of mathematical programming, we are able to formulate rate allocation problems under different traffic scheduling policies.

1.1 Overview of the Thesis

In this thesis, we deal with resource allocation problems in both wireless and wired networks traversed by real-time flows. The rest of the thesis is organized as follows:

- For each resource allocation problem, we highlight the main contributions of our work in Chapter 2.
- Chapter 3 is dedicated to WMNs and to the problem of link scheduling and routing for real-time leaky bucket shaped flows. Under different aggregation policies we give exact formulation of the problems, solving them optimally and proposing heuristics. We devise rules of thumb to suggest which aggregation scheme may best suit a given traffic scenario and where to locate the gateways.
- In Chapter 4 the scheduling formulation introduced for WMNs is ported within the context of 802.11 Wireless LANs to define a hybrid approach denominated Time Division Carrier Sensing Multiple Access (TD-CSMA). TD-CSMA takes the advantages from both the CSMA and TDMA to reduce the co-channel interference, and at the same time not to increase the computation cost too much. The throughput achievable with TD-CSMA is up to three times the capacity in a multi-WLAN network compared with the legacy CSMA.
- In Chapter 5 the underlying concepts introduced in Chapter 3 to model traffic delays for WMNs are applied in the context of resource allocation for real-time traffic in wired networks. Specifically, we discuss the issue of computing resource-optimal routing plans in a network domain. Given a number of known traffic demands, with associated required delays, we discuss how to route them and allocate resources for them at each node so that the demands are satisfied.

2

Statement of contributions

In this thesis, we investigate whether a given set of flows subject to arbitrary deadline constraints can be routed and scheduled in several and heterogeneous contexts: WMNs, 802.11 WLANs and wired networks.

2.1 Delay-aware link scheduling and routing in WMNs

We start our analysis in Chapter 3 by assuming that shortest-path routing to the Internet gateway is in place, so that the logical topology of the WMN is a sink tree. In that setting, we formulate the link scheduling problem as a mixed integer-non linear problem, which we can solve optimally: in other words, we are able to compute a link schedule that can guarantee the required delay bounds - possibly at the price of over-allocating rates - whenever it is possible to do so. The objective to be minimized is the maximum delay violation, (i.e. the maximum difference between the worst-case delay and the requested deadline), the resulting optimal schedules are also robust, i.e. the parameters of some flows can be varied (even by large amounts) with limited impact on the actual delays, as shown in [14].

Furthermore, we show that the schedulability of a set of flows depends on their aggregation policy within the network. Flows may/may not be able to meet their deadline depending on whether they are scheduled in isolation (i.e., buffered at different queues, as in the IntServ framework [11]), or aggregated (i.e., buffered in the same FIFO queue, as in the DiffServ framework [9]), either only at their path ingress node or progressively, as their paths merge on their way toward a common gateway. We show that there are indeed cases when the aggregation policy determines the schedulability, and we devise rules of thumb to suggest which aggregation scheme may best suit a given traffic scenario.

Then, we discuss the practicability of computing delay-constrained link schedules. We show that - again depending on the flow aggregation policy - some constraints of the problem may be non convex. More specifically, this happens when flows are

aggregated progressively. When we deal with only convex constraints, optimal delay-constrained schedules can be computed for networks of up to several tens of nodes (e.g., more than 40) in minutes or hours, i.e. times that are affordable in a resource provisioning time-scale perspective. Otherwise, the computation is limited to a few nodes (e.g., up to 15), and rapidly explodes beyond that figure. Therefore, we consider trading optimality for computation time, so as to make it viable for online admission control in a dynamic environment. We describe a heuristic which allow suboptimal - but still practically good - schedules to be computed in short times, given an estimate of the flows' rate along the links. We also show that heuristics devised for the case with convex constraints only yield good performance also with non-convex constraints.

Finally, a delay-aware joint routing and scheduling formulation is presented. We show that the problem can be optimally solved for networks of up to few nodes (e.g., a $4x4$ grid), though at the price of unfeasibly long computations. We also propose heuristics based on Lagrangian decomposition to compute suboptimal solutions considerably faster and/or for larger WMNs, up to about 50 nodes. We show that the heuristic solutions are near-optimal, and we exploit them to investigate the optimal placement of one or more gateways from a delay bound perspective.

The results presented has been published or submitted for pubblication as:

- P. Cappanera, L. Lenzini, A. Lori, and G. Vaglini. "Minimum latency link scheduling in TDMA wireless multi-hop networks". Proceedings of International Network Optimization Conference (ENOG-EURO), 2009.
- P. Cappanera, L. Lenzini, A. Lori, G. Stea, and G. Vaglini. "Link scheduling with end-to-end delay constraints in wireless mesh networks". In World of Wireless, Mobile and Multimedia Networks Workshops, 2009. WoWMoM 2009. IEEE International Symposium on a, pages, 1-9, 2009;
- P. Cappanera, L. Lenzini, A. Lori, G. Stea, and G. Vaglini. "Optimal link scheduling for real-time traffic in wireless mesh networks in both per-flow and per-path frameworks". In World of Wireless Mobile and Multimedia Networks (WoWMoM), 2010 IEEE International Symposium on a, pages 1-9, 2010.
- P. Cappanera, L. Lenzini, A. Lori, G. Stea, and G. Vaglini. "Efficient link scheduling for online admission control of real-time traffic in wireless mesh networks". DOI: 10.1016/j.comcom.2011.02.004. Elsevier Computer Communications, February 2011.
- P. Cappanera, L. Lenzini, A. Lori, G. Stea, and G. Vaglini. "Optimal link scheduling for real-time traffic in wireless mesh networks in both per-flow and per-path frameworks". Submitted, January 2011.
- P. Cappanera, L. Lenzini, A. Lori, G. Stea, and G. Vaglini. "Optimal joint routing and link scheduling for real-time traffic in TDMA Wireless Mesh Networks". Submitted, March 2011.

2.2 Time division carrier sensing multiple access

The scheduling formulation introduced for WMNs is ported within the context of 802.11 Wireless LANs to define a hybrid approach denominated Time Division Carrier Sensing Multiple Access (TD-CSMA). This is the subject of Chapter 4. In fact, when multiple WLANs are present in a small geographical area - a common scenario in wireless network deployment nowadays - the wireless channel capacity is heavily degraded. This is due to the fact that the legacy 802.11 CSMA MAC protocol is unable to effectively schedule the traffic, hence severely suffering the co-channel interference from the neighboring WLANs. Since there are not enough orthogonal frequency channels, a simple frequency channel allocation cannot solve the problem. TDMA can schedule the traffic and help avoid the co-channel interference, but it brings large overheads and the computational cost steeply grows with the network size.

TD-CSMA takes the advantages from both the CSMA and TDMA to reduce the co-channel interference, and at the same time not to increase the computation cost too much. By assigning time slots to groups of wireless clients, TD-CSMA schedules the traffic of the group as a whole, while CSMA is used to avoid collisions within the group. We develop an analytical framework for TD-CSMA to effectively group the clients and schedule the traffic according to their rate and delay requirements. Our simulation results show that TD-CSMA can always achieve better performances than the legacy CSMA in various scenarios, without the drawbacks of a pure TDMA. The throughput achievable with TD-CSMA is up to three times the capacity in a multi-WLAN network compared with the legacy CSMA.
The results presented has been submitted for pubblication as:

- A. Chan, A. Lori, P. Mohapatra, L. Lenzini, G. Vaglini. “A Time Division Carrier-Sensing Scheme for IEEE 802.11 Wireless LANs”. Submitted, December 2010.

2.3 Towards resource-optimal routing plans for real-time traffic

The underlying concepts introduced in Chapter3 to model traffic delays for WMNs are applied in the context of resource allocation for real-time traffic in wired networks. Specifically, in Chapter 5 we discuss the issue of computing resource-optimal routing plans in a network domain. Given a number of known traffic demands, with associated required delays, we discuss how to route them and allocate resources for them at each node so that the demands are satisfied.

While a globally optimal routing plan requires joint computation of the paths and of the associated resources (which was claimed to be NP-hard), we stick to existing approaches for path computation, and use mathematical programming to model resource allocation once the paths are computed. We show that the problem is either convex or non-convex, depending on the scheduling algorithms adopted at the nodes.

Our results show that, by computing resources per-path, instead of globally, the available capacity can be exceeded even at surprisingly low utilizations.
The results presented has been published as:

- A. Lori, G. Stea, and G. Vaglini. “Towards resource-optimal routing plans for real-time traffic”. Lecture Notes in Computer Science, Vol. 6415, pp. 214-227, 2010, Proceedings of International Symposium On Leveraging Applications of Formal Methods (ISoLA).

3

Delay-aware link scheduling and routing

In this chapter, we present our work on two of the main challenging resource allocation problems arising in wireless mesh networks, the link scheduling and routing problems. Specifically, we consider TDMA WMNs with flows constrained by leaky bucket regulators, i.e. we describe flows using a pair of parameters: the average rate of a flow ρ, and its burstiness σ, which is the biggest block of bits that it can inject into the network in a short time. We formulate the link scheduling problem as a mixed integer-non linear problem, which we can solve optimally: the objective to be minimized is the maximum delay violation, (i.e. the maximum difference between the worst-case delay and the requested deadline).

Furthermore, we show that the schedulability of a set of flows depends on their aggregation policy within the network. We show that there are indeed cases when the aggregation policy determines the schedulability, and we investigate when the adoption of a specific aggregation policy leads to best results.

Finally, we give an exact formulation of the problem of joint routing and link scheduling of leaky-bucket constrained flows that request worst-case delay guarantees and we propose a heuristic to deal with the increased complexity of the problem.

3.1 Related work

In this section we review the available related work on link scheduling in WMNs. As the literature on the routing and link scheduling in WMNs is abundant, here we reviewed only those works that are more germane to our work, leaving out anything connected with multi-radio systems (where the channel assignment problem is the most prominent issue) and/or not dealing with performance bounds. No work that we are aware of considered schedulability in WMNs with arbitrary end-to-end delay constraints. Most of the relevant literature on link scheduling falls into either of the following categories:

1. rate-oriented algorithms, that either provide flows with a minimum guaranteed rate (e.g. [7, 42, 19, 43, 24]), or optimize the total throughput (e.g. [71, 18, 35, 32]).

Guaranteeing a minimum rate no smaller than the flow rate - by (3) or (7) - is a necessary condition for end-to-end delays to be finite, but does not automatically make them smaller than a prespecified bound. These algorithms represent one extreme in the trade-off between rate and delay. We show later on that - by renouncing over-allocating rates - the above schemes often compute schedules with very large delays.

2. TDMA delay-oriented algorithms, that either minimize (e.g. [68, 23]) or try to guarantee a maximum TDMA delay (e.g. [52, 75]). The latter is the sum of TDMA waiting times at every hop, i.e. the time it takes for a packet to travel from the source to the destination, assuming that it is never queued behind other packets. As queuing is a component (and often the dominant one) of the end-to-end delay, especially with VBR traffic, there is no guarantee that such algorithms can actually find a delay-feasible schedule if there exists one.

Within the second category, [52] gives a priori guarantees, whereas [75] uses admission control to check whether the required TDMA delay is feasible. [68] considers both CBR (voice) and VBR (video) flows, however assuming that VBR sources can be described as stationary, ergodic and independent processes with known statistics, so as to characterize them as equivalent CBR sources. In this thesis, we deliberately omit this kind of assumptions, sticking instead to more practical σ, ρ characterizations, which can be conveyed to the network using standard signaling protocols (e.g., RSVP, [11]).

Some works not falling into either of the above categories are also relevant, as they provide frameworks for computing delay bounds a posteriori, after link scheduling has been planned. In [58] authors define the odd/even link activation and routing framework, and employ internal scheduling policies at each link so that the end-to-end delay bound along a path is roughly double than the one obtained in a wired network of the same topology. Authors of [36] show that using throughput-optimal link scheduling and Coordinated-EDF to schedule packets within each link, rate-proportional delay bounds with small additive constants are achieved. Our goal is instead to have prespecified, arbitrary delay bounds respected through link scheduling.

As a final observation, we remark that the approach pursued in this thesis lends itself to some generalizations. As shown in [14], when analyzing WMNs with per-flow and per-path scheduling, a sink-tree routing is not a requirement. The same framework, including the problem formulation, optimal and heuristic solution strategies, can be employed with any topology, provided that i) the conflict graph , and ii) routes are given. On the other hand, in a per-node framework no closed-form delay bound is available except for a sink-tree topology [48].

3.2 Network model

The framework developed in this chapter relies on basic Network Calculus concepts, i.e. arrival curve, service curve and delay bound. Interested readers can find background in [45], from which we also borrow notation.

We assume that each mesh router is equipped with a single time-slotted channel. Transmission slots of a fixed duration T_s are grouped into frames of N slots, periodically repeated every $N \cdot T_S$ time units. This happens, for instance, in 802.16 networks, where the frame length is usually set to 5 ms. Each slot is assigned to a set of non-interfering links through conflict-free link scheduling. At every slot, a subset of links may be activated for transmission only if no conflicts occur at the intended receivers. The WMN is modeled through a connectivity graph, $G = (V, E)$, whose nodes $V = \{v_1, \cdots, v_n\}$ are mesh routers and whose edges $E = \{e_1, \cdots, e_m\}$ are directed links connecting nodes in the transmission range of each other. We assume that each link e has a constant transmission rate W_e.

As already anticipated, our WMN has a sink-tree (or multi-point to point) logical topology, as shown in Figure 3.1, where flows entering a generic node travel towards the root node. The latter is possibly connected to a wired infrastructure, serving as a gateway to the Internet. We do not take into account the presence of downlink traffic. However, as we will show later on, this is not a limitation of the proposed framework. Given the tree structure of the network and the fact that a mesh router is equipped with a single channel, the node label can be used to address both the node and its output link without ambiguity. In such a network, a path P_i is a loop-free sequence of N_i nodes, from ingress node i to the egress one. In order to denote a node's position in a path, we define function $l_i(h)$ that returns the label of the h^{th} node in path P_i if $1 \leqslant h \leqslant N_i$ (e.g. $l_i(1) = i$), and function $p_i(z)$ the "inverse" function that returns the position of node z along path P_i, $l_i(p_i(z)) = z$. Figure 3.1 shows a sink tree with 10 paths defined. Without any loss of generality, we can assume that nodes are labeled so that any path is an increasing sequence of labels.

Paths in the sink tree are traversed by flows, i.e. distinguishable streams of traffic. Each flow has a delay constraint, specified as a required end-to-end delay bound δ. At the ingress node, its arrivals are constrained by a leaky-bucket shaper, with a burst σ and a sustainable rate ρ.

As far as buffering is concerned, we consider three different options, shown in Figure 3.2: a per-flow queuing framework, where packets of each flow are buffered separately at each link. Thus, a link handles as many queues as the flows traversing it. Alternatively, in a per-path queuing framework, packets of flows traversing the same path, i.e. the same set of links, are buffered in a single queue. This way, a link handles as many queues as there are paths traversing it. As a third option, we consider per-node queuing, where only one queue per node exists, where all the traffic waiting to be transmitted is stored. In all three cases, we assume that buffers are FIFO. The purpose of our work is to describe link scheduling algorithms that compute

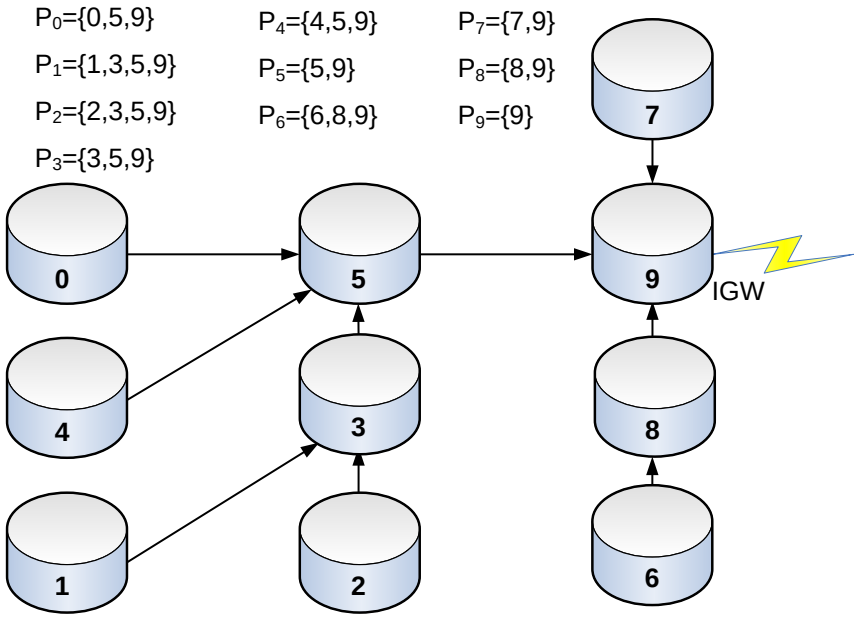

Figure 3.1. Paths in a sink tree

a conflict-free schedule which does not violate the required delay bounds whenever it is possible to do so. Hereafter, we identify the constraints that ensure the conflict-free property, which are common to all three formulations. Delay feasibility constraints, instead, depend on the queuing framework, and they lead to different problem formulations, hence we defer treating them to the next section.

The physical interference phenomenon is modeled by means of the widely used protocol interference model ([34, 3, 23, 36, 13]). For each edge of the network $e \in E$ we define a conflicting set of edges $\mathcal{I}(e)$ which includes all the edges belonging to E which can interfere with ($\mathcal{I}(e)$ contains e itself); the interference condition is straightforwardly defined as follows:

$$\sum_{i \in \mathcal{I}(e)} x_i\,(t) \leqslant 1, \text{ if } e \text{ is active in slot } t = 1, 2, ..., N,$$

where $x_e(t)$ is a binary variable, such that $x_e(t) = 1$ if link $e \in E$ is active in slot t, and 0 otherwise. The condition requires that, if edge e is active in slot t, $\mathcal{I}(e)$ contains one active edge only (the edge e itself). We translate the interference condition to a

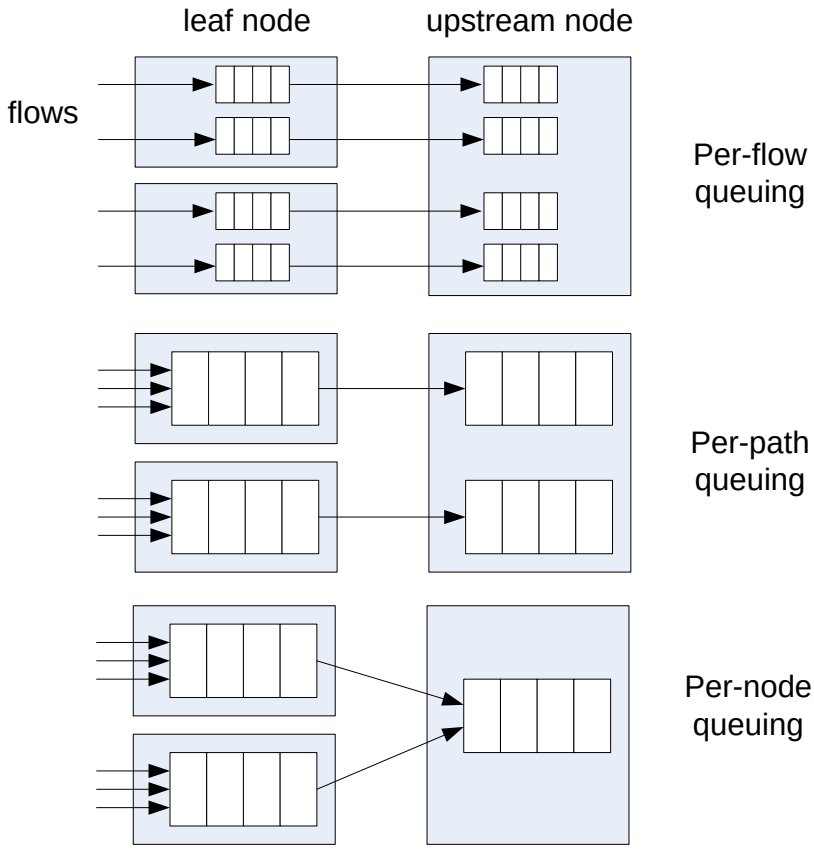

Figure 3.2. Different buffering frameworks

conflict graph $G_c = (E, C)$, shown in Figure 3.3, whose nodes represent links of the connectivity graph and whose edges $C = \{c_1, \ldots, c_r\}$ model the conflicts between links.

Half-duplex constraints are implicitly accounted for into the interference constraints, links being unidirectional. Hence a set $\mathcal{I}(e)$ can be easily obtained by retrieving the one-hop neighborhood of e in the conflict graph, e.g. for Figure 3.3 we have

$$\mathcal{I}((7,8)) = \{(4,7), (5,8), (8,7)\}.$$

Given a conflict graph C, only conflicts between active links, i.e. those with a non-null flow, have to be considered. We thus define $C_f \subseteq C$ as the subset of conflicts involving active links:

$$C_f := \{(i,j) \in C : f_i > 0 \text{ and } f_j > 0\},$$

where f_i denotes the flow going trough link i.

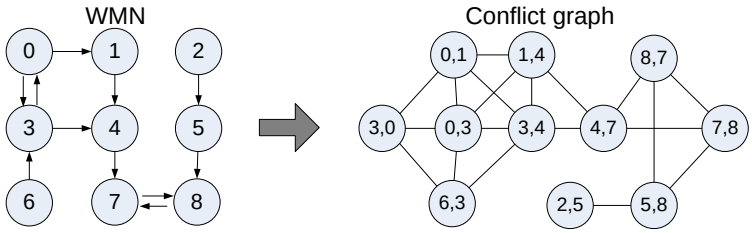

Figure 3.3. Conflicts in a TDMA network

Following the notation in [13], we define an activation offset π_e for link e, $0 \leqslant \pi_e \leqslant N$, and a transmission duration Δ_e the link is allowed to transmit for. Figure 3.4 shows the relevant quantities, plus others that will be defined in the following. Since time is slotted, π_e and Δ_e variables are non negative integers. The fact that a link has one transmission opportunity within a frame, though limiting, ensures that link scheduling maps can be kept compact.

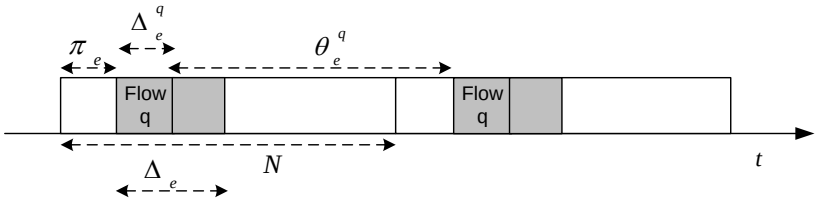

Figure 3.4. Relevant quantities in link scheduling

The schedule must ensure the conflict-free condition: while a link is transmitting, all conflicting links must refrain from transmitting. For any pair of links i and j which are neighboring nodes in C_f we have:

- if j transmits after i, it must wait for i to complete the transmission, i.e.

$$\pi_i - \pi_j + \Delta_i \leqslant 0.$$

- Otherwise, the symmetric inequality holds, i.e.

$$\pi_j - \pi_i + \Delta_j \leqslant 0.$$

In order to linearize the combination of the above constraints, we introduce a binary variable o_{ij}, $(i, j) \in C_f$, which is 1 if i transmits after j, 0 otherwise. The left-hand

side of the previous constraints can thus be upper bounded by N regardless of the relative transmission order, as π_i and Δ_i belong to $[0, N]$. This completes the formulation of the conflict-free constraints, which are necessary and sufficient conditions:

$$\begin{array}{ll} \pi_i - \pi_j + \Delta_i \leqslant N \cdot o_{ij} & \forall (i,j) \in C_f \\ \pi_j - \pi_i + \Delta_j \leqslant N \cdot (1 - o_{ij}) & \forall (i,j) \in C_f \end{array} \tag{3.1}$$

For a schedule to be valid, each link must also complete its transmission within the frame duration, i.e.:

$$\pi_i + \Delta_i \leqslant N \quad \forall i \in E. \tag{3.2}$$

3.3 Problem formulation

Beside those arising from interference, additional constraints are needed to keep into account the end-to-end delay requirements. In this section we expose them and formulate the problem of delay-constrained link scheduling. We first deal with per-flow and per-path queuing frameworks, between which many similarities exist, and then describe the problem under per-node queuing in Section 3.3.2.

3.3.1 Per-flow and per-path queuing

In per-flow queuing, each link e transmits traffic of several flows on each activation. We can therefore partition the link's Δ_e among them, i.e. $\Delta_e = \sum_{q:e \in P_q} \Delta_e^q$. Δ_e^q is the link activation quota reserved for flow q, which need not be an integer, since when a link e is activated it can switch among backlogged queues regardless of slot boundaries. We assume that backlogged flows traversing e are always served in the same (arbitrary) local order, and we call I_e the ordered set of the flow indexes. We assume that each backlogged flow q is served for no less than Δ_e^q. If a flow is idle, its service time can be exploited by other backlogged flows at e, as long as the transmission from any flow z starts within at most $\sum_{x \in I_e : x < z} \Delta_e^x$ from the activation of link e.

Therefore, flow q has a guaranteed rate equal to $R_e^q = W_e \cdot \Delta_e^q / N$ at link e, W_e being the transmission rate of that link. Since each flow has a single transmission opportunity in a frame, then the maximum inter-service time for that flow is

$$\theta_e^q = (N - \Delta_e^q) \cdot T_S, \tag{3.3}$$

irrespective of the local ordering at each link. Thus, each link of a mesh router is a rate-latency server [45] for the flows traversing it, with a rate R_e^q and a latency θ_e^q. Accordingly, each flow has an end-to-end delay bound equal to (see [45]):

$$D_q = \begin{cases} \sum_{e \in P_q} \theta_e^q + \sigma_q / R_{\min}^q & \text{if } \rho_q \leqslant R_{\min}^q \\ \infty & \text{otherwise} \end{cases} \tag{3.4}$$

where

$$R^q_{\min} \triangleq \min_{e \in P_q} \{R^q_e\}\,.$$

The above bound is tight, i.e. D_q is actually the maximum delay under the hypotheses [45]. The first addendum in (3.4) is called latency delay, and it is due to link scheduling and arbitration of the flows at the links. The second is called burst delay, and it is the time it takes for the flow burst to be cleared at the minimum guaranteed rate.

Given a traffic characterization, our aim is to find a conflict-free schedule which is also feasible from a delay point of view. To achieve this, we should solve the following end-to-end delay feasibility problem (E2EFP):

$$\begin{array}{lll}
\text{find} & o_{ij}, \pi_e, \Delta_e, \Delta^q_e & \\
\text{s.t. :} & D_q \leqslant \delta_q & \forall q \in Q \\
& \Delta_e = \sum_{q:e \in P_q} \Delta^q_e & \forall e \in E \\
& \pi_i - \pi_j + \Delta_i \leqslant N \cdot o_{ij} & \forall (i,j) \in C_f \\
& \pi_j - \pi_i + \Delta_j \leqslant N \cdot (1 - o_{ij}) & \forall (i,j) \in C_f \\
& \pi_e + \Delta_e \leqslant N & \forall e \in E \\
& o_{ij} \in \{0,1\} & \forall (i,j) \in C_f \\
& \pi_e, \Delta_e \in \mathbb{N}^+_0 & \forall e \in E \\
& \Delta^q_e \in \mathbb{R}^+_0 & \forall e \in E, \forall q \in Q
\end{array} \tag{3.5}$$

Problem (3.5) is a feasibility problem, i.e. it is solved if a feasible solution is found. In order to have a measure of the quality of the solution that we compute, we transform it into a min-max optimization problem, whose objective function (to be minimized) is the maximum delay violation

$$V_{\max} \triangleq \max_{q \in Q} \{D_q - \delta_q\}\,.$$

We will call this problem the Minimum Max Violation Problem (MinMVP):

$$\begin{array}{lll}
\min & V_{\max} & \\
\text{s.t. :} & D_q - \delta_q \leqslant V_{\max} & \forall q \in Q \\
& R^q_{\min} \leqslant W_e \cdot \Delta^q_e / N & \forall e \in P_q, \forall q \in Q \\
& \Delta^q_e \geqslant N \cdot \rho_q / W_e & \forall e \in P_q, \forall q \in Q \\
& \Delta_e \geqslant \sum_{q:e \in P_q} \Delta^q_e & \forall e \in E \\
& \pi_i - \pi_j + \Delta_i \leqslant N \cdot o_{ij} & \forall (i,j) \in C_f \\
& \pi_j - \pi_i + \Delta_j \leqslant N \cdot (1 - o_{ij}) & \forall (i,j) \in C_f \\
& \pi_e + \Delta_e \leqslant N & \forall e \in E \\
& o_{ij} \in \{0,1\} & \forall (i,j) \in C_f \\
& \pi_e, \Delta_e \in \mathbb{N}^+_0 & \forall e \in E \\
& R^q_{\min} \in \mathbb{R}^+_0 & \forall q \in Q \\
& \Delta^q_e \in \mathbb{R}^+_0 & \forall e \in P_q, \forall q \in Q
\end{array} \tag{3.6}$$

In the MinMVP we linearize the $\min$ operator in 3.4 by means of an additional continuous variable $R^q_{\min}$ for each flow.

Note that the 3rd constraint in 3.6 guarantees that all delays are finite, since enough rate is reserved for each flow at each link. Furthermore, the 2nd constraint in 3.6, i.e.

$$R^q_{\min} \leqslant W_i \cdot \Delta^q_i / N,$$

will be active for the flow q with the maximum violation, i.e.

$$R^q_{\min} = \min_{e \in P_q} \{W_e \cdot \Delta^q_e / N\},$$

as $V_{\max}$ inversely depends on $R^q_{\min}$.

Clearly, the set of feasible solutions for 3.5 is the set of points where the objective function in 3.6 is non positive. The MinMVP formulation leads to a Mixed Integer Non-Linear (MINLP) problem, with convex non linear constraints, that can be solved optimally using a general purpose MINLP solver [10]. An equivalent formulation as a Quadratically Constrained Problem (QCP) is also possible, which allows it to be solved using mixed integer QCP solvers [17].

Rewriting the feasibility problem (i.e., the E2EFP) as an optimization problem (i.e., the MinMVP), bears considerable advantages. First of all, we will show that it allows us to explore the schedulability region, assessing the relationships between the schedulability and the various parameters involved (i.e., flow deadlines, burst, rates). To this end, $V_{\max}$ is a good indicator of how much the WMN is loaded, i.e. whether it might support more traffic or tighter deadlines (or, if $V_{\max}$ is positive, which flow is the most critical). Second, as shown in [14], it yields robust schedules, such that relatively large variations in the flow bursts and rates can often be accommodated without having to compute a schedule anew.

If per-path queuing is used, instead, a set of flows $q_1, ..., q_k$ traversing the same path can be modeled as a single flow. In fact, a well-known result regarding leaky buckets is the following:

Property 1. *If two leaky-bucket shaped flows 1 and 2 are aggregated at a node, then their aggregate is still a leaky bucket shaped flow, with parameters $\sigma = \sigma_1 + \sigma_2$ and $\rho = \rho_1 + \rho_2$. Furthermore, the delay bound 3.4 computed for the aggregate flow is in fact the worst-case delay for each flow.*

This means that all the above modeling and the formulation of the MinMVP still hold, provided that the flow characteristics and requirements are composed as follows:

- the required delay bound for the aggregate is

$$\delta = \min_{1 \leqslant q \leqslant k} \{\delta_q\};$$

- the leaky bucket parameters for the aggregate are

$$\sigma = \sum\nolimits_{1 \leqslant q \leqslant k} \sigma_q, \quad \rho = \sum\nolimits_{1 \leqslant q \leqslant k} \rho_q.$$

From the network management standpoint, under per-path queuing the number of queues managed at each link is reduced with respect to the per-flow case, due to the fact that several flows are aggregated.

In [23], a WMN is modeled as a stop-and-go system. A min-max problem on the round-trip TDMA delay introduced by the scheduling in a sink-tree network is formulated and optimally solved. However minimizing the TDMA delay is not the same thing as computing delay-feasible schedules. To make this more clear, we compare our schedules with the optimal ones derived from [23]. In that work, the activation of each link is computed based on the rate of the flows traversing it, and activations are sequentialized so as to minimize the maximum TDMA delay. Consider a 15-node binary tree, with homogeneous traffic and 20 flows originating at each node. Fix $\delta = 20$, $\rho = 300$, and let the burst of the flows vary as $0 \leqslant \sigma \leqslant 4500$. We plot the value for $V_{\max}$ obtained by: i) optimally solving the MinMVP in the per-flow, per-path and per-node frameworks, and ii) using the optimal solutions given by [23] in the same settings. As Fig. 3.5, shows, the above traffic is always scheduled in such a way as to have large, positive violations according to [23]. More to the point, this happens even with a null burst (i.e., in the most favorable case). On the other hand, the above traffic is perfectly schedulable under per-path, per-node and (except for the largest bursts) per-flow aggregation if we stick to our problem formulation. This is because [23] optimizes only conflict orientations (o_{ij}) and activation instants (π_e), neglecting the activation durations (Δ_e, Δ_e^q), i.e. renouncing trading rate for delay. Our work instead explores the other extreme of the rate-delay trade-off by allocating resources based on the delay.

3.3.2 Per-node queuing

Under per-node queuing, all flows are buffered in a First-Come-First-Served way in the same queue at each node, i.e. they are aggregated as they progress towards the gateway node. Each link e is therefore activated once in the frame for an activation Δ_e. Therefore, each link is itself a rate-latency server, with a guaranteed rate equal to $R_e = W_e \cdot \Delta_e / N$ and a latency $\theta_e = (N - \Delta_e) \cdot T_S$. In this section we describe the formulas for computing the worst-case delay for a flow in a sink-tree network of rate-latency nodes when per-node queuing is in place, i.e. traffic is buffered FIFO in a single queue. The complete derivation process is shown in [49], to which the interested reader is referred for the details. Let us first introduce two preliminary results:

Theorem 2. *([49], Theorem 4.15) Consider a node x and let Ξ be the set so that $x = l_i(h_i)$ for each node $i \in \Xi$ and some $1 \leqslant h_i \leqslant N_i$. Let σ_i, ρ_i be the leaky-bucket parameters for the fresh flow entering node i. Then, the aggregate flow at the output of node x is leaky-bucket shaped, with a burstiness s_x and a sustainable rate r_x as follows:*

$$s_x = \sum\nolimits_{i \in \Xi} \left[\sigma_i + \rho_i \cdot \sum_{1 \leqslant j \leqslant h_i} \theta_{l_i(j)} \right], \; r_x = \sum\nolimits_{i \in \Xi} \rho_i, \tag{3.7}$$

and the values in 3.7 are tight output constraints at x.

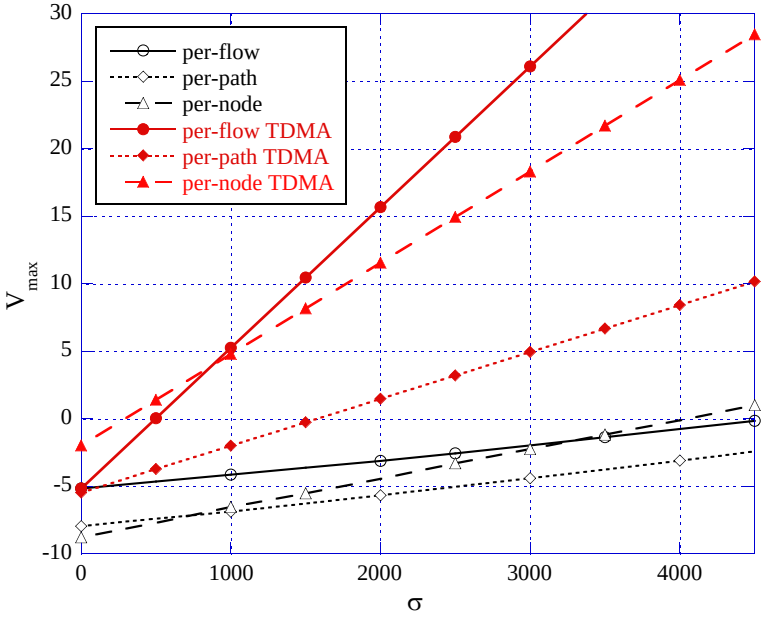

Figure 3.5. Comparison between optimizing on $V_{\max}$ and minimizing the maximum TDMA delay

Let us now consider a sink tree as the one shown in Figure 3.6, and let us focus on path P_i. First of all, although an arbitrary number of flows can traverse that path (i.e. enter at the same node), we do not need to distinguish them. In fact, by Property 1, we can describe their aggregate at the ingress of the path as a single leaky-bucket shaped flow. In order to guarantee each single flow end-to-end delay bound, the worst-case delay experienced by any bit of the aggregate cannot exceed the minimum of the delay bounds required by each single flow. Therefore we can assume without any loss of generality that one flow traverses path P_i, i.e. we have one fresh flow per node. Accordingly, we denote with σ_i, ρ_i the leaky-bucket parameters of that flow and with δ_i its required delay bound. If no fresh flow is injected at node i, we can assume that a "null flow", with $\sigma_i = 0$, $\rho_i = 0$, $\delta_i = +\infty$, is injected in the network at that node.

Based on Property 1 and Theorem 2, we can also model the aggregate traffic that joins path P_i at node $l_i\,(h)$, composed of both the flow arriving from upstream nodes and the fresh flow injected at node $l_i\,(h)$ itself, as a single flow. We call it the interfering flow (i, h), and we denote its leaky-bucket parameters as $\sigma_{(i,h)}, \rho_{(i,h)}$. The following property shows how to compute the leaky-bucket parameters of an interfering flow from node parameters:

Property 3. *In a path P_i, for $2 \leqslant h \leqslant N_i$, it is:*

$$\sigma_{(i,h)} = s_{l_i(h)} - \left[s_{l_i(h-1)} + r_{l_i(h)} \cdot \theta_{l_i(h)}\right],$$
$$\rho_{(i,h)} = r_{l_i(h)} - r_{l_i(h-1)}.$$

Note that, in general, although for two different paths P_i and P_j, $l_i(h) = l_j(k)$, interfering flows (i,h) and (j,k) may not be the same (hence we need a pair of subscripts for denoting them). In fact, from Property 3, given a node $x = l_i(h) = l_j(k)$, $(i,h) \equiv (j,k)$ if and only if there exist a node $y < x$ such that $y \in P_i, P_j$. In the network of Figure 3.1 (a portion of which is shown in more detail in Figure 3.6), we can see that paths P_0 and P_3 merge at node $5 = l_0(2) = l_3(2)$ and $(0,2) \neq (3,2)$ (both being easily identifiable through color codes in the figure). Furthermore we define flow $(i,1)$ as the sum of the output flows at all children of node i (if there are any) and the fresh traffic entering node i. For instance, at a leaf node, $\sigma_{(i,1)} = \sigma_i$and $\rho_{(i,1)} = \rho_i$.

Having said this, we now show how to compute the worst-case delay for a flow along a path. First of all, in order for queues not to build up indefinitely at a node x, the following stability condition must be ensured:

$$r_x^* = R_x - r_x \geqslant 0, \tag{3.8}$$

where r_x^* is called the residual rate of node x, i.e. the rate which is not strictly necessary to sustain the admitted traffic. If holds for all nodes along path P_i, the worst-case delay for the flow traversing that path is upper bounded by:

$$D_i = \sum\nolimits_{h=1}^{N_i} \left[\theta_{l_i(h)} + \frac{\sigma_{(i,h)}}{CR_{l_i(h)}}\right], \tag{3.9}$$

where $CR_{l_i(h)}$ is the clearing rate at node $l_i(h)$. The latter is the rate at which a burst arriving at once at that node $l_i(h)$ leaves the egress node under a worst-case scenario.

In general, $CR_{l_i(h)}$ is a function of both the service rate $R_{l_i(k)}$ and the sustainable rate of interfering flows $\rho_{(i,k)}$ at nodes $h \leqslant k \leqslant N_i$. It can be computed once it is known which nodes act as bottlenecks for node $l_i(h)$, according to the following definition.

Definition 4. *Consider two nodes x and y, such that path P_i traverses them in that order, i.e. $p_i(x) \leqslant p_i(y)$. Then, we say that y is a bottleneck for x if:*

$$r_y^* \leqslant \min\left\{r_j^* : p_i(x) \leqslant p_i(j) < p_i(y)\right\}. \tag{3.10}$$

Intuitively, node y is a bottleneck for node x if its residual rate is the minimum among all nodes in the path from x to y. Note that, by definition, x is a bottleneck for itself. Call

$$B_x = \left\langle b_1^x, b_2^x, ...b_{W_x}^x \right\rangle$$

the sequence of $W_x \geqslant 1$ bottleneck nodes for node x, sorted in the same order as they appear in any path that traverses that node, so that $b_1^x = x$. Then, it is:

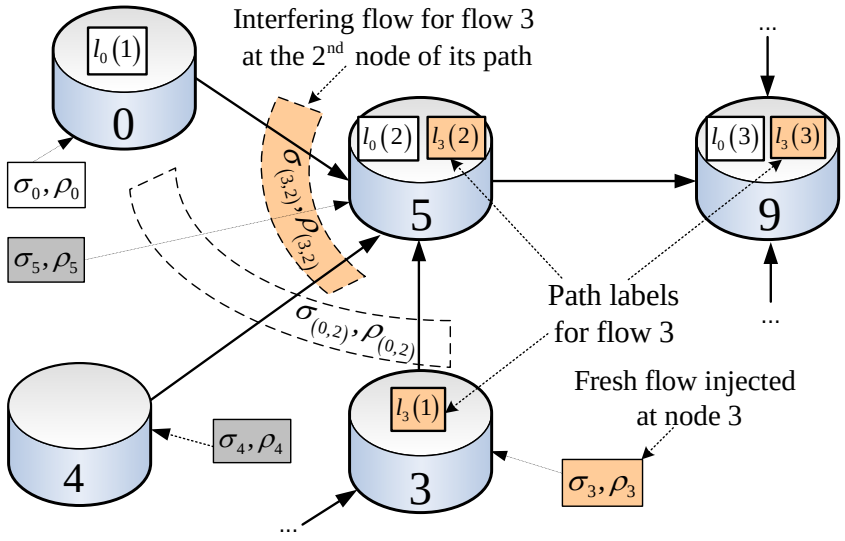

Figure 3.6. Relevant quantities for paths P0 and P3

$$CR_x = R_{b^x_{W_x}} \cdot \prod_{y=1}^{W_x-1} \frac{R_{b^x_y}}{R_{b^x_y} + \left(r_{b^x_{y+1}} - r_{b^x_y}\right)}. \tag{3.11}$$

Note that, since the sequence of bottlenecks depends on rate allocation at the links, (3.11) and, accordingly, (3.9) are non-smooth. Non-smooth functions are very hard to treat, especially on less than very small scales, within optimization problems. However, an alternative formulation of (3.11) can be given as a minimum of smooth functions, which allows us to formulate our problem in a more tractable way. The property is given by the following theorem:

Theorem 5. *Consider a sink-tree path P_x and define: $\Sigma_x = \{S : S \subseteq P_x, x \in S\}$ i.e., the set of those subsets which also include x (note that $B_x \in \Sigma_x$). Denote with $n_S(h)$ the hth node in S, with $n_S(1) = x$. Then it is:*

$$CR_x = \min_{S \in \Sigma_x} \left\{ R_{n_S(|S|)} \cdot \prod_{h=1}^{|S|-1} \frac{R_{n_S(h)}}{R_{n_S(h)} + \left(r_{n_S(h+1)} - r_{n_S(h)}\right)} \right\} \tag{3.12}$$

Proof. Consider a generic sequence of nodes S in Σ_x, and compute the following expression:

$$Q_S = R_{n_S(|S|)} \cdot \prod_{h=1}^{|S|-1} \frac{R_{n_S(h)}}{R_{n_S(h)} + \left(r_{n_S(h+1)} - r_{n_S(h)}\right)}.$$

We show that i) adding a bottleneck node to -, and ii) removing a non-bottleneck node from S leads to a smaller value for Q_S. This proves the thesis, as

$$CR_x = \min_{S \in \Sigma_x} \{Q_S\}.$$

In order to simplify the notation, we drop the path subscript in the proof, without this generating ambiguity.

i) Build the sequence $S' = S \cup \{b\}$, where $b \in B \backslash S$. Now, either b is the last node in S', or it is not. In the first case, call l the last node in S. After some straightforward algebraic manipulation, we obtain:

$$Q_{S'} = Q_S \cdot \frac{R_b}{R_l + (r_b - r_l)}.$$

However, since $b \in B$, the last term is smaller than or equal to 1 (by the very definition of bottleneck), hence $Q_{S'} \leq Q_S$ If, instead, b is not the last node in S', then there exist two nodes in S, call them l, m, such that $l < b < m$. Therefore:

$$\begin{aligned} Q_S &= R_{n(S,1)} \cdot \prod_{h=1}^{|S|-1} \frac{R_{n_S(h+1)}}{R_{n_S(h)} + \left(r_{n_S(h+1)} - r_{n_S(h)}\right)} \\ &= \Delta \cdot \frac{R_m}{R_l + (r_b - r_l) + (r_m - r_b)} \end{aligned}$$

for some positive Δ, and

$$Q_{S'} = \Delta \cdot \frac{R_b}{R_l + (r_b - r_l)} \cdot \frac{R_m}{R_b + (r_m - r_b)}.$$

Therefore, in order to prove that $Q_{S'} \leq Q_S$, we need to prove that:

$$\frac{R_b}{R_l + (r_b - r_l)} \cdot \frac{1}{R_b + (r_m - r_b)} \leq \frac{1}{R_l + (r_b - r_l) + (r_m - r_b)}.$$

After simple algebraic manipulations, the above expression boils down to:

$$\frac{R_b}{R_b + (r_m - r_b)} \leq \frac{R_l + (r_b - r_l)}{R_l + (r_b - r_l) + (r_m - r_b)},$$

which is equivalent to

$$R_b \leq R_l + (r_b - r_l).$$

The latter is true since b is a bottleneck, hence $Q_{S'} \leq Q_S$ in any case.

ii) Consider now a sequence S that includes all bottleneck nodes, i.e. $B \subseteq S$, and at least a node $f \in S \backslash B$ (i.e., a non bottleneck). Call $S' = S \backslash \{f\}$. We show that $Q_{S'} \leq Q_S$. Assume first that the node preceding f, call it b, is a bottleneck (we will show later on that this comes with no loss of generality). Note that, since $1 \in S \cap B$, one such node exists for sure. Now, either f is the last node in S, or it is not. In the first case, call we have:

$$Q_S = Q_{S'} \cdot \frac{R_f}{R_b + (r_f - r_b)},$$

and the last term is larger than 1 since f follows a bottleneck, hence $Q_{S'} \leq Q_S$. In the second case, call m the next node in S following f, i.e. $b < f < m$. Then:

$$Q_S = \Delta \cdot \frac{R_f}{R_b + (r_f - r_b)} \cdot \frac{R_m}{R_f + (r_m - r_f)},$$

and

$$Q_{S'} = \Delta \cdot \frac{R_m}{R_b + (r_f - r_b) + (r_m - r_f)},$$

for some positive Δ. Again, the thesis $Q_{S'} \leq Q_S$ can be easily rewritten as:

$$\frac{R_b + (r_f - r_b)}{R_b + (r_f - r_b) + (r_m - r_f)} \leq \frac{R_f}{R_f + (r_m - r_f)},$$

which holds if and only if $R_b + (r_f - r_b) \leq R_f$. However, since b is the last bottleneck before f and f is not a bottleneck, then

$$R_b + (r_f - r_b) < R_s,$$

and therefore

$$Q_{S'} \leq Q_S.$$

This proves that you can remove every first node following a bottleneck and obtain

$$Q_{S'} \leq Q_S.$$

However, by iterating the same procedure, you can progressively remove every second, third,..., n^{th} node following a bottleneck. Therefore, you can ultimately remove all non bottleneck nodes and obtain a smaller expression.
Wrapping up, if you take a generic sequence S and i) add all bottleneck nodes, and ii) remove all non-bottleneck nodes, you obtain a sequence B such that

$$Q_B = CR \leq Q_S,$$

which is the thesis.

□

Note that the set Σ_x grows exponentially with the length of path P_x, i.e. with the depth of the sink tree. However, since paths in a WMN are not expected to be longer than few hops, this is never a problem in practical cases. The following end-to-end delay feasibility problem (E2EFP) is homologous to (3.5), with the new delay formula given by (3.9):

$$
\begin{array}{lll}
\text{find} & o_{ij}, \pi_e, \Delta_e & \\
\text{s.t. :} & D_e \leqslant \delta_e & \forall e \in E \\
& \pi_i - \pi_j + \Delta_i \leqslant N \cdot o_{ij} & \forall (i,j) \in C_f \\
& \pi_j - \pi_i + \Delta_j \leqslant N \cdot (1 - o_{ij}) & \forall (i,j) \in C_f \\
& \pi_e + \Delta_e \leqslant N & \forall e \in E \\
& o_{ij} \in \{0,1\} & \forall (i,j) \in C_f \\
& \pi_e, \Delta_e \in \mathbb{N}_0^+ & \forall e \in E
\end{array}
\tag{3.13}
$$

Note that, in this case, we can use the same subscript e to denote a flow (or set thereof) and a node, since all flows are aggregated at a node. Problem 3.13 is again a feasibility problem, which we transform into a min-max optimization problem, homologous to 3.6:

$$
\begin{array}{lll}
\min & V_{\max} & \\
\text{s.t. :} & \sum_{h=1}^{N_i} \left[\theta_{l_i(h)} + \sigma_{(i,h)}/CR_{l_i(h)}\right] - \delta_e \leqslant V_{\max} & \forall e \in E \\
& CR_e \leqslant R_{n_S(|S|)} \cdot \prod_{h=1}^{|S|-1} \frac{R_{n_S(h)}}{R_{n_S(h)} + \left(R_{n_S(h+1)} - R_{n_S(h)}\right)} & \forall S \in \Sigma_e, \forall e \in E \\
& \Delta_e \geqslant \sum_{i \in E : e \in P_i} N \cdot \rho_i / W_e & \forall e \in E \\
& \theta_e = (N - \Delta_e) \cdot T & \\
& R_e = W_e \cdot \Delta_e / N & \\
& \pi_i - \pi_j + \Delta_i \leqslant N \cdot o_{ij} & \forall (i,j) \in C_f \\
& \pi_j - \pi_i + \Delta_j \leqslant N \cdot (1 - o_{ij}) & \forall (i,j) \in C_f \\
& \pi_e + \Delta_e \leqslant N & \forall e \in E \\
& o_{ij} \in \{0,1\} & \forall (i,j) \in C_f \\
& \pi_e, \Delta_e \in \mathbb{N}_0^+ & \forall e \in E \\
& CR_e \in \mathbb{R} \cap [0,1] & \forall e \in E
\end{array}
\tag{3.14}
$$

The above mixed integer-non linear formulation contains non-convex non-linear constraints. This can be easily worked out by counterexample, i.e. by constructing instances where local optimization solvers yield different optima with different starting points, which cannot happen with convex problems. Therefore, in order to solve the above optimization problem, global optimization techniques are required, involving a combination of both branch-and-bound - to handle the integrality constraints on the π_e, Δ_e and o_{ij} variables - and linearization/bound reduction techniques - to deal with the non-linear non-convex constraints.

We come back to the problem of computation efficiency in Section 3.5, after comparing the optimal schedules obtained under the different aggregation frameworks for several topologies and traffics. Hereafter, for ease of exposition, we will refer to both (3.6) and (3.14) with the name MinMVP, depending on the context. Possible ambiguities will be resolved by explicitly mentioning the related queuing framework.

3.3.3 Coping with downlink traffic

The problem formulations can be adapted to take into account the simultaneous presence of downlink traffic, i.e. from the gateway to the leaves. In this case, each of the

(uplink) links considered in the above formulations has a new conflicting link in the downlink direction. At that link, one (per-node) or more (per-path, per-flow) queues store the waiting packets. As far as delay bounds are concerned, there is no interaction between uplink and downlink paths, since traffic traverses different links. Therefore, the formulas for computing the delay bounds are the same as explained in the previous sections. While this is obvious for the per-flow and per-path cases, the derivations of Section 3.3.2, strictly speaking, are for uplink (i.e., sink-tree) paths only. However, we showed in [51] that downlink paths in a tree network can be analyzed using the same formulas as (3.9)-(3.12): the proof (which is notationally heavy) relies on the fact that, given a downlink path, we can always construct an uplink path which has the same delay bound.

Therefore, if we incorporate downlink traffic too, the only required modifications are adding the new links to the picture, i.e. i) conflicts to the conflict graph; ii) variables, such as the link activations and offsets, and iii) constraints of the same type as for the uplink case. We do not consider downlink traffic further for the sake of simplicity.

3.4 Schedulability comparison

Being able to optimally solve the MinMVP allows us to explore the solution space, i.e. to assess how the flow and system parameters affect the schedulability. We performed several experiments, solving the MinMVP problem in a per-flow, per-path and per-node queuing frameworks. We consider the 15-node balanced binary tree shown in Figure 3.7, with both homogeneous and heterogeneous flows.

First of all, the criterion of comparison is the value of the objective function $V_{\max}$. The latter, of course, depends on the flow deadlines δ_q in turn. However, when deadlines are homogeneous (i.e. $\delta_q = \delta$), the actual value of $V_{\max}$ can be expressed as $D - \delta$, with $D = \max_q\{D_q\}$ depending on the instance of the problem. Hence, while whether a given traffic demand is schedulable ($V_{\max} \leqslant 0$) or not ($V_{\max} > 0$) does depend on the actual value of δ, the fact that $V^a_{\max} \leqslant V^b_{\max}$, with a and b being two different aggregation frameworks, does not. Thus, $V^a_{\max} \leqslant V^b_{\max}$ actually implies that using a would allow for smaller values of δ (or a higher offered load) to be schedulable, and is therefore considered preferable here. Furthermore, there is no insight to be gained by varying δ as it only acts as an offset to $V_{\max}$, and for this reason we keep it constant, i.e. $\delta = 20$, until further notice.

We start with homogeneous traffic. In Figure 3.10, we plot $V_{\max}$ against the rate in a scenario with 20 flows originating at each node, for two different values of the burst ($\sigma = 0$, $\sigma = 1000$), and with different aggregations. For all flows, ρ varies in $[50; 650]$. Two main observations can be gathered. First of all, under per-flow and per-path aggregation, $V_{\max}$ depends minimally, if at all, on the flow rates, as long as the problem is solvable. This is because D_q does not depend on ρ_q in (3.4), as long as $\rho_q \leqslant R^q_{\min}$. However, the minimum $V_{\max}$ is obtained when each link has the largest possible rate. Hence, modifying the flow rates does not change anything, at least until

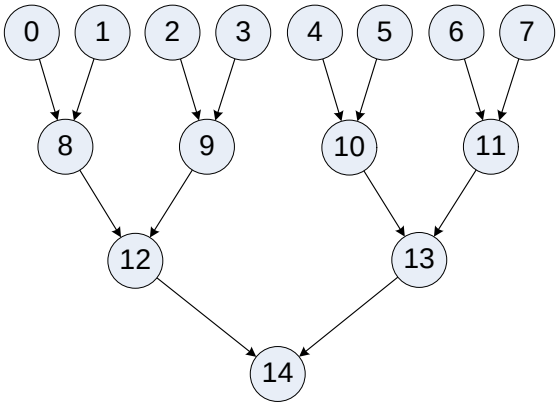

Figure 3.7. Sample binary tree

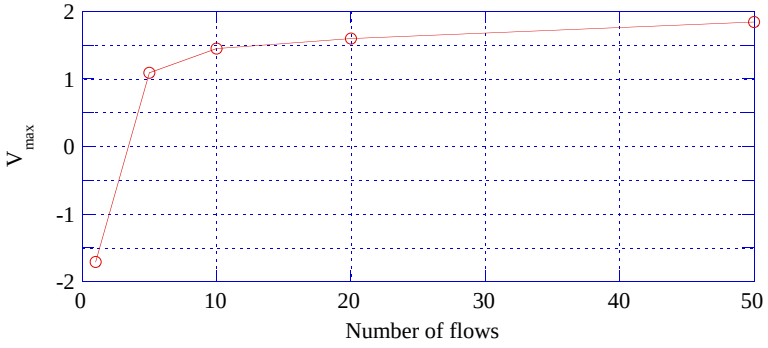

Figure 3.8. Constant overall load (homogeneous flows)

they grow so large that D_q becomes infinite. To prevent the reader from drawing hasty conclusions, we also observe that, if routing was put into the framework (i.e., in a case study where a flow has more than one path to a destination), the outcome would instead depend more heavily on the flow rates, as they would probably influence the path a flow takes. Furthermore, the performance under per-flow aggregation is always remarkably worse than under per-path aggregation. This is because, as flows are aggregated, the latency of the aggregate is generally smaller than the latency of the single flows at each node, since a larger transmission duration is given to the aggregate. This is further confirmed by the following experiment: we inject a constant load of traffic at each node (i.e., same overall σ, ρ), fractioned in 1 to 50 flows. Figure 3.8

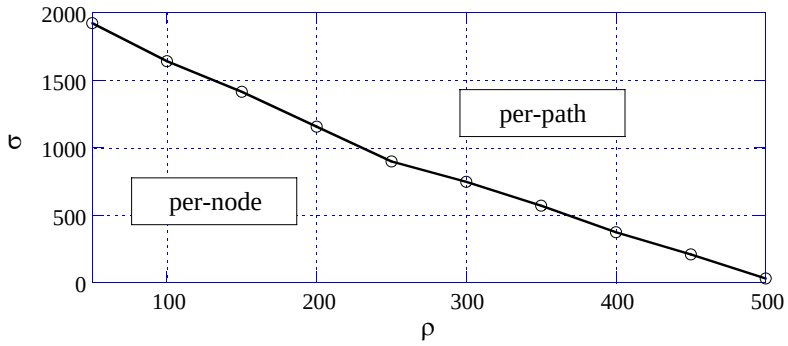

Figure 3.9. Regions of the (σ, ρ) plane where a given aggregation model leads to smaller $V_{\max}$ (homogeneous flows case)

reports $V_{\max}$ against the number of flows per node, and confirms that the gain with a per-path framework increases with the number of flows that are aggregated. Thus, aggregating a large number of smaller flows (besides leading to more manageable implementations, due to the reduced number of queues to be managed) improves the overall performance.

Second, when per-node aggregation is used, Figure 3.10 shows $V_{\max}$ being linearly increasing with the rate. The slope is almost constant, and the values of σ determines the offset. This is because rates intervene in the computation of the delay bound in (3.9) through (3.7) and (3.11). Moreover, it turns out that - for a given burst σ - there exists a value of ρ below which per-node aggregation outperforms per-path aggregation, i.e. it yields a smaller $V_{\max}$. This boundary value occurs at smaller rates as the burst increases. In Figure 3.9, we plot the curve where $V_{\max}$ has the same value in both per-path and per-node frameworks in a (σ, ρ)-plane. The curve is a decreasing line, whose best fit is the following:

$$\sigma = [-4.1364 \cdot \rho + 2035.6]^{+} \tag{3.15}$$

Below the latter, per-node aggregation yields better results. On the other hand, in the region above the curve per-path aggregation is more effective.

The same analysis was repeated for other topologies, i.e., unbalanced and ternary trees, with homogeneous traffic. In all cases qualitatively similar results were obtained, which are henceforth omitted for the sake of conciseness. More specifically, it always holds that per-flow queuing fares the worst, and that a decreasing line separates the regions of the (σ, ρ)-plane where per-node aggregation fares better than per-path aggregation. Coefficients in (3.15), however, are topology-specific.

The above considerations are also true, at least up to some extent, if we relax the assumption of homogeneous flows. Figure 3.11 reports the average $V_{\max}$ for 30 instances, on the balanced tree with heterogeneous random flows (confi-

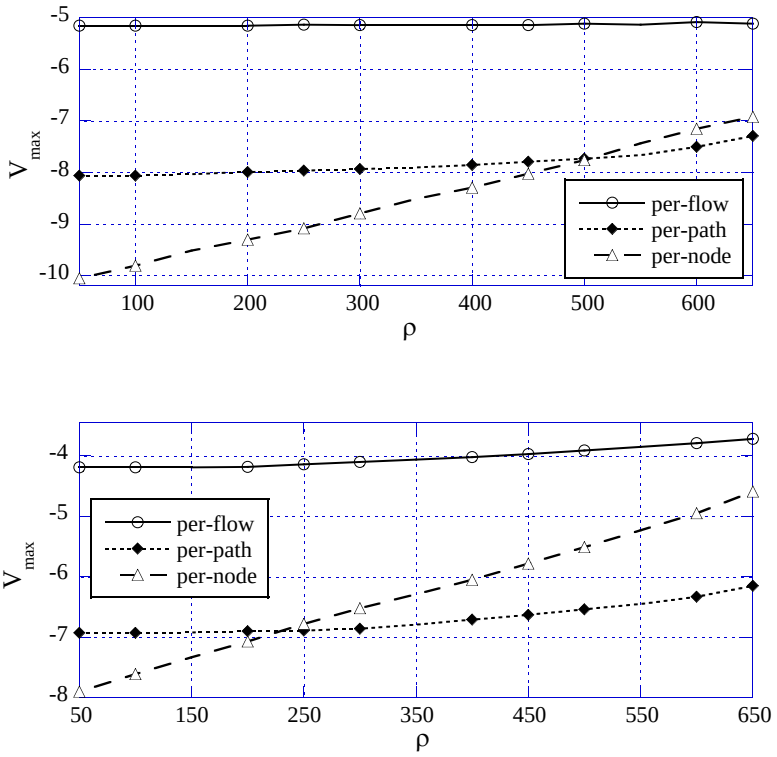

Figure 3.10. Homogeneous flows in a balanced binary tree for $\sigma = 0$ (above) and $\sigma = 1000$ (below)

dence intervals are not visible): rates and bursts are generated uniformly between $[0.8 \cdot 300/K; 1.2 \cdot 300/K]$ and $[0.8 \cdot \sigma/K; 1.2 \cdot \sigma/K]$ respectively, with σ ranging from 1500 to 6000 and $K = 20$ being the number of fresh flows originating at each link. Although the lines represent averages, $V_{\max}^{per-p} \leqslant V_{\max}^{per-f}$ holds for each instance of the problem. For instance, Figure 3.11 suggests that the maximum aggregate burst schedulable in a per-flow framework is 4500, whereas in a per-path framework it is 6200, i.e. 38% larger. This corresponds to approximately 11 additional flows per node. More to the point, the qualitative behavior does not change if we allow for a larger spread for the bursts. We have increased the above interval from $[0.8; 1.2]$ to $[0.2; 1.8]$, without experiencing any noticeable change in the outcome. Even considered non-uniform distributions for the flows, e.g. $[0.2; 0.4]$ for one half of the flows and $[1.6; 1.8]$ for the other, has no significant impact on $V_{\max}$. This seems to suggest that, as long as flows have the same deadline, aggregating heterogeneous flows improves the delay performance, and that the latter depends on the overall burst rather than on how it is distributed. On the other hand, the per-node framework with the generated instances always fares worse. It has to be observed, however, that the point corre-

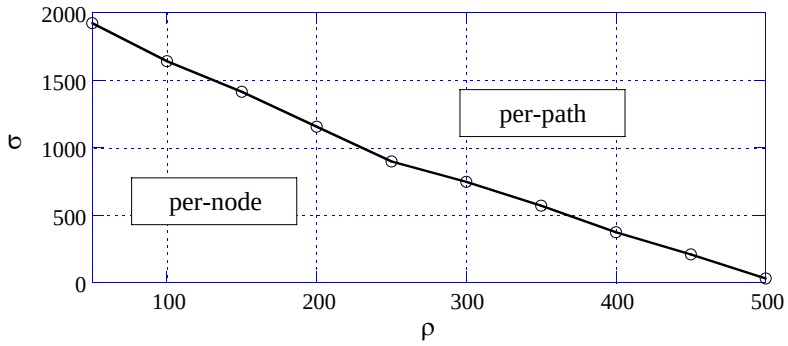

Figure 3.11. Heterogeneous randomly generated flows

sponding to the average values $(E[\sigma], E[\rho])$ in the (σ, ρ)-plane of Figure 3.9 would be located in the region where per-path aggregation performs better. Although we do not show it here for the sake of conciseness, we can select flow parameters so as to obtain the opposite outcome.

So far, we have considered homogeneous deadlines. In fact, the behavior changes if flows with different deadlines are aggregated. In that case, in fact, per-flow scheduling comes back into play. Figure 3.12 shows a case with 3 flows per node having the same σ, ρ, with $\rho = 300$ and σ ranging from 0 to 2000, but different deadlines (30, 60, and 100 respectively), on a balanced binary tree. For small bursts (i.e., below 600), per-path aggregation performs better, whereas per-flow is winning for larger bursts. This can be explained by considering that, depending on the burst size, either the latency or the burst delay may be predominant in (3.4). On one hand, as already noticed, aggregating flows always reduces their latency delay. On the other hand, tighter delay requirements are imposed on the aggregate, which instead increases the maximum violation. The first effect dominates for small bursts. Furthermore, note that in none of the above cases per-node aggregation performs better. The same considerations again apply to different topologies, such as random and ternary trees.

From the above analysis, the following conclusive remarks can be obtained:

- aggregating flows on the same path is always beneficial as long as they have the same deadline. On the other hand, it matters little whether their traffic characteristics (σ, ρ) are similar or not. For instance, real-time traffic of different types (e.g., voice and video) can be aggregated, as long as the deadlines are the same.
- Aggregating flows at each node is beneficial only when flows have the same deadlines and limited bursts (e.g., voice traffic, which is known to be non-bursty). The limit beyond which it stops being beneficial decreases with the flows rate. For instance, high-rate, bursty flows (e.g., compressed video streams) should not be aggregated at each node if delay guarantees are a concern.

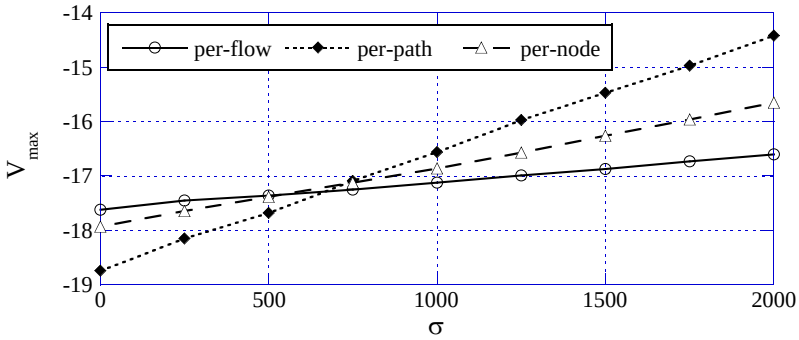

Figure 3.12. Flows with different deadlines in a balanced binary tree

- The above considerations are fairly insensitive of the actual sink-tree network topology. Regular and irregular trees exhibit few differences.

It is also evident that there is no clear winner among the three aggregation frameworks, i.e. one that it is likely to warrant a higher schedulability in all the scenarios. A clearer picture can be obtained by putting computation overhead into the framework, which is what we do in the next section.

3.5 Online admission control

The presence of integer (π_e, Δ_e) and, mostly, binary variables (o_{ij}) makes the MinMVP complex. Furthermore, as already stated, the problem is also non convex under per-node aggregation. Under per-flow and per-path frameworks, the MinMVP can be solved optimally for WMNs of few tens of nodes, which is the expected scale for current and future WMNs. Note that, somewhat counterintuitively, using a per-flow or per-path framework makes almost no difference in the solving times, unless the number of flows per path grows very large. This is due to the fact that the above choice only influences continuous variables, whereas the number of integer and binary variables stays the same in both cases. Figure 3.13, left, shows the distribution of the computation times for solving 100 instances of three different WMNs in a per-path framework, i.e. a balanced binary tree of 15 nodes, a ternary tree of 13 nodes, and a random tree of 12 nodes, using CPLEX [17]. Computations are done on a PC equipped with an Intel Core2 Duo E6400 2.1 GHz, 2 GB RAM and a Linux kernel 2.6.18. Solving larger instances (20-30 nodes) requires instead minutes or hours on the same system. These are clearly affordable times when compared to the timescales of network (re)engineering, but not so when compared to the timescale of admission control decisions. When a per-node framework is used and the problem is non convex, the maximum size that one can expect to solve using available techniques is at most 15-20

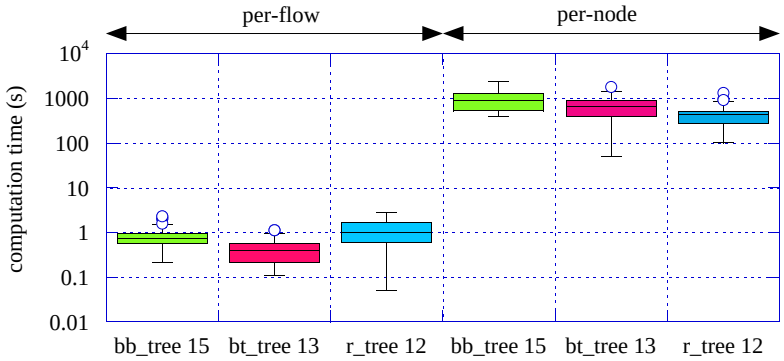

Figure 3.13. Box plot of the resolution times for different WMN topologies under per-flow and per-node frameworks

nodes, depending on both the topology and the number of flows. For larger networks, the computations may be altogether impossible due to memory constraints. Furthermore, computation times tend to be considerably higher for topologies of a similar number of nodes. Figure 3.13, right, shows the distribution of the computation times using the BARON solver [66] on 100 instances of each of the above-mentioned WMN topologies. Note that we had to use a different solver than CPLEX because the latter only solves convex problems.

For this reason, we now tackle the problem of trading optimality for computation time through heuristic approaches. We first discuss heuristics for the convex case, and show that the latter can also be adapted to work in the per-node case.

3.5.1 Heuristics for the per-flow and per-path case

As the problem is convex, the greatest computational burden comes from binary variables, i.e. conflict orientations o_{ij}. Once the latter are set, near-optimal solutions to the remaining mixed integer-convex problem can be computed in few tens or hundred milliseconds for instances of tens of nodes. Hereafter, we refer to the problem of setting the o_{ij} as the conflict subproblem, and to the resulting, simplified MinMVP once the o_{ij} are set as the reduced MinMVP subproblem. Our heuristic solution strategies rely on solving the two separately, in the above order.

More specifically, link transmission orders are represented by binary variables o_{ij}. Setting the o_{ij} variables implies orienting of the edges of the conflict graph. These variables determine the activation of the conflict-free constraints in (3.1). For a fixed conflict orientation string, i.e. given values of o_{ij}, each pair of constraints can be replaced by either of the following:

$$\begin{cases} \pi_i + \Delta_i \leqslant \pi_j \text{ if } \quad o_{ij} = 0 \\ \pi_j + \Delta_j \leqslant \pi_i \text{ otherwise} \end{cases} \qquad \forall (i,j) \in C_f \tag{3.16}$$

Before delving deeper into each subproblem, it is worth mentioning that the quality of the solution of the conflict subproblem has a remarkable impact on the overall schedulability of a given traffic load: starting with a "bad" conflict orientation will thwart any attempt to compute a feasible schedule, even if the reduced MinMVP is solved optimally. Moreover, the conflict subproblem is itself non trivial, so that trying to solve it optimally is out of question if speed is a concern.

In [14], a blind heuristic was presented, which can be used when no information about the underlying network is available. The latter relies on solving the conflict subproblem using a genetic approach, where the fitness of each conflict orientation string is given by the optimum of the reduced MinMVP problem that it generates. Results shown in [14] prove that this allows computation time to be traded for optimality: allowing for a larger number of generations increases the likelihood of hitting a value of $V_{\max}$ close to the real optimum. The trade-off is configurable, so that a criterion based on the maximum response time for admission control can be used to stop the computations. We also developed a new engineered heuristic, where an estimate of the rates that the network is expected to support is used to infer an off-line solution to the conflict subproblem. It is often the case that a network administrator can estimate the average rates that its network links should support. In this case, some of the computations needed for an admission control test can be done off-line. More specifically, the administrator can use that knowledge to solve the conflict problem, setting the o_{ij} once and for all offline, and then solve the reduced problem online, at the time of admission control. We will show that the reduced problem is accurate and only takes a few milliseconds, hence the online part is fast and effective.

Assigning the o_{ij} transforms the unoriented conflict graph into a dependency graph. Define $f_e = \sum_{q \in I(e)} \rho_q$ the overall rate of flows traversing link e. Call $\Delta_e^{LB} = \lceil f_e / W_e \cdot N \rceil$ the lower bound for the activation duration of link e, given an estimate of the average rate request f_e. If $\Delta_e < \Delta_e^{LB}$ at some link, then the problem is clearly infeasible because delays are unbounded according to (3). On the other hand, unless the following condition holds on every path P in the dependency graph:

$$\sum_{e \in P} \Delta_e^{LB} \leqslant N, \tag{3.17}$$

then the problem is infeasible. In fact, in this case there would be no way to partition the frame into activation durations while preserving bounded delays. Therefore, a good heuristic is one that allows you to set o_{ij} so that i) (3.17) holds for all paths when it is possible to do so, and ii) the duration of the activation durations are maximized, so that delays are kept as small as possible. Given a rate estimate f_e, we formulate the conflict subproblem as follows:

$$
\begin{array}{lll}
\max & \sum_{e \in E} f_e \cdot \Delta_e & \\
\text{s.t. :} & \Delta_e \geqslant N \cdot f_e / W_e & \forall e \in E \\
& \pi_i - \pi_j + \Delta_i \leqslant N \cdot o_{ij} & \forall (i,j) \in C_f \\
& \pi_j - \pi_i + \Delta_j \leqslant N \cdot (1 - o_{ij}) & \forall (i,j) \in C_f \\
& \pi_e + \Delta_e \leqslant N & \forall e \in E \\
& o_{ij} \in \{0,1\} & \forall (i,j) \in C_f \\
& \pi_e, \Delta_e \in \mathbb{N}_0^+ & \forall e \in E
\end{array}
\tag{3.18}
$$

Note that, since we have f_e in the objective playing the role of weights, solving (3.18) gives preference to the links carrying the largest rates. Once an assignment for the o_{ij} variable is obtained from the solution of the above problem, the online part consists in solving the remaining reduced MinMVP.

A fast heuristic solution approach for the reduced MinMVP (shown in Figure 3.14) relies on formulating a nonlinear convex problem starting from the MinMVP, relaxing the integrality constraints on variables π_e and Δ_e, and substituting (3.16) for (3.1), i.e.:

$$
\begin{array}{lll}
\min & V_{\max} & \\
\text{s.t. :} & D_q - \delta_q \leqslant V_{\max} & \forall q \in Q \\
& R^q_{\min} \leqslant W_e \cdot \Delta^q_e / N & \forall e \in P_q, \forall q \in Q \\
& \Delta^q_e \geqslant N \cdot \rho_q / W_e & \forall e \in P_q, \forall q \in Q \\
& \Delta_e \geqslant 1 + \sum_{q : e \in P_q} \Delta^q_e & \forall e \in E \\
& \pi_e + \Delta_e \leqslant N & \forall e \in E \\
& \pi_e, \Delta_e \in \tilde{S} & \forall e \in E \\
& \pi_e, \Delta_e \in \mathbb{R}_0^+ & \forall e \in E \\
& R^q_{\min} \in \mathbb{R}_0^+ & \forall q \in Q \\
& \Delta^q_e \in \mathbb{R}_0^+ & \forall e \in P_q, \forall q \in Q
\end{array}
\quad ,
\tag{3.19}
$$

where $\tilde{S}$ concisely denotes constraints (3.16). Then, the solutions of this reduced model $(\widetilde{\pi_e}, \widetilde{\Delta_e})$ are rounded to their integer part. Note that a "+1" is required in the 4th constraint to ensure that $\left\lfloor \widetilde{\Delta_e} \right\rfloor \geqslant \sum_{q \in I_e} \Delta^q_e$, i.e. to prevent the rounding from reducing the minimum guaranteed rate below the required one. The rounded solution is still feasible from a conflict-free point of view, since:

$$
\pi_i + \Delta_i \leqslant \pi_j \Rightarrow \lfloor \pi_i \rfloor + \lfloor \Delta_i \rfloor \leqslant \lfloor \pi_j \rfloor \tag{3.20}
$$

Finally, the rounded solution is given as an input to the following auxiliary problem:

$$
\begin{array}{lll}
\min & V_{\max} & \\
\text{s.t. :} & D_q - \delta_q \leqslant V_{\max} & \forall q \in Q \\
& R^q_{\min} \leqslant W_e \cdot \Delta^q_e / N & \forall e \in P_q, \forall q \in Q \\
& \left\lfloor \widetilde{\Delta_e} \right\rfloor \geqslant \sum_{q \in I_e} \Delta^q_e & \forall e \in E \\
& R^q_{\min} \in \mathbb{R}_0^+ & \forall q \in Q \\
& \Delta^q_e \in \mathbb{R}_0^+ & \forall e \in P_q, \forall q \in Q
\end{array}
\tag{3.21}
$$

The purpose of solving (3.21) is to compute the actual $V_{\max}$: in fact, some Δ^q_e computed in the relaxation of the reduced MinMVP might still be increased (thus further reducing $V_{\max}$) before the last constraint in (3.21) becomes active. Note that scheduling variables π_e, Δ_e are not part of (3.21) since the schedule is not modified.

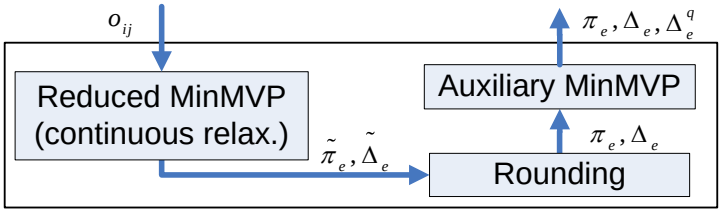

Figure 3.14. Solution scheme for the reduced MinMVP

Both (3.19) and (3.21) are convex, hence solvable in polynomial time using interior point methods. The rounding procedure runs in $\Theta(2 \cdot |E|)$ time. On the same system mentioned before, the heuristic solves 30-node instances within few tens of ms in a per-path framework, whereas optimally solving the reduced MinMVP problem takes seconds or tens thereof. As shown in Figure 3.15, which reports results on 100 random instances on a 15-node tree, the heuristic solutions appear to be very accurate. The online part takes tens to hundreds of milliseconds for large-scale WMNs (30-40 nodes), which is acceptable for most applications.

We now show that, if the rate estimate is accurate, the proposed approach is effective. We consider a balanced binary tree of 31 nodes, where we perform the offline conflict resolution assuming that each node generates $\rho = 300$ units of rate as an estimate. Then, we solve the reduced MinMVP, however using different rates than those estimated for the previous step. More specifically, we solve 100 instances, which are all feasible according to the MinMVP formulation, with homogeneous bursts and deadlines, $\sigma = 500$, $\delta = 40$. For the rates, a Gaussian distribution is assumed, with an average $avg = 300$ and an increasing standard deviation $stdev \in [1, 90]$. As Figure 3.16 shows, the percentage of instances declared unfeasible is below 10% as long as the standard deviation is below 16% of the average, a result which we consider reasonable.

3.5.2 Per-node case

Under per-node aggregation, separating the conflict and reduced MinMVP subproblems does not pay off, as the latter is non convex, and therefore still hard to solve. While general-purpose meta-heuristics for solving non-convex problems have been around for a while (e.g., multi-start methods), they all share some common features: they are hardly predictable, very dependent on parameter tuning, and - being general purpose - they seldom exploit possible underlying structures of the problem to be solved. Instead of trying to adapt the above meta-heuristics to finding suboptimal solutions in the per-node case, we take a different approach: we show that solutions of the per-path problem (whether optimal or computed through the heuristic approach)

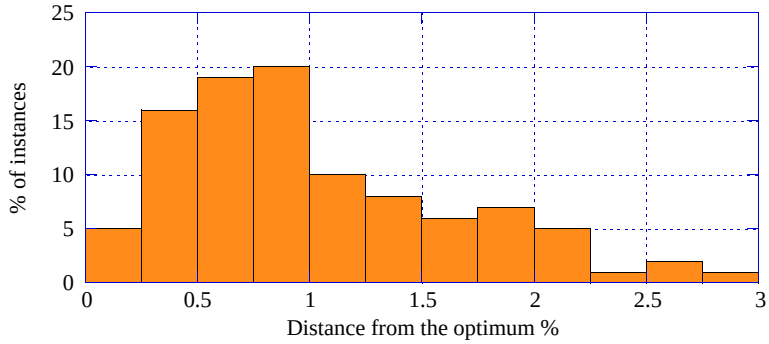

Figure 3.15. Accuracy of the reduced MinMVP problem

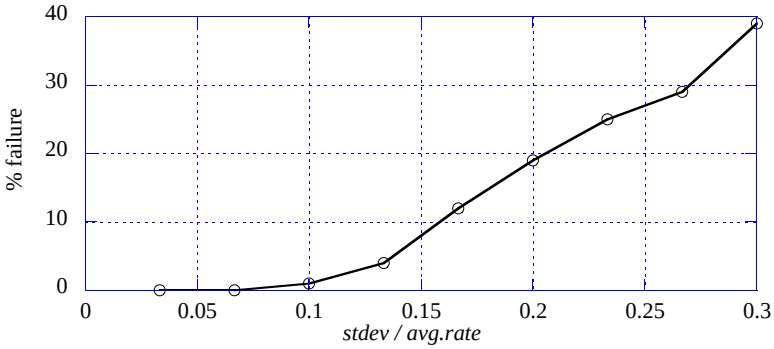

Figure 3.16. Percentage of solved instances against the standard deviation of the rate distribution

can be exploited to find good suboptimal solutions in the per-node case. Our claim is that given a solution of the per-path case in the following form:

$$\begin{cases} \overline{\pi_e} & e \in E \\ \overline{\Delta_e^q} & e \in E, q \in I(e) \end{cases}$$

the following solution, used in a per-node framework:

$$\begin{cases} \pi_e = \overline{\pi_e} & e \in E \\ \Delta_e = \sum_{q \in I(e)} \overline{\Delta_e^q} & e \in E \end{cases}, \tag{3.22}$$

yields a value of $V_{\max}$ which, though obviously suboptimal, is often reasonably close to the optimal value of the per-node case. Therefore, we can compute a good suboptimal solution for the per-node case without having to solve any non-convex problem. The above claim is supported by the results shown in Figure 3.17, where we plot the relative distance to the optimum of the heuristic obtained by using (3.22). The box

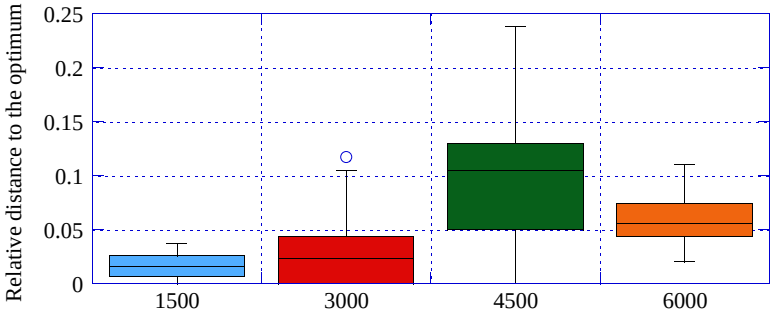

Figure 3.17. Box plot of the relative deviation between the optimal solution of the per-node case and the one computed using the engineered heuristic and (3.22).

plot is obtained by running 30 random instances of a balanced binary tree network, with rates uniformly selected in $[240, 360]$ and burst selected in $[0.8 \cdot K, 1.2 \cdot K]$, $K = 1500, 3000, 4500, 6000$. As the figure shows, the heuristic solution is seldom above 15% of the optimal value.

Note that, while computing an optimal link schedule in the per-node framework is a tough problem, a delay feasibility test for a given link schedule, which is required to check whether the solution is admissible in the per-node case, only takes $O(|E|)$ time [48]. Such time overhead is negligible with respect to the one required for computing heuristic solutions in the per-path case. Therefore, by using the above procedure, heuristics of the same efficiency as the per-flow/per-path cases can be applied to the per node case too.

As a final note, we observe that formulating the problem as a min-max problem pushes a solver to allocate the largest possible number of slots, so as to minimize the max violation. As discussed in [14], this intrinsically produces robust schedules, where relatively large variations in some flows' parameters can be tolerated without violating the deadlines, even when $V_{\max}$ is close to zero. This can be exploited in order to avoid computing a link schedule altogether in response to changes in the traffic parameters. We showed in [14] a method to assess in polynomial time whether a new computation is needed or not.

3.6 Joint routing and scheduling

In this Section we give the exact formulation of the problem of joint routing and link scheduling of leaky-bucket constrained flows that request worst-case delay guarantees. We formulate it as an optimization problem, the Delay-Aware Routing and Scheduling (DARS) problem, with the objective of minimizing the maximum deadline violation. As for the previous link scheduling formulations, whenever a solution with

a negative objective is computed, each flow will follow a route that makes it meet its deadline despite interference. We show that the problem can be optimally solved for networks of up to few nodes (e.g., a grid), though at the price of unfeasibly long computations. For that reason, we propose a suboptimal heuristics, that rely on extrapolating the link conflict serialization (LCS) from the DARS. In the LCS, sequences of conflicting link activations are statically precomputed using a coloring approach [56], so as to minimize the longest sequence. In the remaining reduced DARS, the activation of each link is computed jointly with routing, so as to minimize the maximum deadline violation. Once conflicting links are serialized, the reduced DARS problem can be solved optimally for a larger scale (e.g., a grid); beyond that scale, optimality has to be traded off for computation time.

3.6.1 The DARS model

Given the above traffic, the network topology and the conflict graph, our purpose is to find a joint conflict-free routing and scheduling which is also feasible from a delay point of view. To achieve this, we formulate the Delay-Aware Routing and Scheduling (DARS) problem, as shown in Equation 3.23.

$$
\begin{array}{lll}
\min & V_{\max} & \\
\text{s.t. :} & \sum_{e\in E}\theta_e^q + \frac{\sigma_q}{R_{\min}^q} - \delta_q \leqslant V_{\max} & \forall q\in Q \quad (i)\\
& (N-\Delta_e^q)\cdot T_S \leqslant \theta_e^q + N\cdot T_S\cdot(1-t_e^q) & \forall e\in E,\forall q\in Q \quad (ii)\\
& R_{\min}^q \leqslant \frac{W_e}{N}\cdot\Delta_e^q + (1-t_e^q)\cdot\max_{i\in E}\{W_i\} & \forall e\in E,\forall q\in Q \quad (iii)\\
& \Delta_e^q \geqslant \frac{\rho_q}{W_e}\cdot N - N\cdot(1-t_e^q) & \forall e\in E,\forall q\in Q \quad (iv)\\
& \Delta_e^q \leqslant N\cdot t_e^q & \forall e\in E,\forall q\in Q \quad (v)\\
& \Delta_e \leqslant N\cdot\sum_{q\in Q} t_e^q & \forall e\in E \quad (vi)\\
& \Delta_e \geqslant \sum_{q\in Q}\Delta_e^q & \forall e\in E \quad (vii)\\
& \pi_i + \Delta_i - \pi_j \leqslant (1-o_{ij})\cdot N & \forall i,j\in C_f \quad (viii)\\
& \pi_j + \Delta_j - \pi_i \leqslant o_{ij}\cdot N & \forall i,j\in C_f \quad (ix)\\
& \pi_i + \Delta_i \leqslant N & \forall i\in E \quad (x)\\
& \sum_{e\in OUT(v)} t_e^q - \sum_{e\in IN(v)} t_e^q = \begin{cases} 1 & \text{if } v=s(q)\\ -1 & \text{if } v=d(q)\\ 0 & \text{otherwise}\end{cases} & \forall v\in V,\forall q\in Q \quad (xi)\\
& t_e^q \in\{0,1\} & \forall e\in E,\forall q\in Q\\
& o_{ij}\in\{0,1\} & \forall i,j\in C_f\\
& R_{\min}^q,\theta_e^q,\Delta_e^q\in\mathbb{R}_0^+ & \forall e\in E,\forall q\in Q\\
& \Delta_e,\pi_e\in\mathbb{N}_0^+ & \forall e\in E,\forall q\in Q
\end{array}
\tag{3.23}
$$

The objective function to be minimized is the maximum delay violation $V_{\max}$, defined as

$$V_{\max} \triangleq \max_{q \in Q} \{D_q - \delta_q\}.$$

If the optimum is negative, then the DARS problem has a solution which is feasible from a delay point of view. There are two sets of variables, related to link scheduling (o_{ij}, π_e, Δ_e) and routing (t_e^q) decisions. As for routing, $t_e^q = 1$ if and only if flow q traverses link e. As single-path (as opposed to multi-path) routing is assumed, t_e^q are binary. Constraints (xi) ensure flow conservation at each node. Constraints (i-vii) ensure a delay-aware link scheduling. Specifically, (i) represents $D_q - \delta_q$ according to (3.4) for flow q, assuming that its delay is finite. Constrains (ii-iv) include at the right hand side terms which depend on $(1 - t_e^q)$. Those terms are computed such that, if $t_e^q = 0$, then the constraints always hold regardless of the value given to Δ_e^q, θ_e^q, $R_{\min}^q$. In other words, those constraints are inactive for those links that are not traversed by a flow. On the other hand, when $t_e^q = 1$, (ii) sets the latency according to (3.3), (iii) guarantees that $R_{\min}^q$ is the minimum guaranteed rate among all the links traversed by flow q, i.e.

$$R_{\min}^q = \min_{e: t_e^q = 1} \{W_e \cdot \Delta_e^q / N\},$$

and (iv) ensures that the activation quota for flow q is set to ensure that the delay is finite. On the other hand, constraints (v) and (vi) are active when $t_e^q = 0$, when they guarantee that Δ_e^q is forced to zero when flow q does not traverse link e. Those constraints always hold when $t_e^q = 1$, instead. Constraint (vii) relates the activation of a link with the activation quotas of each flow traversing it. Constraints (viii-x) mirror (3.1)-(3.2), and are thus related to conflict free scheduling.

The DARS problem is a Mixed Integer Non-Linear (MINLP) problem, whose non-linear constraints are convex and for which efficient general purpose MINLP solver (e.g. [35, 32]) exist. The latter can be easily reformulated as a quadratic problem by introducing auxiliary variables, which makes it possible to use the efficient solver CPLEX. Despite the quadratic formulation, the solution time of the above problem is prohibitive for mesh networks of medium to large size. For instance, CPLEX may take days to find the optimum for a 4×4 grid, and cannot solve a 5×5. For this reason, in the next section we present a heuristic approach to solve the DARS problem.

We justify the need to solve the routing and link scheduling jointly using a simple example. Figure 3.18 reports a sample 4×4 grid mesh, where four homogeneous flows need be routed from their source (nodes 0-3) to the gateway (node 15). It is $\sigma = 1000$, $\rho = 2000$, $\delta = 30$ for all flows. The link capacity is 9600 for all links except (7,11), whose capacity is 5000. The figure also reports the routes selected by the DARS (the other variables are omitted for ease of reading). A quick glance suffices to convince the reader that these routes are not shortest paths, and it takes only a little more to verify that no shortest path routing leads to a feasible link scheduling. For instance, if flow 3 were routed along its shortest path 3-7-11-15, link (7,11) should be active for at least 40% of the time, leaving no more than 60% for conflicting link

11-15, which would then be unable to support flows 1, 2, 3 together. The latter, in fact, require an activation of 62.5% on that very link just to keep their delay bounded, let alone below any requirement.

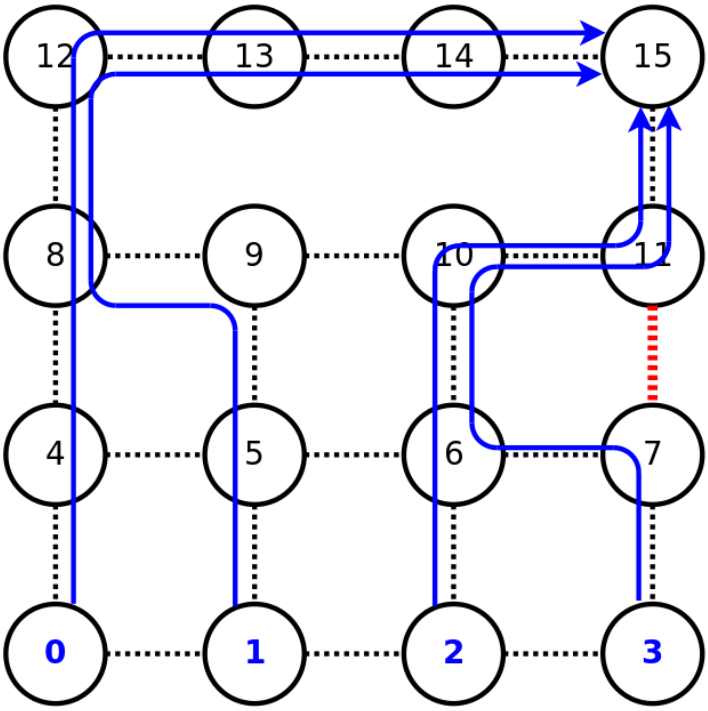

Figure 3.18. Sample mesh

3.6.2 Heuristic solutions

The high complexity of the DARS problem stems from the high number of binary variables related to conflict (o_{ij}) and routing (t_e^q). Of course, we cannot separate the routing variables without incurring in the problems outlined in the previous example. Therefore, in order to reduce the computation time, we separate the link conflict serialization (LCS) from the DARS problem. In other words, we set the o_{ij} variables offline, based on the conflict graph, and then solve the reduced DARS, where the o_{ij} are constants. As we will show later on, this allows larger-scale problems to be solved, with a negligible loss of accuracy. To increase the scale further, we also propose a Lagrangian heuristic to solve the reduced DARS (r-DARS henceforth) problem sub-

optimally. We first describe how to solve the LCS, and then we move to the r-DARS. Our solution scheme is detailed in Figure 3.19.

Link conflict serialization

The high complexity of the DARS problem stems from the high number of binary variables related to conflict (o_{ij}) and routing (t_e^q). Therefore, in order to reduce the computation time, we extrapolate the link conflict serialization (LCS) from the DARS problem. In other words, we set the o_{ij} variables offline, based on the conflict graph, and then solve the reduced DARS, where the o_{ij} are constants. As we will show later on, this comes with a remarkable reduction in the solution time.

Solving the LCS problem consists in setting the o_{ij} variables, i.e. directing the edges in the conflict graph, which in turn translates to serializing conflicting links within the frame. In fact, all the links belonging in the same clique in the conflict graph - e.g., (0,1), (1,4) in Figure 3.3 - cannot be activated in parallel, hence have to be serialized. A link may belong in up to two cliques. For instance, (0,1) also belongs to a 3-clique with (3,0) and (0,3). The objective to be pursued by the LCS is thus to minimize the maximum length path in the resulting directed conflict graph, i.e. to minimize the max number of serialized links. This can be done by employing a general K-coloring method [56]. The K-coloring is exponential in the number of vertices. However, it can be solved up to scales much larger than the ones we are dealing with, and efficient methods - e.g., based on column generation - can be exploited to achieve a fast solution time. Thus the LCS can be solved optimally, given the conflict graph. Therefore, as traffic changes, a new routing and link scheduling can be computed without modifying the conflict serialization. The negative side of solving the LCS without taking traffic into account is that a possibly short path in the conflict graph (i.e., one with few links) may end up carrying a large amount of traffic because of routing, and hence become critical. Nevertheless, since routing decisions are taken afterwards in the reduced DARS, flows would be routed around such critical paths as a consequence of routing decisions.

Lagrangian heuristic

The r-DARS is still a complex problem. While it can be solved in a matter of seconds in a 4×4 grid, it takes hours to solve it on a 5×5 grid. Therefore, we propose a heuristic scheme to solve it. The latter is based on a Lagrangian relaxation, which has a twofold advantage: (i) by exploiting the very structure of the r-DARS, it allows the problem to be partitioned, hence gaining in efficiency and/or scale; (ii) it allows one to compute both a lower and an upper bound on the optimum solution to the r-DARS. We first explain how to obtain a Lagrangian relaxation, and then show how the heuristic is built upon the latter. The r-DARS has two blocks of variables: the link scheduling variables, involved in constraints (i), (viii-x) and the routing variables in constraint (xi). In addition, a set of coupling constraints, i.e. (ii-vi), collate link scheduling and routing

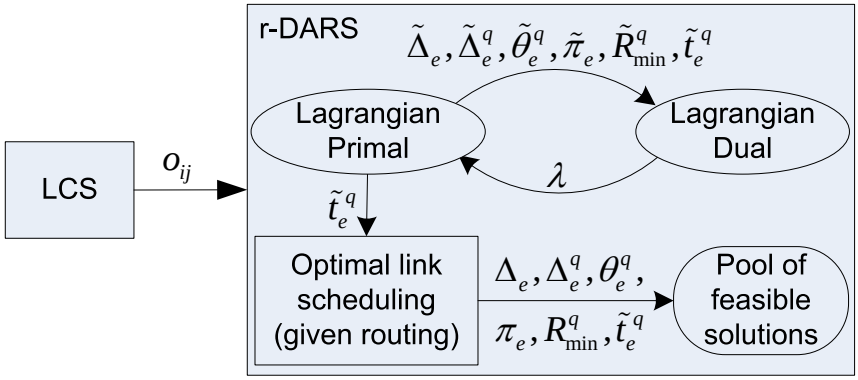

Figure 3.19. Separate heuristic approach

decisions. In the absence of the latter, r-DARS could be decomposed in two sub-problems: a link scheduling problem and a routing problem respectively. Hence we perform a Lagrangian relaxation with respect to the coupling constraints: the latter are dualized by inserting them in the objective function and associating a non-negative Lagrangian multiplier λ_i to each of them. The Lagrangian primal problem to be solved is the following:

$$\varphi(\lambda) = \min_{s.t.\ (i),(vii-x)} \{V_{\max} + s(\lambda; \Delta_e, \Delta_e^q, \theta_e^q, \pi_e, R_{\min}^q)\} + \min_{s.t.(xi)} \{r(\lambda; t_e^q)\}$$

where $s(\lambda; \Delta_e, \Delta_e^q, \theta_e^q, \pi_e, R_{\min}^q)$ and $r(\lambda; t_e^q)$ are linear cost functions depending on the Lagrangian multipliers (updated Lagrangian costs). The Lagrangian multiplier λ_i plays two roles: i) it penalizes the variables for which the relaxed i-th constraint is violated by adding a positive term to the original objective function, and ii) it favors solutions for which the relaxed i-th constraint is satisfied, by adding a negative term to the objective function. Function $\varphi()$ is separable: for a given value of λ, solving the Lagrangian primal implies solving separately a scheduling problem and a routing problem, which is considerably faster than solving them jointly. Yet this scheme keeps routing and scheduling together through the multipliers, hence retaining the benefits of a joint approach. The solution thus computed is a lower bound on the optimum of the r-DARS, and is in general infeasible. It is thus necessary to compute the best lower bound among the possible choices of λ, i.e., to solve the Lagrangian dual:

$$\max_{\lambda \geq 0} \{\varphi(\lambda)\}. \tag{3.24}$$

By iterating between the primal and dual problems (see Figure 3.19), the solution moves towards the admissible solution from the outside. However, this scheme may never converge to an admissible solution. For this reason, we choose to exploit the routing part of the solution of the Lagrangian primal (i.e., the t_e^q variables), and we solve the reduction of the r-DARS where routing variables are set (optimal link

scheduling in Figure 3.19). This last box entails solving a Mixed Integer Non-Linear problem, whose non-linear constraints are convex. If a feasible link scheduling is computed on a given routing, then the solution verifies all the constraints, and is thus admissible for the r-DARS problem (although not necessarily optimal), hence it is an upper bound on the optimum. As the Lagrangian scheme is iterated, possibly many feasible solutions are computed this way and stored in a pool. When the Lagrangian converges:

1. the best solution in the pool is returned;
2. the lower bound is given by the solution of the Lagrangian primal.

Note that, even though routing and link scheduling are decided in two separate modules in Figure 3.19 (i.e., the Lagrangian primal and the optimal link scheduling), the fact that the Lagrangian scheme iterates between the primal and dual, computing bounds on the activation variables, implies that routing decisions are affected by scheduling decisions and viceversa, which makes the approach joint in all respects. A solution approach like this belongs in the Lagrangian heuristics family ([4]). As far as solution efficiency is concerned, we solve the Lagrangian dual via a bundle type method ([27, 26]), which is more efficient than a standard subgradient method, as it takes into account information from previous iterations when searching for the ascent direction and step.

Performance evaluation

The contribution of this section is twofold. First, we evaluate the performance of our heuristic approach to solve the DARS problem, in terms of optimality and complexity. Second, we exploit it to infer structural properties of the WMN, i.e. optimal placement of one or more Internet gateway nodes. We present the above contributions in two separate subsections. A. Evaluation of the heuristic approach As for the first objective, we make simulations on a grid of varying diameter, up to 7×7 nodes. All links have a capacity equal to 9600, and the gateway is located in one corner. We assume that each link interferes only with those that are one hop away, and set the conflict graph accordingly. One flow is originated at each node, and is to be routed to the gateway. Instances are solved using an Intel Core 2 Duo CPU, 2.33GHz using IBM ILOG CPLEX 12.1 As for optimality, we compare the optimal DARS solutions, where available (up to a 4×4 grid) and those computed with the heuristic LCS+r-DARS. In this last approach, the r-DARS is solved both optimally and via the Lagrangian heuristic. For each test set, we evaluate the objective on a set of 30 randomly generated instances, with heterogeneous flow requirements: rates and bursts are generated uniformly between $[0, 9600/(2 \cdot |Q|)]$ and $[0, 1000]$, while the deadlines are set to either 60 or 90. Frames have 100 slots. We first show that separating the LCS and the r-DARS yields accurate results. Figure 3.20 shows the relative gap with respect to the DARS optimum in a 4×4 grid. The figure clearly shows that the suboptimal solutions of the two schemes are within few percentage points to the optimum. However, solving the r-DARS opti-

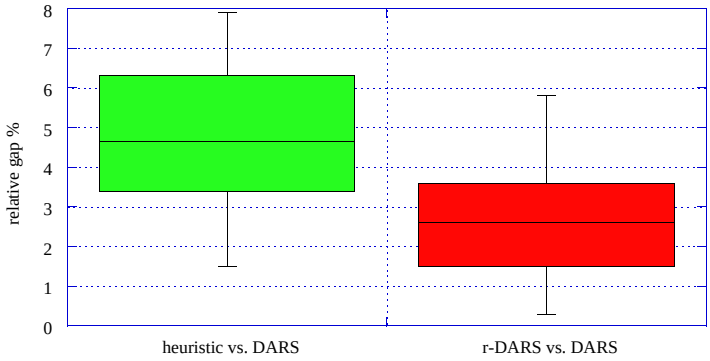

Figure 3.20. Accuracy comparison of the heuristic schemes

mally is time consuming: already with 5×5 grids, we could not find instances this took less than 8000s. Instead, the Lagrangian heuristic is considerably faster. Figure 3.21 reports a box plot of the solution times of 30 instances of grids, from 4×4 to 7×7. The figure shows that routing plans can be done in a few hours for grids up to 7×7, which is quite a large dimension for a WMN. Furthermore, it is worth remarking that the number of conflicts in a grid topology is predictably larger than in a real-life WMN deployment with the same number of nodes. Hence we can expect computation times to be considerably smaller in practical cases. Next, we show the benefits of having a

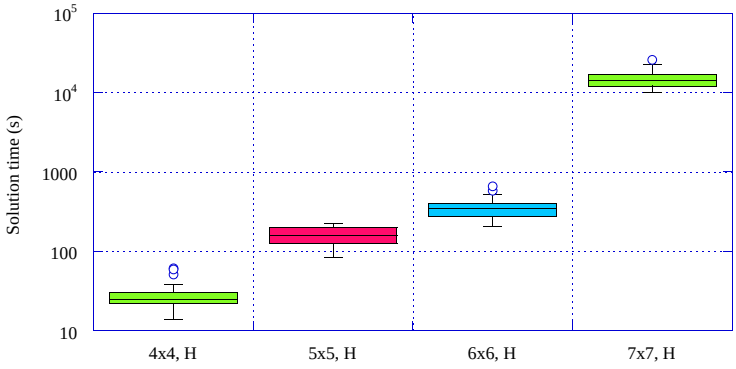

Figure 3.21. Solution time for the LCS+r-DARS, using the Lagrangian heuristic

joint routing and scheduling, by comparing it to a cascading approach, where routing decisions are taken first, oblivious of link scheduling. In the latter, we use a capacitated multicommodity flow (CMF) routing, where each flow q requires a capacity equal

to its rate ρ_q, and the routing that minimizes the overall number of traversed links is chosen, keeping into account the capacity constraints. The CMF sets the t_e^q variables, and then the link scheduling is solved optimally given the routing, as in [9]. In the joint approach, we use LCS+r-DARS, with the latter solved through the Lagrangian heuristic. Figure 3.22 shows the relative gap between the cascading and the joint approaches for two sets of instances of a 6×6 grid: for the first set rates and burst are again generated uniformly between $[0, 9600/(2 \cdot |Q|)]$ and $[0, 1000]$, for the second one the rates are generated between $[0, 9600/(1.2 \cdot |Q|)]$; this leads to instances where the WMN is highly congested, with the links close to the gateway approaching the saturation point. For the first set a joint approach (although solved suboptimally) always performs 10%-15% better in terms of objective function, despite the fact that both subproblems are solved optimally in the cascading approach. For the second set the gap grows to 20%. However, the cascading approach fails to compute a feasible solution in as many as 37% of the instances, whereas our joint approach solves them all. Then, we show how schedulability of a set of flows changes with their rate and

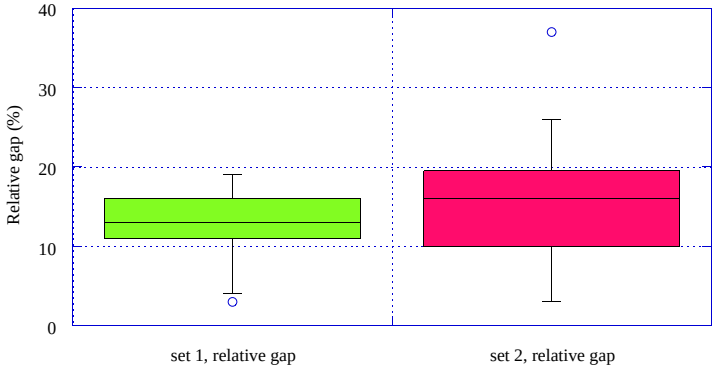

Figure 3.22. Relative gap between the cascading and the joint approach (the latter solved through the Lagrangian heuristic) on a 6x6 grid WMN

burst. Figures 3.23, 3.24, and 3.25 show the maximum violation as a function of the burst and rate of the flows. Figures 3.23 and 3.24 show results for a burst value of 1000 against a rate from 50 to 300 on a 5×5 and 6×6 grid respectively. Figure 3.25 reports results for a burst size ranging from 0 to 2000 and a rate of 150. In the above figures, the (unfeasible) solution of the continuous relaxation of the r-DARS problem is shown for comparison. The latter is a lower bound on the optimum, and its purpose its to show that - despite we cannot compute the optimum DARS solution - both the r-DARS optimum and its heuristic approximation are quite close to the DARS lower bound, hence to the DARS optimum itself.

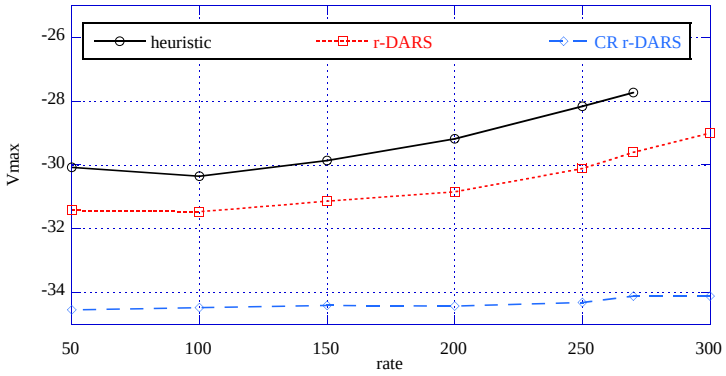

Figure 3.23. Maximum violation as a function of the rate for a 5x5 grid topology

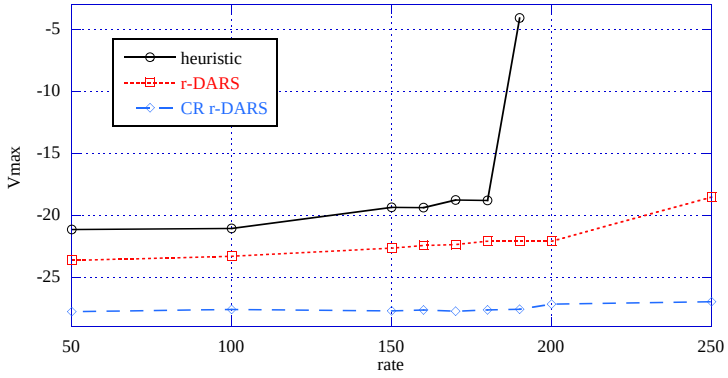

Figure 3.24. Maximum violation as a function of the rate for a 6x6 grid topology

Case study: optimal gateway placement

We now show how to exploit our solution scheme to infer properties which are useful from a network engineer perspective. More specifically, we discuss optimal gateway placement in both single-gateway and multi-gateway WMNs. We take as an example a 5×5 grid mesh, shown in Figure 3.26, and we initially place a single gateway and homogeneous traffic, one flow from each node to the gateway. For obvious reasons of symmetry, we only move the gateway toward one border and corner of the WMN. Figure 3.27 shows $V_{\max}$ as a function of the rate when a single gateway is placed at various nodes, from the center to the corner, for a burst equal to 1000 and a deadline of 60. The figure shows that $V_{\max}$ is minimized when the gateway lies in the center. The result makes sense since a central gateway minimizes the length of the longest path as well, which are the ones likely to contribute to $V_{\max}$. Figure 3.28 further clarifies that a larger $V_{\max}$ is obtained at the price of a higher resource expen-

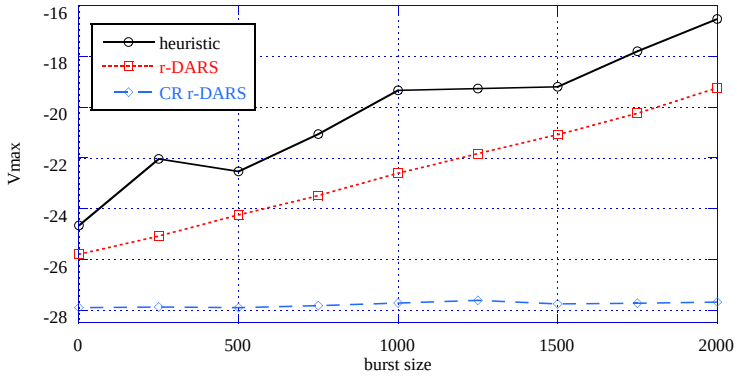

Figure 3.25. Maximum violation as a function of the burst size

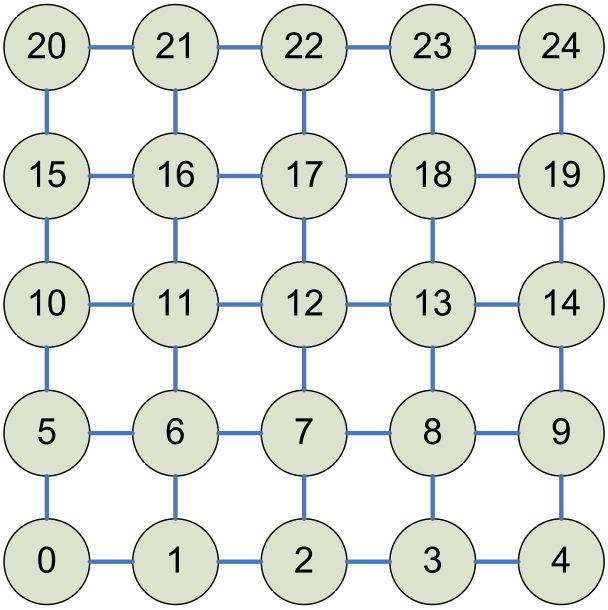

Figure 3.26. The test-case 5x5 WMN

diture, its vertical axis reporting the sum of the allocated capacity on all the slots of the schedule. Note that it is not possible to obtain a feasible schedule with $\rho = 350$ when the gateway is placed in the corner. We repeated the evaluation with random flows, whose parameters are the same as in the previous section. The results, shown in Figure 3.29, show that the distribution of $V_{\max}$ moves to the right as we move the gateway from the center to one corner. Finally, we compared the single-gateway scenario to one where the WMN has two gateway nodes. Figure 3.30 shows both $V_{\max}$ (left vertical axis) and the allocated capacity (right vertical axis) as a function of the

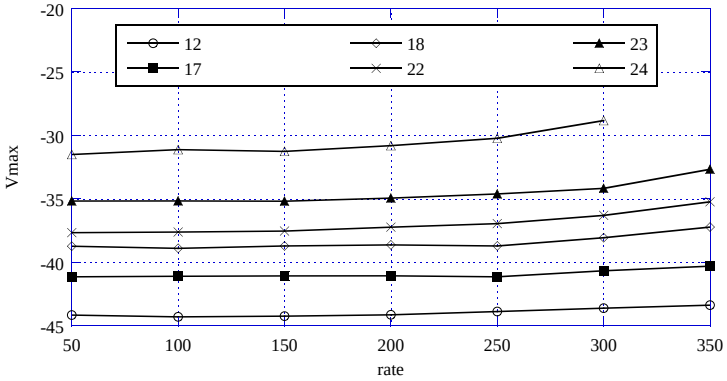

Figure 3.27. Vmax as a function of the rate for various gateway placements - homogeneous traffic

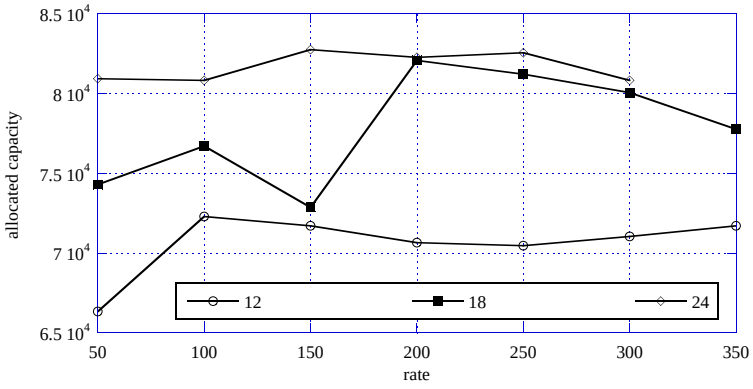

Figure 3.28. Allocated capacity as a function of the rate for various gateway placements - homogeneous traffic

placement of the gateways. The most favorable single-gateway scenario is reported on the left for comparison. All data are related to a homogeneous traffic scenario, with one flow from each non-gateway node whose characteristics are $\rho = 100$, $\sigma = 1000$ and $\delta = 90$. Note that the two-gateway scenarios have one flow less than the single-gateway scenario, as gateways send no traffic themselves. The figure shows that the more far apart the two gateways are, the worse $V_{\max}$ is, and the higher (in general) is the allocated capacity. However, it also shows that the only result that can be achieved by putting two gateways is to improve $V_{\max}$ marginally, at the price of a 27% increase in the allocated capacity. Within the limit of the considered scenarios, this suggests that a single gateway, placed at the center, is the optimal solution for a WMN of this topology and traffic.

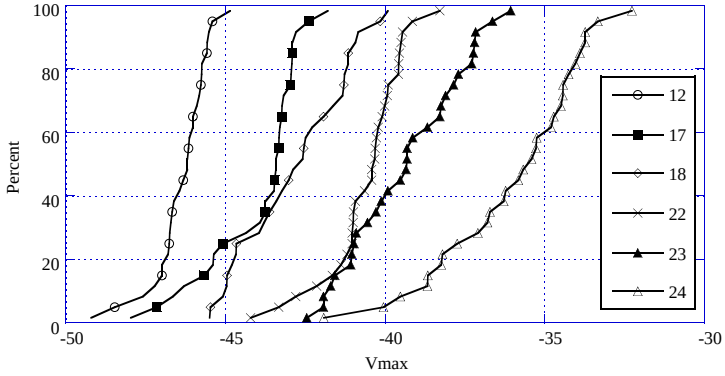

Figure 3.29. Distribution of Vmax over 30 random instances with different placements of the gateway node

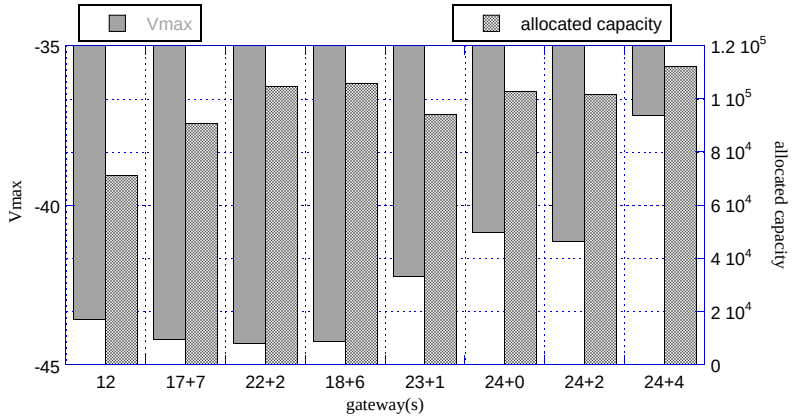

Figure 3.30. Vmax and allocated capacity for a single gateway and two-gateway scenario

3.7 Conclusions

This chapter has addressed the problem of routing and link scheduling and for real-time traffic in Wireless Mesh Networks. Real-time traffic requires a priori end-to-end delay bounds, and available existing formulations of the above problems do not take the latter into account. We showed that, given a conflict graph and flow routes, we can formulate the link scheduling problem as an integer non-linear optimization problem. The latter can be solved optimally for WMNs of small to medium size, depending on the aggregation framework at the nodes. Furthermore, we showed that the feasibility of a link schedule does depend on the aggregation framework, and we derived guidelines to choose the appropriate aggregation framework given a network scenario. We then proposed a heuristic solution approach that computes good suboptimal sched-

ules for WMN of larger sizes and/or in smaller times. The latter can be used as an online admission control scheme for real-time traffic.

Finally, we brought routing back into the framework, i.e. addressing the problem of jointly solving the routing and link scheduling problem optimally, taking into account end-to-end delay guarantees. We have formulated the problem as an optimization problem, which is however too complex to solve optimally already at relatively small scales (e.g., a 4x4 grid WMN). We have devised a heuristic, based on (i) extrapolating the link conflict serialization from the rest of the DARS problem, and (ii) solving the reduced DARS problem using a Lagrangian heuristic, which allows one to reap the benefits of a joint routing and scheduling approach, without paying the price of the added model complexity. Our results show that the heuristic scheme is fast and accurate, allowing a network administrator to provision a WMN of several tens of nodes so as to meet pre-specified delay guarantees for real-time traffic. Furthermore, we have used the above technique to identify guidelines for the optimal placing of gateways in the WMN.

This is the first work considering delay bounds as an objective, despite the abundant literature on joint routing and scheduling. Future work will include considering multipath routing, i.e. allowing a traffic flow to be split among several paths in order to balance the link utilization.

4

A time division carrier-sensing scheme for IEEE 802.11 Wireless LANs

With the proliferation of WLANs, it is common to find numerous WLANs within a small geographical area. For instance, in a typical office building, it is not hard to find tens of APs, each of them supporting numerous wireless devices, ranging from laptops to smart phones. As there are not enough orthogonal frequency channels in IEEE 802.11 standard (the most commonly deployed standard, 802.11g, only has three orthogonal channels), co-channel interference among different cells pervasively exists, and hence makes the channel capacity severely under-utilized. A previous work [15] showed this capacity degradation, and such problem becomes especially evident when the network is saturated in traffic from real-time applications, such as voice over IP (VoIP). Our NS2simulations show up to an 85% of throughput capacity degradation in different settings, ranging from a single isolated WLAN to a 25-WLAN network.

Some previous work has considered Time Division Multiple Access (TDMA) MAC to schedule the wireless links in WLANs [1, 40, 12]. However, in their proposals, carrier sensing multiple access (CSMA) is replaced entirely by TDMA. Although TDMA solves the co-channel interference problem, its implementation is very challenging in large-scale WLANs. In fact TDMA requires a tight synchronization among the stations, which creates a lot of overhead resulting in throughput degradation. In addition, the link scheduling problem is NP-complete , and the computational cost of the scheduling task grows exponentially as the number of clients increases.

In this chapter we propose a Time Division Carrier Sensing Multiple Access (TD-CSMA) scheme to increase the throughput capacity of large-scale multiple WLANs. In TD-CSMA, instead of assigning different time slots to individual wireless clients, a relatively large number of time slots is allocated to a group of clients. The clients scheduled in the same time slot then contend for channel access using the original 802.11 CSMA scheme. As we schedule time slots to groups of clients, not every single individual client, the number of scheduling entities is largely reduced. This has the practical advantage to reduce the computational cost of the scheduling task. We formulate a time slot scheduling problem in TD-CSMA so that the capacity of the closely co-located multiple WLANs can be maximized. Hence, the total utilization of

the wireless channel can be increased. To our best knowledge, this is the first work that develops a formal analytical framework of this type.

From extensive simulations, we find that TD-CSMA yields much higher throughput than the legacy CSMA. Moreover, TD-CSMA incurs smaller delays in packet transmissions than the legacy CSMA, which has also a packet dropping rate up to three times bigger. The main contributions of our work are:

1. Development of TD-CSMA, a new delay-aware client scheduling scheme, which is incorporated within the legacy CSMA approach to schedule large-scale multiple WLANs;
2. Analysis of the performances of TD-CSMA in WLANs for different topologies and traffic requirements in terms of rates and delays.

TD-CSMA greatly enhances the efficiency of the popular IEEE 802.11 protocols. As a result, the total capacity of co-located multiple WLANs is remarkably increased. This work provides a practical solution for supporting bandwidth-demanding applications, such as the high definition video streaming, over large-scale multiple WLANs. The analytical framework we develop for TD-CSMA helps to evaluate the efficiency of incorporating the deterministic scheduling (TDMA) into random access (CSMA) networks.

4.1 Related work

Previous work on this topic mainly focus on either using pure TDMA or CSMA schemes in WLANs. Due to the large overhead of synchronization in TDMA networks and the inefficiency of CSMA on crowded networks, there is a trend to incorporate TDMA into CSMA networks. The works in [22] and [63] focus on realizing the concept of TDMA by using the CSMA mechanism. They do not really integrate the principles of the CSMA and TDMA. So, the common inefficiencies of TDMA are still there. The work in [16] uses a pure TDMA, but when a node has nothing to send in its designated slot other nodes can contend to access the unused channel through CSMA. They address the problem of the inefficient channel capacity usage in TDMA when the traffic loads of certain nodes are low. However, their protocol simply activate TDMA and CSMA in different time periods so that they can be complementary to each other. When TDMA is working, it is basically a pure TDMA system, so all the disadvantages of pure TDMA arise again. All of the work mentioned above assume all the nodes are in the same contention region. Such scenario can no longer be applied to the crowded WLANs deployment in which not all the nodes can hear one another. Moreover, when the network grows larger and there are multiple WLANs interfering each other, the scheduling problem becomes too complicated.

In [15] authors suggest assigning time slots not to individual wireless clients, but to a group of clients in the context of voice over IP (VoIP) application. This type of time-slot assignment is called “Coarse-grained TDMA” (CoTDMA). From a preliminary

analysis, CoTDMA can help increase the channel utilization and hence increases the number of concurrent VoIP sessions. A heuristic based on a graph coloring algorithm is used to assign the time slots to different VoIP sessions. However, a formal analytical framework for CoTDMA is missing, and the general performance (not merely for VoIP) of this coarse-grained TDMA largely remains unclear. More analyses are needed for finding how the time slots can be assigned optimally, and how the delay of the real-time traffic flow is bounded.

An optimal link scheduling formulation for real-time traffic in wireless mesh networks is studied in [14] in the context of the pure TDMA. The proposed framework can find an optimal link scheduling with end-to-end delay performance guarantees. Borrowing some concepts from [14], we generalize the idea of CoTDMA in [15], and develop a novel scheme, Time Division Carrier Sensing Multiple Access (TD-CSMA), for scheduling traffic in large-scale WLANs. CoTDMA in [15] denotes a particular time division scheme for VoIP traffics.

4.2 Time division carrier sensing multiple access

Carrier sensing multiple access (CSMA) is a random-based medium access protocol[1] [28]. In IEEE 802.11 WLANs, if a client has a packet to send, it first randomly chooses a backoff window size. The client then counts down the backoff window, and senses the medium at the same time. If it senses another transmission, it will freeze the counter. Otherwise, when it counts down to zero, it will transmit its packet. A previous work [8] shows that when the number of contending clients is large (greater than five in an 802.11b WLAN) CSMA loses its efficiency, i.e. the total throughput steeply decreases, and the wireless channel capacity is severely under-utilized. The reason is that when the number of contending clients increases, the mechanism of randomly choosing a backoff window size to resolve the contention is no longer effective. Furthermore, when there are multiple 802.11 WLANs co-located in the same geographical area, the situation becomes worse due to the co-channel interference from different WLAN cells, and the wireless channel capacity is eroded further [15].

4.2.1 Basic idea of TD-CSMA

We first explain the concept of Time Division Carrier Sensing Multiple Access (TD-CSMA) using the simplified scenario depicted in Figure 4.1. The figure shows two adjacent WLAN cells. Clients C1 to C5 are associated with AP1, while clients C6 to C8 are associated with AP2. The clients communicate with the respective associated AP only, i.e. assuming no multi-hop communications. We partition the wireless clients into groups called clusters. We require each cluster to be active without interfering with the other clusters of the network.

[1] We assume Distributed Coordination Function (DCF) is used, as it is the most commonly deployed CSMA scheme in nowadays WLANs. CSMA standard also has a Poll Coordination Function (PCF). However, PCF is not well tested in practice and is seldom used.

Figure 4.1. Different time slots (TS) are assigned to disjoint groups of clients in two nearby WLANs

In Figure 4.1, the clients are divided into five disjoint clusters (we will explain how to form clusters in Section 4.3.1). Each cluster is also assigned with a time slot ID (TS1 to TS4). In that particular time period, the clients in the cluster can be active (sending or receiving packets). In our example, we assume clients C7 and C8 are far away from C1, C2 and C3, so the cluster of C7 and C8 can be assigned either TS2 or TS3 without interfering with C1, C2 or C3. Note that within each cluster, there may still be multiple clients, and the original 802.11 CSMA scheme is used to coordinate transmissions among these clients. The advantages of this approach are three-fold:

1. the carrier-sensing range of the nodes can be reduced to only cover the nodes in the same cluster. This causes less interference to other clients;
2. the co-channel interference from neighboring WLANs can be isolated by time slot assignment, e.g. clients C5 and C6 will be active not at the same time, avoiding degrading each other's performance;
3. only a relatively small number of clients in the cluster accesses the wireless medium using CSMA. This enhances the efficiency of CSMA [8].

Compared to a pure TDMA scheme, the adoption of TD-CSMA in heavily crowded WLANs has also some practical advantages. When the clients are grouped into clusters, the resulting interfering entities are only a subset of those present if the pure TDMA is used, i.e. when each single client is competing for the slot resource. Thus, the scheduling problem in the pure TDMA context is indeed more difficult to solve as the size of the networks increases, because of the complexity that arises while dealing with interference-free conditions. Therefore, the TD-CSMA protocol overhead is only a fraction of the one caused by a pure TDMA approach.

4.2.2 Isolating co-channel interference

According to the widely used protocol interference models [34], we first define the interference range (IR) of a single node:

$$IR_k = (1 + \delta)d_k \tag{4.1}$$

where IR_k is the Interference Range of a node k, (either a client or an AP), d_k is the length of the link associated with the node; δ is a distance margin for interference-free reception with typical value of 0.78 [60, 37]. Within IR_k, any other transmission will interfere with the node k's reception of the signal. The interference condition for clusters can be derived from a conflict graph $G_c = (U, E)$,whose vertices are the links of the connectivity graph and whose edges $E = \{e_1, \ldots, e_r\}$ model the conflicts within the network. Figure 4.2 shows how the conflicting set of clusters can be derived. First, we have a connectivity graph that shows how clients associate with their APs

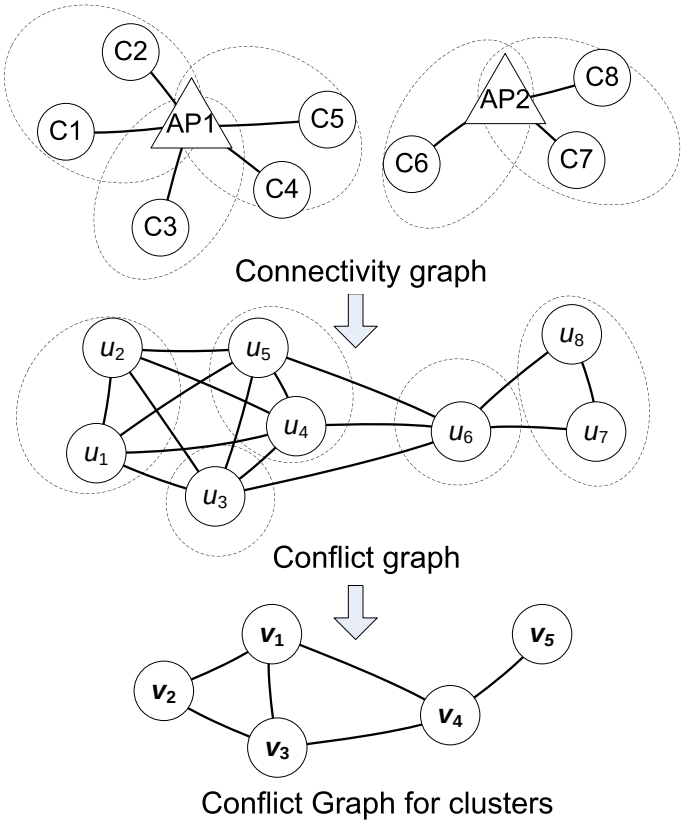

Figure 4.2. Derivation of the cluster conflict graphs

(same as Figure 4.1); dotted ellipses represent cluster borders. A conflict graph is then created, in which a vertex u_i represents the link of the connectivity graph between client c_i and the respective AP, while edges model conflicts between links. Given a pair of nodes u_i and u_j, an edge is added to the conflict graph if one of the following inequalities holds:

$$\begin{aligned} IR_{u_i} &> \min(d_{u_i,u_j}, d_{u_i,\bar{u}_j}) \\ IR_{\bar{u}_i} &> \min(d_{\bar{u}_i,u_j}, d_{\bar{u}_i,\bar{u}_j}) \\ IR_{u_j} &> \min(d_{u_i,u_j}, d_{u_i{}',u_j}) \\ IR_{\bar{u}_j} &> \min(d_{u_i,\bar{u}_j}, d_{\bar{u}_i,\bar{u}_j}) \end{aligned} \tag{4.2}$$

where d_{u_i,u_j} is the distance between the nodes u_i and u_j. $\bar{u}_i$and $\bar{u}_j$ are the APs u_i and u_j are associated to, respectively. Finally, the conflict graph for clusters is built: here vertices $v_i \in V$ represent clusters, and there is an edge between any two clusters v_i and v_j if any of the clients belonging to v_i is within the interfering

range of any of the clients in v_j. To avoid interference, if a cluster v_i is active in slot t, all the clusters that are in its neighborhood in the conflict graph must refrain from transmission. For the sake of conciseness, indexes are now dropped when referring to clusters.

We define an activation offset π_v for cluster v, and a transmission duration Δ_v the clients of the cluster v are allowed to transmit for. Without loss of generality, the frame starts right after the beacon frame. Since time is slotted, π_v and Δ_v are non negative integers. The schedule must ensure the conflict-free condition: while a cluster is transmitting, all conflicting clusters must refrain from transmission. For any pair of clusters u and v which are neighboring vertices in V we have that if v transmits after u, it must wait for u to complete the transmission, i.e.

$$\pi_u - \pi_v + \Delta_u \leqslant 0.$$

Otherwise, the symmetric inequality holds, i.e.

$$\pi_v - \pi_u + \Delta_v \leqslant 0.$$

In order to linearize the combination of the above constraints, we introduce a binary variable o_{uv}, $(u, v) \in V$, such that which is 1 if u transmits after v, 0 otherwise.

$$o_{uv} = \begin{cases} 1 & \text{if } u \text{ transmits after } v, \\ 0 & \text{otherwise} \end{cases}$$

The left-hand side of the previous constraints can be upper bounded by the TD-CSMA frame size N regardless of the relative transmission order, as π_u and Δ_u belong to $[0, N]$. This completes the formulation of the conflict-free constraints, which are necessary and sufficient conditions:

$$\pi_u - \pi_v + \Delta_u \leqslant N \cdot o_{uv} \ \forall (u, v) \in V \tag{4.3}$$

$$\pi_v - \pi_u + \Delta_v \leqslant N \cdot (1 - o_{uv}) \ \forall (u, v) \in V \tag{4.4}$$

Finally, a valid schedule must have all the clusters accomplishing their transmissions within the frame duration, i.e.:

$$\pi_v + \Delta_v \leqslant N \quad \forall v \in V \tag{4.5}$$

During the Δ_v slots assigned by the TD-CSMA scheduler to cluster v, each client of the cluster v will access the wireless channel through CSMA, transmitting its packets if it wins the contention. Please note that there is no need to add explicit half-duplex constraints on the APs, since the access to them from different clusters within the same WLAN cell is regulated through the time slot scheduling, while communications within a specific cluster take advantage of the CSMA mechanism. We focus the analysis on the upstream case only. It is worth noting that this is not a limitation, since the current formulation can be easily extended to consider the downstream flows.

4.2.3 Frame structure in TD-CSMA

Now we look at the structure of the time frames in TD-CSMA. We assume each client to transmit on the same time-slotted channel. As shown in Figure 4.3, each TD-CSMA frame has a length of N slots of T_S seconds each one, hence periodically repeated every $N \cdot T_S$ seconds. Beacon frames are used for synchronization. Each slot is assigned to a non-interfering subset of clusters.

Similar to [15], we also introduce the concept of guard time. No packet transmission should be initiated within the current activation period if the beginning of the activation period for the next scheduled cluster is only a guard time away. The guard time should be set at least to the minimum time needed to complete a packet transmission. This is to ensure that packet transmission will not straddle across two activation periods. Let G be the guard time slot, and it can be calculated by the following equation:

$$G = \left\lceil \frac{\max(P)}{R_{802.11} T_s} \right\rceil \tag{4.6}$$

where max(P) is the maximum packet size and $R_{802.11}$ is the bit rate for a particular 802.11 protocol (it is 11Mbps for 11b, while it is 54Mbps for 11a/g). For ease of notation, all the time quantities we derived in the rest of chapter are in terms of slots. That means we drop the term T_S.

The delay analysis proposed in this work is based on the following:

Assumption. *If a packet cannot be successfully transmitted after k activation period, it will be dropped.*

Let k be the packet survival time, i.e. k is the number of TD-CSMA frames a packet is kept alive if it cannot be transmitted due to CSMA collisions. k is an integer greater than one. There are two reasons that justify this assumption:

1. If the packet is generated within the cluster activation period but cannot be transmitted before the assigned duration time Δ_v elapses, it may not be able to meet the delay requirement if it still cannot be sent out after k activation periods;
2. Older packets should be discarded to let newly generated packets meet their delay requirements.

A packet originated from a client belonging to cluster v, before being dropped, experiences its maximum delay if it was generated at the beginning of a frame, and was sent out right before the beginning of the guard time of the next k^{th} activation period. Therefore, the worst case delay a packet from cluster v can experience is

$$D_v^{\max} = \pi_v + (k-1) \cdot N + \Delta_v - G.$$

If δ_v is the delay requirement for the packets in cluster v, it must hold that

$$D_v^{\max} \leqslant \delta_v, \tag{4.7}$$

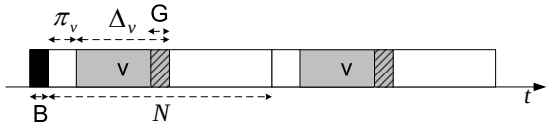

Figure 4.3. The structure of the time frames in TD-CSMA

where

$$\delta_v = \min\{\delta_i : i \in v\}$$

is the minimum delay requirement among the WLAN clients i in the cluster v. The beacon frame adds an additional small constant delay which is not considered in the model, but it can be easily plugged in.

In a real implementation, the simplest way for realizing TD-CSMA is to use a central controller connected to the APs through a backbone network. Exchanging control packets between wireless clients and their APs, the conflict relationships among the clients can be found. One of such algorithms for finding conflict relationships is the power exchange (PE) algorithm proposed in [38]. Once conflicts are defined, the central controller can draw a conflict graphs, as the one in Figure 4.2. To emulate the time slots, the "sleep mode" of the 802.11 protocol [28] can be used. A detailed discussion of it can be found in [15]. Another option for emulating the time slots is to use the Soft-TDMAC proposed in [22]. Instead of doing network synchronization for individual clients, in TD-CSMA, synchronization is done for groups of clients. As the total entities needed for synchronization is largely reduced after clustering, the overhead for synchronization by Soft-TDMAC is also reduced.

4.3 TD-CSMA resource optimization

Since the way clients are grouped together directly affects the scheduling task, a joint formulation considering both the construction of the clusters and their scheduling should be addressed. However, a formulation of this type involves all the possible partitions of the WLAN clients in clusters, and all the potential conflicts that could arise from the use of those clusters. The resulting formulation can not be handled due to its complexity; the model is not reported here due to lack of space. Instead, we decompose the problem into two disjoint sub-problems:

1. the constrained cluster assignment, where the number of clusters formed meeting the specific requirements is minimized;
2. the cluster scheduling, where the clusters are finally scheduled for transmission.

Since the client mobility modifies the conflict graph over time, the cluster assignment and scheduling problems must be solved periodically so as to consider both new clients that are joining the network and old clients that changed their position.

4.3.1 Constrained cluster assignment

TD-CSMA partitions clients into clusters meeting two basic constraints:

1. the clients grouped into the same cluster should be within the carrier sensing range of each other;
2. the number of clients in the same cluster should be relatively small.

The constraint 4.1 ensures that the clients in the same cluster can carrier sense each other for medium access coordination. That follows the basic idea of 802.11 MAC DCF [28]. The constraint 4.1 maintains the efficiency of CSMA, as explained in Section 4.2.

In the following discussion, v is still referred to a cluster while i and j are referred to the WLAN clients within the cluster. For each cell of the WLAN, i.e. for all the clients of the network associated with the same AP, an independent clustering problem is formulated. Let $C = \{v_1, \ldots, v_{|C|}\}$, where $|C| = 2^{|\Gamma|}$, be the set of all the possible clusters built from the $|\Gamma|$ clients of a WLAN cell, where Γ is a set of all the clients within the cell. We introduce some binary variables, x_{iv}, which is 1 if client i is assigned to cluster v, and 0 otherwise. y_v, which is 1 if cluster v is used, and 0 otherwise, and $i \in \Gamma$. We formulate the constrained clustering problem (CCP) as:

$$
\begin{array}{lll}
\min & \sum\limits_{v \in C} y_v & \\
\text{s.t. :} & x_{iv} \leqslant y_v & \forall i \in \Gamma, \forall v \in C \\
& \sum\limits_{v \in C} x_{iv} = 1 & \forall i \in \Gamma \\
& d_{ij} \leqslant CS_{range} + L \cdot (2 - x_{iv} - x_{jv}) & \forall i, j \in \Gamma : i \neq j, \forall v \in C \\
& \sum\limits_{i \in N} x_{iv} \leqslant \Omega & \forall v \in C
\end{array}
$$

where the objective function is the number of clusters used, to be minimized. While the first constraint is needed to tie together the two sets of variables, the second constraint ensure that every client i is assigned to one and one cluster only; the third constraint ensures that if two clients are assigned the same cluster, they must be within the carrier sensing range, CS_{range}, of each other. L is a constant large enough so as to activate the constraint only if both node i and j are in the same cluster. Finally, the fourth constraint prevents to assign more than Ω clients to the same cluster. This formulation is computationally expensive even for small instances, since the number of variables and constraints steeply grows as the size of the network increases. We then reformulate the above problem using a set partitioning formulation, whose Linear Programming (LP) relaxation can be efficiently solved using column generation. This technique is based on the observation that in problems with a large number of columns (variables), only a few of them are used in the optimal solution. First introduced in [29, 30], column generation decomposes the problem into a master problem, that contains a subset of the columns, and a sub-problem, which is used to identify whether any column should be added to the master problem or the current solution is optimal. Column generation alternates between the master and sub-problem, until the master solution optimality is proved. Let $\bar{C} \subseteq C$ be the set of all the clusters in

C satisfying the two basic constraints of the clustering (3^{rd} and 4th constraints of the CCP problem as well); using the same $y_v$variables defined above, we consider the following formulation:

$$\begin{array}{ll} \min & \sum_{v \in C} y_v \\ \text{s.t.} & \sum_{v \in \bar{C}} a_{iv} y_v = 1 \ \forall i \in \Gamma \end{array} \tag{4.8}$$

where a_{iv} is a constant equal to 1 if cluster v contains client i, 0 otherwise. The continuous relaxation of the above problem, with $0 \leqslant y_v \leqslant 1$ and the set $\bar{C}$ replaced by $\bar{C}_{CG}$, constitutes the master problem of the column generation approach; the set $\bar{C}_{CG}$ initially contains singleton clusters only, i.e. every client is assigned an independent cluster, so as to ensure the feasibility of the master problem. When the master problem is solved, using LP duality we need to check if any of the clusters in $\bar{C}$ not considered in the current master problem has a strictly negative reduced cost

$$\hat{c}_v = 1 - \sum\nolimits_{i \in N} \hat{\beta}_i a_{iv},$$

where $\hat{\beta}_i$ are the optimal dual variables associated to the set partitioning constraints of the master formulation (4.8). This in LP terms means that we should include the cluster with the minimum reduced cost into the set $\bar{C}_{CG}$, since it could lead to an improvement of the objective value. The sub-problem, usually referred as pricing, aims at finding the cluster in $\bar{C}$ that minimizes the reduced cost corresponding to the current optimal solution of the master. It's formulated by means of a binary variable z_i for each client of the cell, such that

$$z_i = \begin{cases} 1 & \text{if client } i \text{ is included in the cluster,} \\ 0 & \text{otherwise} \end{cases}$$

leading to the following formulation:

$$\begin{array}{lll} \min & 1 - \sum_{i \in N} \hat{\beta}_i z_i & \\ \text{s.t. :} & \sum_{i \in N} z_i \leqslant \Omega & \\ & d_{ij} \leqslant CS_{range} + L \cdot (2 - z_i - z_j) & \begin{array}{l} \forall i, j \in \Gamma : \\ i \neq j, \forall v \in C \end{array} \end{array} \tag{4.9}$$

The first and second constraints of the sub-problem correspond to the third and fourth constraints of the CCP model respectively.

If the optimal solution of the sub-problem has a strictly negative objective, the new cluster is added to the master problem, which is then re-optimized, progressing with the next iteration of the column generation. Otherwise, the LP relaxation of the CCP problem has been solved optimally, since none of the clusters in $\bar{C}$, if added to the current master, can lead to an improvement of its objective value. The column generation solves the LP relaxation of (4.8), while the original problem has binary variables; in case some of the variables in the optimal solution of the master are

fractional-valued, a branch-and-bound scheme has to be applied [72]. The interested reader can find a set of standard branching rules for column generation algorithms in [5].

While the above approach can solve optimally a large-scale partitioning problem, it does not guarantee the computational efficiency. We then propose a greedy algorithm to build sub-optimal clusters for large-scale networks in the order of milliseconds: it maintains a list of clients that are still not assigned to any cluster, and greedily pack them together in increasing order of their distance as long as the two basic constraints are met. i.e. clients are within the carrier sensing range of each other, and the overall number of clients is not exceeding Ω. Even using this simple heuristic, the TD-CSMA approach already shows a remarkable improvement in the channel utilization of 802.11 networks over the legacy CSMA as shown in Section 4.4.

4.3.2 TD-CSMA cluster scheduling

The goal of the TD-CSMA cluster scheduling is to minimize the dropping probability meeting rate and delay requirements. The dropping probability for a packet generated from a client within the TD-CSMA wireless network depends on two factors. First, according to [28], we assume that each packet in 802.11 DCF collides with constant and independent probability p_c regardless of the number of retransmissions has already suffered. Therefore, the probability of dropping a packet, p_D, is as$p_D = {p_c}^m$, where m is the maximum number of retransmissions. After m retries, the packet will be dropped. The larger the number of retransmissions, the smaller the probability of dropping the packet. Second, a packet could also be dropped if after k activation periods it has not yet been transmitted, even if it has not reached the maximum number of retransmissions. As we mentioned in Section4.2.3, k is a parameter of the TD-CSMA and it is called packet survival time. The bandwidth assigned to each cluster is proportional to the ratio Δ_v/N, where Δ_v are the slots assigned to cluster v and N is the length of the frame. It plays a major role in the dropping of packets whose survival time k has not expired. More the bandwidth assigned to a cluster, the higher is the probability that a packet is transmitted .

Our goal is to minimize the packet dropping probability, p_D^v, for all the clusters v of the network. Value of parameters m and k are given as input to the TD-CSMA scheduler and they are defined by the 802.11 standard and application respectively. Therefore, our objective is to maximize the bandwidth assigned to each cluster. Let s_i be the number of slots that should at least be assigned to client i in order to fulfill its rate requirement assuming no overhead is considered from the client's point of view; then $s_i = N \cdot \rho_i / c$, where ρ_i is the rate requirement of client i, and c the data rate for a particular 802.11 protocol (e.g., in 802.11b, c=11Mbps, and in 11g, c=54Mbps). The rate requirement can be written as $\rho_i = g_i \mathsf{E}[P]$, where g_i is the packet generation rate of client i, and E[P] is the expected packet size. It must then hold that

$$\Delta_v \geqslant \sum_{i \in v} (s_i + s_i T_s g_i (OH_{trans})), \tag{4.10}$$

where OH_{trans} is the overhead for a complete packet transmission. In 802.11, a complete packet transmission includes the backoff time, DIFS, SIFS, the transmission time of the ACK packet from the receiver, and the transmission time for the physical preambles and headers for both DATA and ACK packets. Therefore, the TD-CSMA *cluster scheduling problem* (CSP) can be formulated as the following integer optimization problem:

$$
\begin{array}{lll}
\max & \sum_{v \in V} \Delta_v / N & \\
\text{s.t. :} & \pi_u + (k-1) \cdot N + \Delta_v - G \leqslant \delta_v & \forall v \in V \\
& \Delta_v \geqslant \sum_{i \in Q(v)} (s_i + s_i \cdot T_S \cdot g_i \cdot (OH_{trans})) & \forall v \in V \\
& \Delta_v \geqslant \alpha \cdot \frac{|Q(v)|}{|Q(v)| + \sum_{u \in N(v)} |Q(u)|} \cdot N & \forall v \in V \\
& \pi_u - \pi_v + \Delta_u \leqslant N \cdot o_{uv} & \forall (u,v) \in V \\
& \pi_v - \pi_u + \Delta_v \leqslant N \cdot (1 - o_{uv}) & \forall (u,v) \in V \\
& \pi_v + \Delta_v \leqslant N & \forall v \in V
\end{array}
$$

where the first constraint ensures that the scheduling will meet the deadline requirement of every cluster of the network (refer to Equation (4.7)). The second constraint assigns a lower bound of the cluster transmission durations based on the client rate requirements (refer to Equation (4.10)). The third constraint is needed to ensure that a cluster with a large number of clients but a small rate requirement gets at least an amount of slots proportional to a local fairness condition, α, which is a constant to control the fairness influence on the scheduling solution. $Q(v)$ is the number of clients in cluster v, and $N(v)$ the set of neighbors of cluster v in the cluster conflict graph. Finally, constraints fourth to sixth are needed for the schedule to be valid and conflict-free, as explained in the Section 4.2.2. The CSP problem is an integer problem which can be solved optimally with a general purpose MIP solver (e.g. [17, 10]). Unfortunately, the presence of disjunctive constraints, and therefore, binary variables o_{uv}, makes the problem very hard to be solved even for small networks. Again, to solve large-scale instances efficiently, an heuristic approach is needed.

Such heuristic is inspired by the well-known greedy algorithm for the knapsack problem proposed by Dantzig in [20] and it can be decomposed in two main tasks:

1. given the minimum number of slots to be assigned to each cluster (2^{nd} and 3^{rd} constraint), it tries to pack cluster transmission durations Δ_v into the frame first;
2. then to expand the durations to fill the whole frame.

1) Packing phase

The packing subroutine tries to pack larger clusters first in non-conflicting *scheduling sets*. When packing a cluster, if it finds an already packed scheduling set containing clusters not in conflict with the cluster at hand, this is added to the set; otherwise it will keep on searching until it reaches the last scheduling set packed, then building a new set right after it, subjecting to Equation (4.5). Once the scheduling sets are

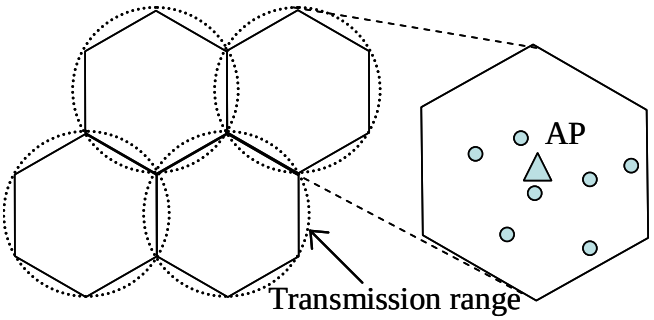

Figure 4.4. A 2-by-2 4-WLAN network

built, the procedure checks the deadline requirements given from Equation (4.7). If any deadline is violated, the algorithm then reorders the scheduling sets in increasing order of the average set deadline, which for a specific set h is defined as

$$\hat{\delta}_h = \sum_{v \in C_h} \frac{\delta_v}{|C_h|},$$

where C_h is the set containing the indices of the cluster within set h. If all the deadline requirements are met, the algorithm proceeds to the next phase, otherwise the scheduler fails in finding a feasible solution for the instance at hand.

2) Expansion phase

After the packing subroutine, some slots in the frame may still be unused. Since the objective is to maximize the cluster transmission durations, a scheduling set expansion subroutine is needed to improve the objective value of the current solution. If none of the delay constraints indicated by Equation (4.7) is tighter than validity constraints from Equation (4.5), each cluster v is assigned a number of slots proportional to the ratio

$$\frac{F \cdot |Q(v)|}{\sum_{u \in V} |Q(u)|},$$

where F is the number of unassigned slots in the frame. Otherwise, a delay-aware expansion tries to assign each scheduling set the maximum number of slots supported by the delay constraints of the stacked clusters. Finally, after the transmission duration of a set has been expanded, the procedure must also ensure that the following sets of transmissions scheduled are shifted to the right slots accordingly, without violating their deadline constraints. If there are still some unused slots in the frame, the heuristic reduces the frame size length to wrap the last set. This increases the bandwidth assigned to the clusters.

4.4 Performance evaluation

To evaluate the performance of TD-CSMA, we set up a D-by-D multi-WLAN network for NS2 simulations. Figure 4.4 shows an example of 2-by-2 network, where four hexagonal cells are placed side by side. The AP is at the center, and the dotted-line circle represents its coverage. Wireless clients are randomly placed in each cell. The clients generate UDP packets to send to the APs they associate, and we measure the network performance in terms of goodput, end-to-end delay and packet dropping rate. By varying the value of D, the number of clients in each cell and the packet generation rate of each client, we can evaluate TD-CSMA with different topologies. Although we simulate uplink flows only, the simulation can be easily extended to the downlink. For simplicity, we implemented the concept of time-slot division by setting the Network Allocation Vector (NAV) of wireless clients according to the scheduling results. NAV is originally used to resolve contention among the clients. A client will refrain from transmission for a period of time indicated in NAV[28]. In the simulations, we exploit the usage of NAV. As we have the full control of the simulator, like a central controller, we set different NAV values for different groups of clients to make the clients silent when a time slot is not assigned to them.

Solutions of TD-CSMA are solved optimally using the SCIP solver [67] for the CCP problem, and the IBM ILOG CPLEX [17] solver for the scheduling problem. Heuristic solutions are obtained combining the two heuristic algorithms proposed in Section4.3. Figure 4.5 shows the distribution of the computation times for solving three instances of a 2-by-2 topology with different number of clients in each cell. Computations are done on a PC equipped with an Intel Core2 Duo E6400 2.1 GHz, 2 GB RAM, and a Linux 2.6.32. The optimal solutions for both the CCP and the scheduling problem take up to several minutes even for such a small topology. However, the heuristic approaches run very fast and it is in the order of tens of milliseconds. Due to its efficiency, our network simulations mostly use the heuristic solutions for TD-CSMA, especially for large topologies.

For each topology instance, we form the clusters and input the schedule provided by TD-CSMA. We then run NS2 simulations for both TD-CSMA and the legacy CSMA. All the transmission parameters are set according to IEEE 802.11b standard default values. Although the simulation settings are based on 802.11b protocols, the results apply to all other 802.11 standards. As most of the 802.11 standards use CSMA DCF, the problems of CSMA pointed out in this thesis are still there. Other parameters are shown in Table I. It is worth noting that the setting of parameter values in TD-CSMA in a real implementation would depend on the network topology and applications. For example, in the simulation, we assume a real-time application which has a stringent delay requirement, so a small value of k is used. All the results shown below are averaged on ten runs of simulations; 95% confidence intervals are also reported in the graphs.

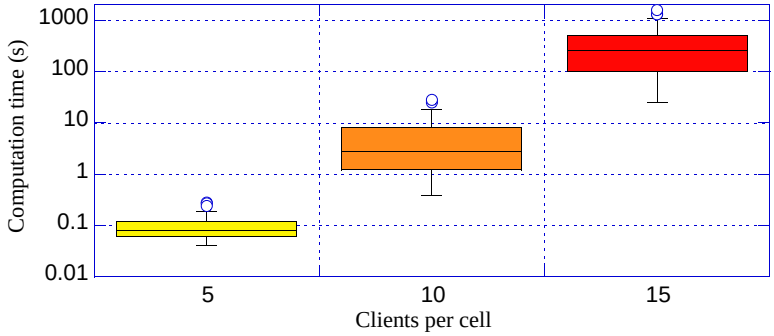

Figure 4.5. Solving time for a 2-by-2 topology

Radius of Cell	250m
Transmission Range (dmax)	250m
Slot Size	20 μs
Frame Size	500 slots
Guard Time	4 slots
Packet Survival Time (k)	2

Table 4.1. Parameters used for the simulations

4.4.1 Quality of services provided by TD-CSMA

We first evaluate how good the TD-CSMA helps to meet the delay requirement. We set up a 5-by-5 network and increase the number of clients per cell from five to 20. Each client generates 25 40-byte packet every second to send to its AP. We find from Figure 4.6 that the traffic in CSMA starts suffering large delay when the number of clients getting large. However, the traffic in TD-CSMA remains small and meets our 100ms delay requirement.

We are also interested in seeing how the network performance changes when different packet generation rates are used. With the same 5-by-5 topology, we increase the packet generation rate from 25 to 200 packets per second. From Figure 4.7, the goodput in the CSMA network quickly reaches the limit. Also the CSMA network with 20 clients each cell (Legacy CSMA20) has a decreased goodput when the packet generation rate increases. This is because the network is too saturated and many collisions occur. TD-CSMA20 network has a higher goodput than TD-CSMA10 as it has more clients each cell. It also reaches the capacity limit faster than TD-CSMA10. Even though TD-CSMA network also reaches the capacity limit as the packet generation rate goes very high, compared to the legacy CSMA network, it increases the throughput capacity of a 5-by-5 WLAN by more than 100%. Similarly, from Figure 4.8, although the packet dropping increases as packet generation rate, TD-CSMA networks always have lower dropping rates than CSMA networks. When the packet gen-

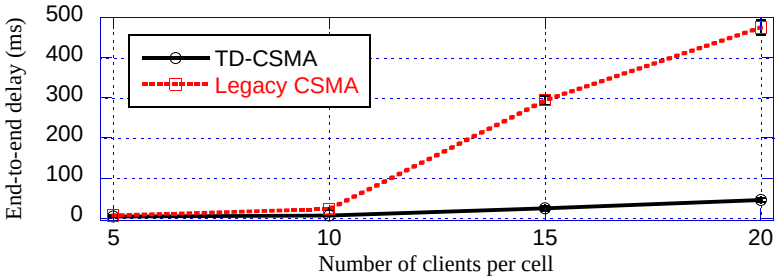

Figure 4.6. End-to-end delay vs. number of clients per cell

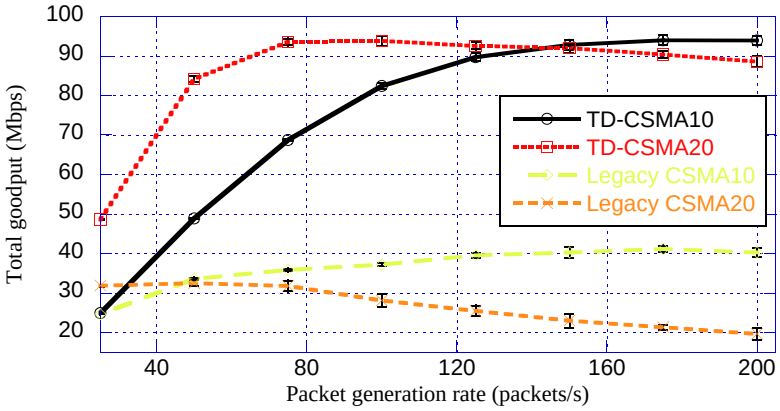

Figure 4.7. TD-CSMA vs. CSMA networks: total goodput of network

eration rate is high and the network is saturated, both TD-CSMA and legacy CSMA have traffic with large delays (Figure 4.9). TD-CSMA also cannot meet the delay requirement when the network is too saturated. There are two possible reasons for this phenomenon. First, in our implementation of TD-CSMA in NS2, the synchronization and time slots assignment are not done in a tight manner. We use beacon frames for synchronization and NAV [28] for time slot assignment. Synchronization may be lost in a saturated network. The second possible reason is that our model does not consider the control packets for routing protocols in NS2. These “extra” packets occupy the time slot and delay the normal data packets. Despite these adversities, TD-CSMA is still robust enough to incur smaller delay than the CSMA network. We will discuss the robustness of TD-CSMA in Section 4.4.3.

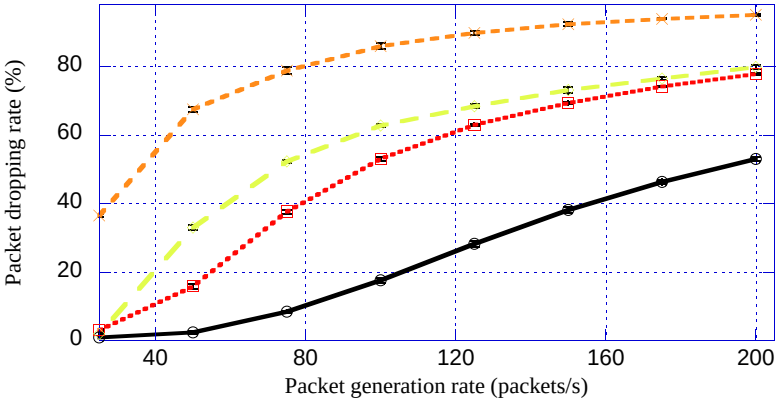

Figure 4.8. TD-CSMA vs. CSMA networks: packet dropping rate of the network

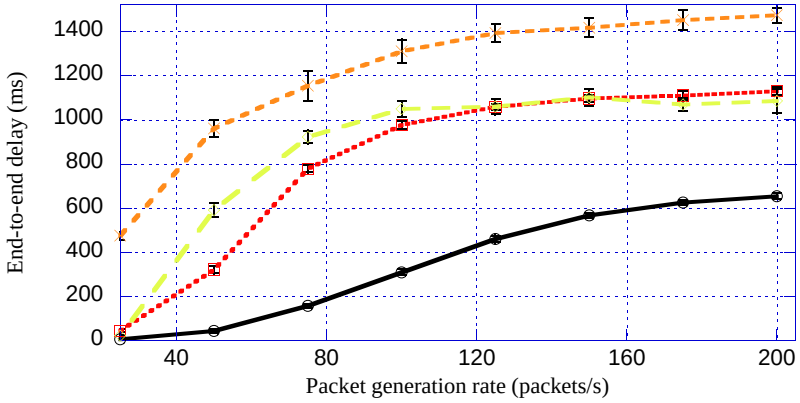

Figure 4.9. TD-CSMA vs. CSMA networks: end-to-end delay of the network

4.4.2 Scalability of TD-CSMA

To evaluate the scalability of TD-CSMA, we vary the topology size from 2-by-2 to 5-by-5, 15 clients per cell. To push the network to the limit, we make the clients generate 500-byte packets. For 2-by-2 networks, the packet generation rate is 85 packets per second, while for other topology sizes, as a lot more clients are there, we decrease the rate to 50 packets per second.

Figure 4.10 shows that TD-CSMA is more scalable than CSMA network. As the topology size and the number of clients increase, more goodput should be yielded. The total goodput of TD-CSMA has a steeper increase compared with CSMA. As the topology size increases, the distance between some nodes becomes large enough

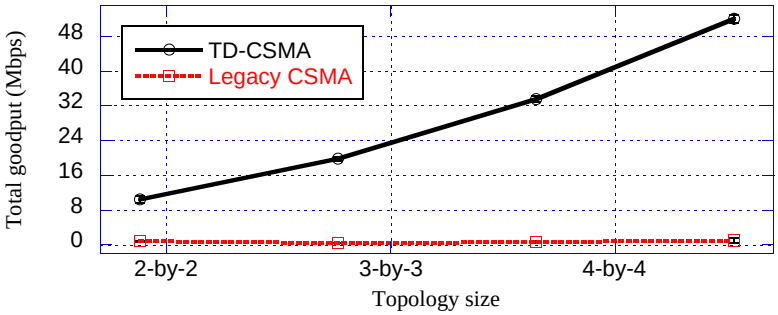

Figure 4.10. The scalability of TD-CSMA

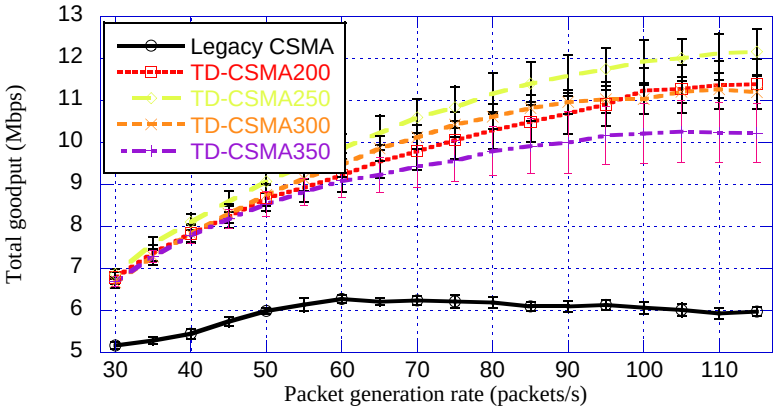

Figure 4.11. The robustness of TD-CSMA

to allow simultaneous transmission without interference. This phenomenon is obvious when the size increase from 2-by-2 to 3-by-3, as in the 2-by-2 network, the longest distance between two APs are still within the interference range of each other, but it is not the case in 3-by-3 network. Therefore, the increase in the number of simultaneous transmissions will decrease the packet dropping rate and delay. TD-CSMA takes this advantage and shows the drops in both dropping rate and delay. However, due to the large carrier-sensing range in CSMA (no clustering), its packet dropping rates and delay do not decrease much as the topology size increases. For lack of space, we do not reports the plots for packet drops and delay here.

4.4.3 Robustness of TD-CSMA

Although TD-CSMA schedules the activation of clusters of nodes according to specific packet size and packet rate, we are interested to see how the performance of the network would be affected if the network deviates from these settings. Another important parameter in TD-CSMA is the CS_{range}. We run simulations for various values of this parameters. Figure 4.11 shows the goodput of the CSMA and that of the TD-CSMA with different CS_{range} values. The TD-CSMA is optimally scheduled using a packet size of 500B and a packet generation rate of 85 packets per second. In the simulation, we set a packet size of 460B, varying the packet generation rate from 30 to 115 packets per second. It can be observed that TD-CSMA is robust to small perturbation of the packet size and generation rates. In fact, it still achieves a larger goodput than CSMA network. There are similar trends for the packet dropping rate and delay, so we avoid repeating the discussion here. Furthermore, we find that TD-CSMA's advantage is not affected by different CS_{range} values very much. We also notice a trade-off for increasing or decreasing the CS_{range} values. The larger this value is, the more the clients become eligible to be included in the same cluster, that in turn will reduce the number of clusters in the network, hence the traffic is easier to be scheduled. However, large carrier sensing ranges will cause more interference and thus limit the throughput capacity of the network. We conjecture there is an "optimal" value of the CS_{range} for the TD-CSMA.

4.5 Conclusions

In this chapter we have introduced a time division approach in the legacy Carrier Sensing Multiple Access MAC protocols. By grouping wireless clients and scheduling time slots to these groups, not only the delay of packet transmission can be decreased, but also the goodput of multiple WLANs can be largely increased. We show that, given a network topology and clients traffic demands, the problems of grouping the clients in clusters and scheduling them can be formulated by means of two integer programs, and can be solved optimally for small scale instances for up to 60 clients. We have also proposed heuristic approaches to compute feasible solutions for large scale instances in few tens of milliseconds.

Our simulation results show that although heuristics only provide sub-optimal scheduling, TD-CSMA demonstrates remarkable improvement in network performance compared to the legacy CSMA even without the tight synchronization among the stations. In a large-scale network, that contains 100 to 500 wireless clients (in a 5-by-5 WLAN network with different number of clients each WLAN), and when the packet generation rate is low, TD-CSMA can ensure a small delay under 100ms while the legacy CSMA suffers a delay more than 400ms. In addition, TD-CSMA increases the throughput capacity of the large-scale network by more than 100% compared to CSMA.

With this work, we provide a pioneering analytical exploration of incorporating the deterministic time-division scheduling into the random-based CSMA approach. The analytical framework and results give insights for providing quality of service in widely deployed 802.11 WLANs. It sheds light on supporting today's bandwidth-demanding applications in WLANs.

5

Towards resource-optimal routing plans for real-time traffic

Real-time traffic over IP networks has become a reality. Several applications, e.g. industrial control, remote sensing and surveillance systems, live IPTV and VoIP etc., all requiring real-time guarantees (i.e., a bound on the end-to-end delay) are increasingly being deployed. Internet Service Providers are already facing, or will soon face, the challenge of configuring their network domains so as to provide deterministic delay bound guarantees to their customers - whether single users or lower-tier providers themselves - by negotiating real-time oriented Service Level Agreements (SLAs). Supporting SLAs with real-time constraints requires proper Traffic Engineering (TE) and resource optimization practices. Multi-Protocol Label Switching (MPLS, [64]) with TE extensions (MPLS-TE, [2]) allows traffic trunks to be routed along arbitrary paths, and resources to be allocated on those paths at the time of flow setup.

Supporting delay constrained traffic requires in fact both computing paths and reserving resources along those paths. The usual assumption (to which we stick in the rest of the chapter) is that traffic trunks are scheduled at each node so as to be reserved a minimum guaranteed rate. As far as path computation is concerned, a relevant amount of literature has been published since the late '90s, under the name of QoS routing, a good review of which can be found in [65]. Most papers (see, e.g. [57, 54, 73]), assume that delays are static and/or additive per-link metrics. However, delay bounds do depend on the amount of reserved resources at each link, i.e. on the number and amount of flows traversing them, and the expression of the delay bound is not linear in the number of links. Other papers tackle the problem from a probabilistic point of view, assuming a stochastic characterization of traffic and attempting to minimize or bound the average delay, which is hardly relevant for real-time traffic (e.g. [65]). A limited number of works [55, 61] propose path computation techniques constrained by deterministic (non additive) delay bound constraints, taking resource allocation into account. [55] shows that it is possible to compute a shortest path for a single flow, subject to end-to-end delay bounds, also computing the rate to be reserved on each node during path computation, at a polynomial cost. It also assumes

that an equal rate has to be reserved at each node for the path. [61] proposes lower-complexity approximate solutions to the problem solved exactly in [55].

As far as resource allocation is concerned, the problem is often referred to as QoS partitioning in the literature. On that topic, several works exist that achieve optimal partitions for additive delays on a given path (see, e.g., [65]). An interesting work [21] shows that, when using end-to-end delay bounds as constraints, reserving the same rate (as done in [55, 61]) may be suboptimal and lead to failing of paths which might indeed be admissible. Authors propose an algorithm that allows a delay-feasible resource allocation to be computed on a given path, if such an allocation exists. To the best of our knowledge, the problem of making a global routing and resource allocation plan under delay bound constraints has received little attention so far. [61] claims that the problem is NP-hard . However, this does not mean that it is not solvable for practical dimensions (i.e., comparable to those of today's and tomorrow's network domains), nor it implies that good suboptimal solutions cannot be computed in reasonable time, even for large dimensions. Besides, global routing plans do not need to be computed in real time. Network engineering and optimization cycles - where new routing plans are made from scratch, based on the traffic forecast and negotiated SLAs - do not take place more frequently than daily or weekly, hence computation time can be traded for optimality. Second, per-path computation and resource allocation is feasible in a dynamic environment (online TE), but is clearly suboptimal when routing plans are considered (offline TE).

Our work marks a first step in this direction by tackling global resource allocation with delay bound constraints in a network domain. We assume that paths have been selected (and we evaluate several existing options for the path computation phase), and we exploit optimization techniques to minimize the amount of rate reserved in the network domain. We show that the problem can be solved optimally for several classes of schedulers. For some schedulers it has a convex formulation, which means that optimal solutions can be found in a reasonable time. Our first results show that, even at surprisingly low network loads, global allocation is necessary to be able to guarantee delay bounds when it is indeed feasible to do so, as per-path solutions are generally ineffective.

While in this work we assume that routing is given, and we only aim at optimizing the resource allocation, the long-term goal of this stream of research is to provide effective algorithms for joint path computation and resource reservation in a network domain, which is actively being pursued at the time of writing.

5.1 Related Work

In the context of resource-optimal real-time traffic strategies in wired networks, as far as path computation is concerned, a relevant amount of literature has been published since the late '90s, under the name of QoS routing, a good review of which can be found in [65]. Most papers (see, e.g. [57, 54, 73]), assume that delays are static

and/or additive per-link metrics. However, delay bounds do depend on the amount of reserved resources at each link, i.e. on the number and amount of flows traversing them, and the expression of the delay bound is not linear in the number of links. Other papers tackle the problem from a probabilistic point of view, assuming a stochastic characterization of traffic and attempting to minimize or bound the average delay, which is hardly relevant for real-time traffic (e.g. [65]). A limited number of works [55, 61] propose path computation techniques constrained by deterministic (non additive) delay bound constraints, taking resource allocation into account. [55, 61] shows that it is possible to compute a shortest path for a single flow, subject to end-to-end delay bounds, also computing the rate to be reserved on each node during path computation, at a polynomial cost. It also assumes that an equal rate has to be reserved at each node for the path. [61] proposes lower-complexity approximate solutions to the problem solved exactly in [55].

As far as resource allocation is concerned, the problem is often referred to as QoS partitioning in the literature. On that topic, several works exist that achieve optimal partitions for additive delays on a given path (see, e.g., [65]). An interesting work [21] shows that, when using end-to-end delay bounds as constraints, reserving the same rate (as done in [55, 61]) may be suboptimal and lead to failing of paths which might indeed be admissible. Authors propose an algorithm that allows a delay-feasible resource allocation to be computed on a given path, if such an allocation exists.

To the best of our knowledge, the problem of making a global routing and resource allocation plan under delay bound constraints has received little attention so far. [61] claims that the problem is NP-hard . However, this does not mean that it is not solvable for practical dimensions (i.e., comparable to those of today's and tomorrow's network domains), nor it implies that good suboptimal solutions cannot be computed in reasonable time, even for large dimensions. Besides, global routing plans do not need to be computed in real time. Network engineering and optimization cycles - where new routing plans are made from scratch, based on the traffic forecast and negotiated SLAs - do not take place more frequently than daily or weekly, hence computation time can be traded for optimality. Second, per-path computation and resource allocation is feasible in a dynamic environment (online TE), but is clearly suboptimal when routing plans are considered (offline TE).

5.2 System model

We represent a network domain through a graph $G \equiv \{V, E\}$, where V is a set of nodes, i.e. routers, and $E \equiv \{(i,j) : i,j \in V, i \neq j\}$ are a set of directed links. We assume that links are bidirectional, so that $(i,j) \in E \Rightarrow (j,i) \in E$. Each link is characterized by a physical link speed $W_{(i,j)}$, a propagation delay $pd_{(i,j)}$, both constant, and a reservable TE capacity $C_{(i,j)} \leq W_{(i,j)}$. While it is normally $W_{(i,j)} = W_{(j,i)}$ and $pd_{(i,j)} = pd_{(j,i)}$ due to technological constraints, the same cannot be said a priori regarding TE capacities without this affecting the generality. Nodes may have a constant

transit delay td_x, $x \in V$. This does not include queuing delays, which are variable and considered separately. Routers are further distinguished into core and edge nodes. Let $B \subseteq V$ be the set of edge routers, so that $F = \{B \times B\} \setminus \{(i,i), i \in B\}$ denotes the possible routes for traffic flowing through the domain. Note that we can account for local destinations inside the domain by including into B those LSRs where traffic is originated/destined. Furthermore, note that - in general - the same route can be connected by traversing different paths, i.e. sequences of nodes and links. Define a path that connects the route (i,e) as:

$$P_{(i,e)} = \left\{ \begin{array}{l} (x_j, y_j) \in E, 1 \leq j \leq N_{(i,e)} : \\ x_1 = i,\ y_{N_{(i,e)}} = e,\ \forall j : 1 \leq j \leq N_{(i,e)} - 1\ y_j = x_{j+1} \end{array} \right\}$$

i.e. a set of links that connect node i to node e. We are only interested in loop-free paths, i.e. those for which $j \neq k \Leftrightarrow y_j \neq y_k$. Note that the above formulation allows paths to be arbitrary, i.e. not to form a tree based at the destination node (as would happen instead with destination-based forwarding).

Within the domain, traffic trunks or flows have to be accommodated. The latter are characterized by a route $(i,e) \in F$, which denotes their ingress and egress points, a traffic constraint in the form of leaky-bucket parameters $\sigma_{(i,e)}, \rho_{(i,e)}$, and a required delay bound $\delta_{(i,e)}$. For the sake of readability (i.e., to avoid adding further subscripts), we describe the model under the assumption that one flow exists for a given route. The alert reader will easily notice that multiple flows on the same route can be accounted for in this model.

We assume that each link is managed by a packet scheduler, which arbitrates packets of different flows according to their reserved rates. The only assumption that we make on the scheduling algorithms is that they can be modeled via rate-latency service curves [45]. Several types of commonplace schedulers fit into this category, from Packet Generalized Processor Sharing (PGPS, [62]) to Worst-case Fair Weighted Fair Queuing (WF2Q, [6]), to Self Clocked Fair Queuing (SCFQ, [31]), to Deficit Round Robin (DRR, [69]). This kind of service curves, however, leaves out some popular schedulers, such as the well-known Earliest Deadline First (EDF), which have often been used in connection with QoS partitioning problems [25]. Schedulers need not be the same at each link for the model to be valid: nevertheless, we will often assume so when performing the analysis, especially to show that different schedulers lead to different performance. A rate-latency scheduler is able to divide the reservable TE capacity among the flows, giving to each one a reserved (long-term) rate, subject to a short-term vacation called latency. We denote with $R_{(x,y),(i,e)}$ the reserved rate at link (x,y) for flow (i,e). The latter may be null, for instance if flow (i,e) does not traverse link (x,y). TE capacity constraints need be accounted for, which is done by ensuring that:

$$\sum_{(i,e) \in F} R_{(x,y),(i,e)} \leq C_{(x,y)}. \tag{5.1}$$

The latency is denoted by $\theta_{(x,y),(i,e)}$, and it is monotonically decreasing with the reserved rate. The exact expression for the latency is scheduler-specific. We will come

back to this later on, showing that rate-latency schedulers may fall into three categories, thus giving birth to slightly different formulations for the problem.

Whatever the expression for the latency, the end-to-end delay for a flow along path $P_{(i,e)}$ is the following:

$$D_{P_{(i,e)}} = \sum_{(x,y)\in P_{(i,e)}} \left[\theta_{(x,y),(i,e)} + tp_{(x,y)} + td_x\right] + \frac{\sigma_{(i,e)}}{\min\limits_{(x,y)\in P_{(i,e)}} \left\{R_{(x,y),(i,e)}\right\}}, \quad (5.2)$$

provided that: $\min\limits_{(x,y)\in P_{(i,e)}} \left\{R_{(x,y),(i,e)}\right\} \geq \rho_{(i,e)}$, and $D_{P_{(i,e)}} = \infty$ otherwise.

5.2.1 Scheduling and latency

That of packet scheduling for wired networks has been a flourishing literature stream during the last two decades (see, e.g., [74]). Some (actually, a minority) of the devised scheduling algorithms have been implemented in commercial routers (e.g., [69, 50]), or made their way into the codebase of open-source operating systems (e.g., [69]). There are three main expressions for latency, to which we associate names for ease of notation. A good survey on the subject can be found in [70]. Call L the Maximum Transmit Unit (MTU) in the network (assumed to be equal at all links and for all flows for notational simplicity, although the model can be easily generalized). The following latency expressions can be defined.

1. Strictly rate-proportional (SRP) latency (PGPS [62], WF2Q [6], Virtual Clock [46], etc.):
$$\theta_{(x,y),(i,e)} = \frac{L}{R_{(x,y),(i,e)}} + \frac{L}{W_{(x,y)}} \quad (5.3)$$
2. weakly rate proportional (WRP) latency (e.g., Self-Clocked Fair Queuing, [31]):
$$\theta_{(x,y),(i,e)} = \frac{L}{R_{(x,y),(i,e)}} + \left(n_{(x,y)} - 1\right) \cdot \frac{L}{W_{(x,y)}}, \quad (5.4)$$
where $n_{(x,y)}$ is the number of flows traversing link (x, y).
3. frame-based (FB) latency (e.g. DRR [69], [47][1]):
$$\theta_{(x,y),(i,e)} = \frac{L}{W_{(x,y)}} \left[\left(W_{(x,y)} - R_{(x,y),(i,e)}\right) \cdot \left(\frac{1}{\min\limits_{(a,b):(x,y)\in P_{(a,b)}} \left\{R_{(x,y),(a,b)}\right\}} + \frac{1}{R_{(x,y),(i,e)}} \right) + n_{(x,y)} \right] \quad (5.5)$$

Other frame-based schedulers have recently been derived from DRR (e.g. EBDRR [46], ALiQueM [47] etc.), and improve on its latency by dividing some of the above

[1] The latency expression reported here can be worked out via straightforward algebraic manipulations from the one reported in [47]. The one in [70] is instead an overrated bound.

addenda by a constant term. As the purpose of our work is to investigate resource allocation under delay constraints (rather than surveying all possible schedulers) we leave these minor generalization to the interested reader.

In all three cases, increasing the rate of a flow decreases its latency, although the effectiveness of such a tuning clearly decreases when we move from category 1) to 3). In fact, 2) contains an $n_{(x,y)}$ term, whereas 3) includes the latter and a minimum rate at the denominator, which cannot be modified by increasing $R_{(x,y),(i,e)}$.

5.2.2 Path computation algorithms

The path computation algorithms that we consider in this work are the following:

1. Constrained Multicommodity Shortest Path First (CM-SPF): assuming that links are characterized by capacities and have unitary weights, it computes the shortest path from a source to a destination having at least the required capacity. All requests are considered jointly, and the result is the set of paths having the minimum total number of hops.
2. Constrained Shortest Path First (C-SPF): the same as the previous one, but with sequential computations, so that the outcome depends on the order in which path requests are considered.
3. Widest-Shortest Path First (WPF): the same as 2), with the difference that links have weights which are inversely proportional to the residual capacity on each link.
4. Maximum Maxflow (MM, [41]): for each request, the path that yields the maximum (weighted) sum of the maxflows between any source and destination pair is selected. This way, the ability to route future requests between a source and destination is maximized, though generally at the expenses of having longer paths. MM was proved in [41] to be NP-hard. Authors propose a heuristic algorithm (called Minimum Interference Routing, MIRA) to approximate the MM solution in polynomial time.

5.3 Optimal resource allocation

A joint routing and resource allocation problem can be formulated as follows:

Joint Routing and Resource Allocation Problem (JRRA)

For each flow (i, e), i) compute a path $P_{(i,e)}$ and ii) reserve a rate on all the links in $P_{(i,e)}$ (subject to constraints (5.1)) so that $D_{P_{(i,e)}} \leq \delta_{(i,e)}$, if it is possible to do so.

The above one is a feasibility problem, claimed to be NP-hard in [61] (where, however, it is formulated assuming that $R_{(x,y),(i,e)} = R_{(i,e)}$, although we do not believe that relaxing the above constraint is going to make the problem any easier). It can be turned into an optimization problem once a suitable objective function to be minimized or maximized is identified. Several such functions can be envisaged, such as:

1. maximizing the minimum slack with respect to the delay bound. A non-positive objective means that all slacks are non negative, i.e. that all delay bound inequalities are verified, hence the solution is feasible. It has been shown in [14], although in a slightly different context, that this formulation leads to robust schedules, which can easily tolerate uncertainties in the parameters. On the cons side, such an approach tends to allocate rates too liberally, thus depleting the resources. Even though we deal with a static environment, where all requests are known a priori, it is intuitively reasonable to try to minimize the amount of reserved rate, so as to leave the maximum possible room for future requests or cope with parameter uncertainties.
2. Maximizing the total unreserved capacity in the network, i.e. the sum of the slacks in (5.1), again having the delay bounds as constraints. This allows for a higher number of future requests to be considered. If necessary, a cost $c_{(x,y)}$ can be statically associated to a capacity unit on each link, so as to reflect their relative importance.
3. Maximizing the sum of the maxflows between each source/destination pairs.

As already anticipated, in this chapter we do not solve the above problem, but instead mark a first step in that direction by researching the utility of global resource minimization techniques. We reformulate the resource allocation sub-problem as follows:

Global Resource Allocation Problem (GRA)

Given a set of paths $P_{(i,e)}$ for all flows $(i,e) \in F$, compute the vector of the allocated rates along a path $P_{(i,e)}$, $\mathbf{R} = \left\{R_{(x,y)}, (x,y) \in P_{(i,e)}\right\}$, (subject to constraints (5.1)) so that i) $D_{P_{(i,e)}} \leq \delta_{(i,e)}$, and ii) the sum of the allocated rates is minimum, if it is possible to do so.

The GRA problem assumes that paths have been precomputed, using any of the techniques described in Section 5.2.2. Given the flow routes, leaky-bucket profiles and deadline requirements, the GRA problem can be formulated as the following optimization problem:

$$\begin{array}{lll} \min & \sum\limits_{(i,e)} \sum\limits_{(x,y)\in P_{(i,e)}} R_{(x,y),(i,e)} & \\ \text{s.t.:} & D_{P_{(i,e)}} \leq \delta_{(i,e)} & \forall (i,e) \in F \\ & R_{\min}^{(i,e)} \leq R_{(x,y),(i,e)} & \forall (x,y) \in P_{(i,e)}, \forall (i,e) \in F \end{array} \tag{5.6}$$

We select the sum of the allocated rates as the objective to be minimized, without considering link costs. The latter can obviously be added back if necessary, without changing the nature of the problem. The 1^{st} constraint ensures that all the flows meet their deadline delay requirements on their respective paths. The $D_{P_{(i,e)}}$ delay expression can be defined using any of the three latencies defined in Section 5.2.1 2.1. Latencies (5.3) and (5.4) are convex, since they involve summations of convex function, while (5.5) is not defined. Hence the resulting GRA problems are convex non-linear

optimization problems for latencies (5.3) and (5.4), non-convex non-linear optimization problems for latency in (5.5) respectively. The non linear constraints of the convex formulations contain hyperbolic constraints that can be reformulated as Second Order Cone Programming (SOCP) programming [53] and solved using a solver for quadratically constrained programs. In this case, interior point methods can be used, which complete in polynomial time and are generally very fast. In the non convex case, instead, global optimization is required, which is considerably more complex. We solved these problems using general purpose solvers such as CPLEX [17] and BARON [66].

To deal with the min operator in (5.2), additional variables $R_{\min}^{(i,e)}$, representing the minimum rate on a path $P_{(i,e)}$, are required: the 2^{nd} constraints couple these variables with the link rates. Since the objective function is a minimization problem of the link rates, these variables will be assigned the lowest rate that guarantees that no deadline is violated, according to the 1^{st} constraint. If the latency defined in (5.5) is used instead, additional variables $R_{\min}^{(x,y)}$, representing the minimum rate at a link (x, y) (among those allocated to the flows traversing that link), are also required together with the following constraints:

$$R_{\min}^{(x,y)} \leq R_{(x,y),(i,e)} \ \forall (x,y) \in P_{(i,e)}, \forall (i,e) \in F.$$

The above problem has $O\left(|F| \cdot |E|\right)$ variables and constraints.

As already observed in [21], allocating the same rate at all links for a flow, i.e.

$$R_{(x,y),} = R_{(i,e)}, \ \ (x,y) \in P_{(i,e)},$$

may lead to suboptimal rate allocations. However, under that assumption the allocation problem can be solved analytically, once the path and the schedulers at all links are known. Hence, we will use this approach (henceforth referred to as equal rate allocation, ERA) as a comparison, to test how much can be harvested by using global minimization. The minimum required rate on a single path can be computed by solving (5.2) with respect to the rate, under the assumption that $D_{P_{(i,e)}} = \delta_{(i,e)}$. Such allocation is in fact exploited by RSVP to compute the rates to allocate at each link for IntServ guaranteed-rate connections [11]. We remark that, since this allocation is done path-wise, then 5.1 may be violated even if a feasible (e.g. globally computed) rate allocation exists.

For strictly and weakly rate-proportional latencies (5.3) and (5.4), the alert reader can check that the ERA solutions are the following:

$$R_{(i,e)} = \max\left(\rho_{(i,e)}, \frac{\left|P_{(i,e)}\right| \cdot L + \sigma_{(i,e)}}{\delta_{(i,e)} - \sum_{(x,y)\in P_{(i,e)}} \left[\frac{L}{W_{(x,y)}} + tp_{(x,y)} + td_x\right]}\right), \tag{5.7}$$

$$R_{(i,e)} = \max\left(\rho_{(i,e)}, \frac{\left|P_{(i,e)}\right| \cdot L + \sigma_{(i,e)}}{\delta_{(i,e)} - \sum\limits_{(x,y)\in P_{(i,e)}} \left[\left(n_{(x,y)} - 1\right) \cdot \frac{L}{W_{(x,y)}} + tp_{(x,y)} + td_x\right]}\right), \tag{5.8}$$

if the denominator is positive (which is a necessary condition for the problem to have a solution at all to the allocation problem, whether global or path-wise). $\left|P_{(i,e)}\right|$ denotes the number of hops in path $P_{(i,e)}$.

On the other hand, for frame-based latency (5.5), computing the same required rate $R_{(i,e)}$ is considerably more involved, and requires global optimization, due to the min term in that expression. In fact, since the latency at a link depend on the minimum rate allocated at a link (possibly to some other flow), all flows traversing the same link should be considered simultaneously in order to determine the latency of each one. However, we can do a reasonable approximation by considering that, by definition, it is:

$$R_{(i,e)} \geq \min_{\substack{(a,b):(x,y)\in P_{(a,b)} \\ (x,y)\in P_{(i,e)}}} \left\{R_{(x,y),(a,b)}\right\}. \tag{5.9}$$

Hence, we can obtain a rate $R_{(i,e)}$ that leads to feasible delays (if that rate is indeed available at all links), by merging (5.5) and (5.2) and assuming that equality holds in (5.8):

$$R_{(i,e)} = \max\left(\rho_{(i,e)}, \frac{2\left|P_{(i,e)}\right| \cdot L + \sigma_{(i,e)}}{\delta_{(i,e)} - \sum\limits_{(x,y)\in P_{(i,e)}} \left[\left(n_{(x,y)} - 2\right) \cdot \frac{L}{W_{(x,y)}} + tp_{(x,y)} + td_x\right]}\right). \tag{5.10}$$

Note that, with (5.10), it is $D_{P_{(i,e)}} \leq \delta_{(i,e)}$, and inequality may actually hold, i.e. the rate can be overprovisioned with respect to the one strictly required to meet the deadline. The alert reader can easily check that the amount of required rate to meet a given deadline increases with the latency model, from (5.7) to (5.8) and (5.10).

5.4 Numerical results

In order to prove the effectiveness of the proposed approach, we performed simulations on a sample network, shown in Figure 5.1. Links are bidirectional, with either $W_{(x,y)}$=35Mbps or $W_{(x,y)}$=75Mbps speed. We assume that $tp_{(x,y)} = 0$, $td_x = 0$ and $C_{(x,y)} = W_{(x,y)}$ for simplicity. Furthermore, it is $L = 1.5$kB. We generate 96 flows, between random pairs of nodes. All the flows have homogeneous QoS requirements with $\rho = 1$Mbps, $\sigma = 12$kB and $\delta = 15$ms. We use all the path computation algorithms mentioned in Section 5.2.2, and all the latency models of Section 5.2.1, and

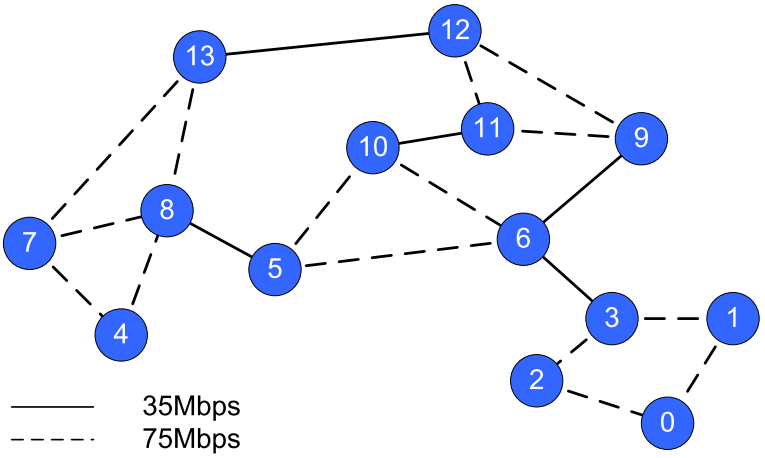

Figure 5.1. Sample network for numerical analysis

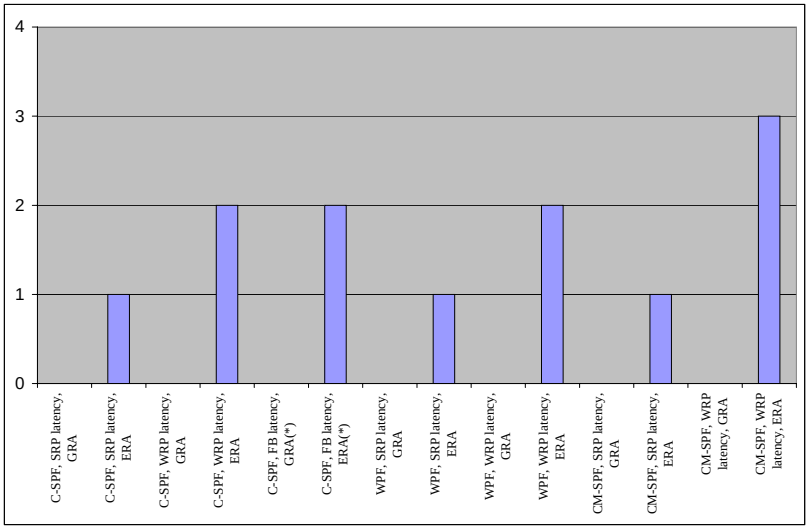

Figure 5.2. Number of oversubscribed links

we compute both the optimal solution of the GRA in the above settings and the ERA (5.7), (5.8) and (5.10).

ERA is known to be optimal when link capacities are unbounded [65]. While this seems to imply that at low loads, i.e., when capacity bounds are not active constraints,

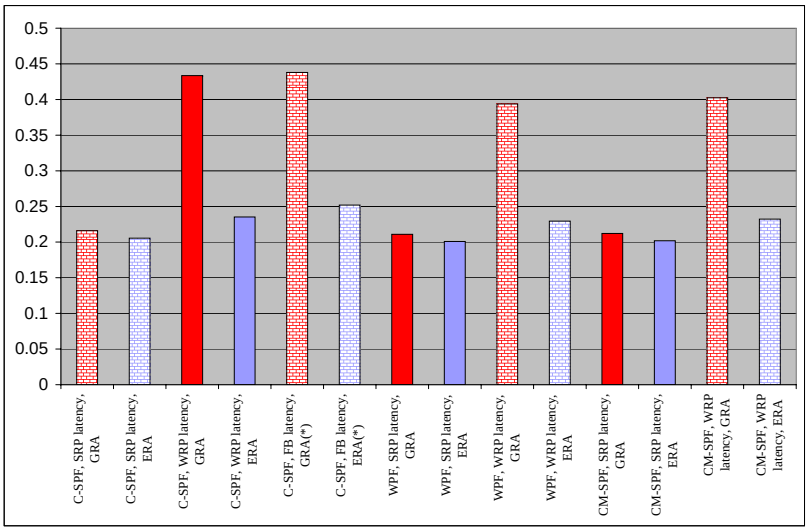

Figure 5.3. Average utilization for the links

an equal allocation is the optimal one, we show that even at low utilizations this approach may yield infeasible solutions. For instance, in our case the total available bandwidth is 2530Mbps, and the overall rate demand is 96Mbps, which is 3.8%. Using ERA, the average link utilization is between 20% and 24%, depending on the path computation scheme and latency adopted. Yet, there are links whose capacity is exceeded, thus leading to an unfeasible solution, whereas a feasible one can be found by optimally solving the GRA.

Consider for instance the case of C-SPF and strictly rate-proportional latencies. In this case, the capacity reserved by ERA at link (6,3) exceeds the available one by 6.28%, and accordingly bounds cannot be guaranteed to all flows traversing that link. On the other hand, the optimal solution to the GRA problem is feasible, as it exploits the ability to assign different rates on different links of the same flow path. For instance, the optimal assignment for flow (8,0) - which traverses link (6,3) - is the following:

Link (x,y)	$R_{(x,y),(8,0)}$ (Mbps)	Link utilization
(8,5)	3.90	1
(5,6)	6.91	1
(6,3)	1.14	1
(3,2)	5.52	1
(2,0)	10.75	0.97

The ERA rate would instead have been 1.16 Mbps, i.e. slightly more (although critically so) than the one allocated by GRA on the critical link. Note that the lack of rate at the critical link is made up for by allocating more resources on uncongested peripheral links, so that the allocated rate is highly inhomogeneous. For flows traversing unloaded links the rate provided using the analytical framework is the optimal solution as shown for the rate assignment of flow (12, 4):

Link (x,y)	$R_{(x,y),(12,4)}$	Link utilization
(7,4)	1.16	0.11
(13,7)	1.16	0.14
(12,13)	1.16	0.56

On a more general note, we can also derive some considerations from aggregated utilization data. Figure 5.2 shows the number of oversubscribed links for each path computation, rate-proportional latency model and resource allocation scheme. Note that, for frame-based latency, we only report one example, using C-SPF as path computation (due to the larger overhead of solving non-convex problems). Furthermore, the scenarios with FB latency are with a reduced number of flows (80 instead of 96), otherwise the problem is not feasible. As the figure shows, GRA always finds feasible solutions, whereas ERA is always unfeasible. The oversubscribed links with ERA are (6,3), whereas (3,6) is oversubscribed under WRP and FB latency, and (5,8), only with CM-SPF and WRP latency. Data related to the MM path computation scheme were not reported, as no feasible solution to the GRA can be found for the scenario under consideration in that sense.

Figure 5.3 shows the average utilization for the links under the various configurations. The figure shows that, while the latency model does not play a big difference with ERA (few percentage points), it does so under GRA, where WRP latency requires almost double as many resources to be allocated than SRP latency, in order to maintain feasible delays. The figure also shows that negligible differences exist among the path computation schemes (except for the case of MM, as already explained). The main difference between MM and the rest is that the former computes considerably longer paths, so that the amount of resources required to maintain feasible delays is multiplied by a higher path length.

Figure 5.4 and 5.5 show the utilization per link under WRP latency and CM-SPF, with both ERA and GRA. Consecutive pairs of links are the forward and reverse direction of the link showed in the horizontal axis. As already explained, ERA oversubscribes three links, whereas GRA does not. What is remarkable in that figure is that the amount of resources that are required in order to keep the delay bounds within

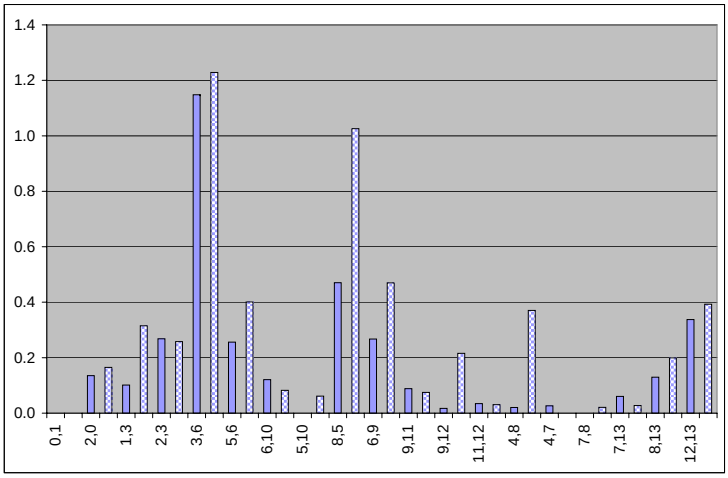

Figure 5.4. Link utilization - CM-SPF, with WRP latency, ERA

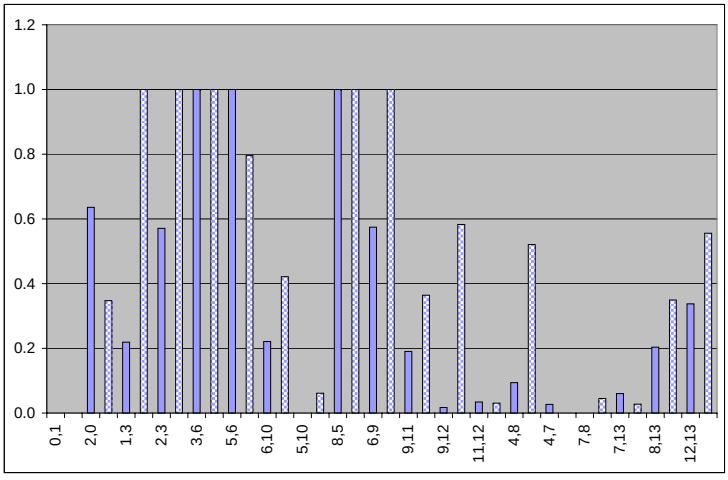

Figure 5.5. Link utilization - CM-SPF, with WRP latency, GRA

s bounded is indeed taxing, with eight links fully booked. This seems to suggest that there is room for improving the efficiency of the allocation by jointly solving routing and resource allocation.

5.5 Conclusions

This chapter explored the space for global optimization in resource allocation for guaranteed-delay traffic engineering. We formulated and solved the problem under different latency models, showing that the adopted latency does indeed make a difference as far as resource consumption is concerned. Our results on a case-study network show that, even at surprisingly low average loads, using global optimization can help feasible schedules to be computed, whereas local resource allocation schemes would fail to do so.

This work marks the first exploration in a rather broad research field, and, as such, calls for extension along several directions. The first one is to derive a joint framework for path computation and resource allocation, taking into account real-time constraint. Second, we are currently considering including stochastic network calculus [39] in the framework. This would allow us to relax the assumption of deterministic worst-case delay, while still retaining quantifiable probabilistic guarantees, although possibly at the expenses of additional hypotheses on the traffic. This, in turn, would allow us to capitalize on statistical multiplexing, possibly increasing the amount of carried traffic within the network. Third, while the pipe (i.e., point-to-point) path model is probably the most widely used in TE practices, the funnel and hose models (i.e., multipoint-to-point and point-to-multipoint respectively) can also be used. In those cases, resource allocation is done on a per-tree basis, and delay bounds have different formulations [49]. Analyzing these networks is part of the ongoing work.

6

Conclusions

In this thesis we applied operations research tools and technologies to model, solve and analyze resource allocation problems in computer networks. We first addressed the problem of routing and link scheduling for real-time traffic in Wireless Mesh Networks. The problem has been formulated as an integer non-linear optimization problem. We showed that the feasibility of a link schedule does depend on the aggregation framework, and we derived guidelines to choose the appropriate aggregation framework given a network scenario. We then proposed a heuristic solution approach that computes good suboptimal schedules for WMN of larger sizes and/or in smaller times. Finally, we studied the problem of jointly solving the routing and link scheduling problem optimally, taking into account end-to-end delay guarantees. We formulated an optimization problem and we devised a heuristic to reduce the complexity of the given formulation. We have used the above technique to identify guidelines for the optimal placing of gateways in the WMN.

As a second resource allocation problem, we exploited the proposed scheduling formulation to introduce a time division approach in the legacy Carrier Sensing Multiple Access MAC protocols. By grouping wireless clients and scheduling time slots to these groups, not only the delay of packet transmission can be decreased, but also the goodput of multiple WLANs can be largely increased. Operations Research, again, played a major role in the solution of both the client grouping and the group scheduling tasks, resulting in effective and efficient solutions. TD-CSMA showed remarkable improvement in network performance compared to the legacy CSMA even without a tight synchronization among the stations. In a large-scale network and when the packet generation rate is low, TD-CSMA can ensure a small delay under 100ms while the legacy CSMA suffers a delay more than 400ms. In addition, TD-CSMA increases the throughput capacity of the large-scale network by more than 100% compared to CSMA.

Finally, we explored the space for global optimization in wired networks resource allocation for guaranteed-delay traffic engineering. We formulated and solved the problem under different latency models, showing that the adopted latency does in-

deed make a difference as far as resource consumption is concerned. Our results on a case-study network show that, even at surprisingly low average loads, using global optimization can help feasible schedules to be computed, whereas local resource allocation schemes would fail to do so.

References

1. F.N. Ali, P.K. Appani, J.L. Hammond, V.V. Mehta, D.L. Noneaker, and H.B. Russell. Distributed and adaptive tdma algorithms for multiple-hop mobile networks. In *MILCOM 2002. Proceedings*, volume 1, pages 546 – 551 vol.1, October 2002.
2. D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for traffic engineering over mpls, 1999.
3. L. Badia, A. Erta, L. Lenzini, and M. Zorzi. Scheduling, routing, and related cross-layer management through link activation procedures in wireless mesh networks. In Ekram Hossain and Kin Leung, editors, *Wireless Mesh Networks*, pages 191–226. Springer US, 2007.
4. Cynthia Barnhart. Dual-ascent methods for large-scale multicommodity flow problems. *Naval Research Logistics (NRL)*, 40(3):305–324, 1993.
5. Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1996.
6. Jon C. R. Bennett and Hui Zhang. WF2Q: Worst-Case Fair Weighted Fair Queueing. In *INFOCOM 1996*, pages 120–128, 1996.
7. R. Bhatia and M. Kodialam. On power efficient communication over multi-hop wireless networks: joint routing, scheduling and power control. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1457 – 1466 vol.2, March 2004.
8. G. Bianchi. Performance analysis of the ieee 802.11 distributed coordination function. *Selected Areas in Communications, IEEE Journal on*, 18(3):535 –547, March 2000.
9. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated service, 1998.
10. BONMIN user's manual. `http://www.coin-or.org/Bonmin/`.
11. R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: an overview, 1994.
12. Zhijun Cai and Mi Lu. Sndr: a new medium access control for multi-channel ad hoc networks. In *Vehicular Technology Conference Proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st*, volume 2, pages 966 –971 vol.2, 2000.
13. P. Cappanera, L. Lenzini, A. Lori, G. Stea, and G. Vaglini. Link scheduling with end-to-end delay constraints in wireless mesh networks. In *World of Wireless, Mobile and Multimedia Networks Workshops, 2009. WoWMoM 2009. IEEE International Symposium on a*, pages 1 –9, June 2009.
14. Paola Cappanera, Luciano Lenzini, Alessandro Lori, Giovanni Stea, and Gigliola Vaglini. Optimal link scheduling for real-time traffic in wireless mesh networks in both per-flow and

per-path frameworks. In *World of Wireless Mobile and Multimedia Networks (WoWMoM), 2010 IEEE International Symposium on a*, pages 1–9, June 2010.
15. An Chan and Soung Chang Liew. Performance of voip over multiple co-located ieee 802.11 wireless lans. *Mobile Computing, IEEE Transactions on*, 8(8):1063 –1076, August 2009.
16. I. Chlamtac, A. Farago, A.D. Myers, V.R. Syrotiuk, and G. Zaruba. Adapt: a dynamically self-adjusting media access control protocol for ad hoc-networks. In *Global Telecommunications Conference, 1999. GLOBECOM '99*, volume 1A, pages 11 –15 vol.1a, 1999.
17. IBM ILOG CPLEX software, v. 12. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`.
18. J. Crichigno, M.Y. Wu, J. Khoury, and W. Shu. A joint routing and scheduling scheme for wireless networks with multi-packet reception and directional antennas. In *World of Wireless, Mobile and Multimedia Networks Workshops, 2009. WoWMoM 2009. IEEE International Symposium on a*, pages 1 –9, June 2009.
19. R.L. Cruz and A.V. Santhanam. Optimal routing, link scheduling and power control in multihop wireless networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 1, pages 702 – 711 vol.1, March 2003.
20. George B. Dantzig. Discrete-Variable Extremum Problems. *Operations Research*, 5(2), 1957.
21. A. Diwan, J. Kuri, and A. Kumar. Optimal per-node rate allocation to provide per-flow end-to-end delay guarantees in a network of routers supporting guaranteed service class. In *Communications, 2002. ICC 2002. IEEE International Conference on*, volume 2, pages 1112 – 1117 vol.2, 2002.
22. P. Djukic and P. Mohapatra. Soft-tdmac: A software tdma-based mac over commodity 802.11 hardware. In *INFOCOM 2009, IEEE*, pages 1836 –1844, April 2009.
23. P. Djukic and S. Valaee. Delay aware link scheduling for multi-hop tdma wireless networks. *Networking, IEEE/ACM Transactions on*, 17(3):870 –883, June 2009.
24. J. El-Najjar, C. Assi, and B. Jaumard. Joint routing and scheduling in wimax-based mesh networks: A column generation approach. In *World of Wireless, Mobile and Multimedia Networks Workshops, 2009. WoWMoM 2009. IEEE International Symposium on a*, pages 1 –10, June 2009.
25. K.M.F. Elsayed. A framework for end-to-end deterministic-delay service provisioning in multiservice packet networks. *Multimedia, IEEE Transactions on*, 7(3):563 – 571, June 2005.
26. Antonio Frangioni. Generalized bundle methods. *SIAM J. on Optimization*, 13:117–156, May 2002.
27. Antonio Frangioni and Giorgio Gallo. A bundle type dual-ascent approach to linear multicommodity min-cost flow problems. *INFORMS J. on Computing*, 11:370–393, April 1999.
28. Matthew Gast and Matthew S. Gast. *802.11 Wireless Networks: The Definitive Guide (O'Reilly Networking)*. O'Reilly Media, 1 edition, 2002.
29. P. C. Gilmore and R. E. Gomory. A Linear Programming Approach to the Cutting-Stock Problem. *Operations Research*, 9(6):849–859, 1961.
30. P. C. Gilmore and R. E. Gomory. A Linear Programming Approach to the Cutting Stock Problem–Part II. *OPERATIONS RESEARCH*, 11(6):863–888, Nov 1963.
31. S.J. Golestani. A self-clocked fair queueing scheme for broadband applications. In *INFOCOM '94. Networking for Global Communications., 13th Proceedings IEEE*, pages 636 –646 vol.2, June 1994.
32. O. Goussevskaia, R. Wattenhofer, M.M. Halldorsson, and E. Welzl. Capacity of arbitrary wireless networks. In *INFOCOM 2009, IEEE*, pages 1872 –1880, April 2009.
33. I. E. Grossmann and Z. Kravanja. Mixed-integer nonlinear programming: A survey of algorithms and applications. In Biegler, Coleman, Conn, and Santosa, editors, *The IMA Volumes in Mathematics and its Applications, Vol.93, Large-Scale Optimization with Applications. Part II: Optimal Design and Control*, pages 73–100. Springer Verlag, 1997.

34. P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, mar 2000.
35. Kamal Jain, Jitendra Padhye, Venkata N. Padmanabhan, and Lili Qiu. Impact of interference on multi-hop wireless network performance. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, MobiCom '03, pages 66–80, New York, NY, USA, 2003. ACM.
36. P. Jayachandran and M. Andrews. Minimizing end-to-end delay in wireless networks using a coordinated edf schedule. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –9, March 2010.
37. Li Bin Jiang and Soung Chang Liew. Hidden-node removal and its application in cellular wifi networks. *Vehicular Technology, IEEE Transactions on*, 56(5):2641 –2654, September 2007.
38. Li Bin Jiang and Soung Chang Liew. Improving throughput and fairness by reducing exposed and hidden nodes in 802.11 networks. *Mobile Computing, IEEE Transactions on*, 7(1):34 –49, January 2008.
39. Yuming Jiang and Yong Liu. *Stochastic Network Calculus*. Springer Publishing Company, Incorporated, 1 edition, 2008.
40. Akimitsu Kanzaki, Takahiro Hara, and Shojiro Nishio. An adaptive tdma slot assignment protocol in ad hoc sensor networks. In *Proceedings of the 2005 ACM symposium on Applied computing*, SAC '05, pages 1160–1165, New York, NY, USA, 2005. ACM.
41. K. Kar, M. Kodialam, and T.V. Lakshman. Minimum interference routing of bandwidth guaranteed tunnels with mpls traffic engineering applications. *Selected Areas in Communications, IEEE Journal on*, 18(12):2566 –2579, December 2000.
42. J. Kazemitabar, V. Tabatabaee, and H. Jafarkhani. Global optimal routing, scheduling and power control for multi-hop wireless networks with interference. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1 –5, December 2008.
43. Murali S. Kodialam and Thyaga Nandagopal. Characterizing the capacity region in multi-radio multi-channel wireless mesh networks. In *MOBICOM*, pages 73–87, 2005.
44. Murali S. Kodialam and Thyagarajan Nandagopal. Characterizing achievable rates in multi-hop wireless networks: the joint routing and scheduling problem. In *MOBICOM*, pages 42–54, 2003.
45. Jean Y. Le Boudec. *Network Calculus*. Springer Verlag, LNCS 2050, New York, NY, 2001.
46. L. Lenzini, E. Mingozzi, and G. Stea. Eligibility-based round robin for fair and efficient packet scheduling in wormhole switching networks. *Parallel and Distributed Systems, IEEE Transactions on*, 15(3):244 – 256, March 2004.
47. L. Lenzini, E. Mingozzi, and G. Stea. Tradeoffs between low complexity, low latency, and fairness with deficit round-robin schedulers. *Networking, IEEE/ACM Transactions on*, 12(4):681 – 693, August 2004.
48. Luciano Lenzini, Linda Martorini, Enzo Mingozzi, and Giovanni Stea. A novel approach to scalable cac for real-time traffic in sink-tree networks with aggregate scheduling. In *valuetools '06: Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, page 9, New York, NY, USA, 2006. ACM.
49. Luciano Lenzini, Linda Martorini, Enzo Mingozzi, and Giovanni Stea. Tight end-to-end per-flow delay bounds in fifo multiplexing sink-tree networks. *Performance Evaluation*, 63(9-10):956–987, October 2006.
50. Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. Performance analysis of modified deficit round robin schedulers. *Journal of High Speed Networks*, 16(4):399–422, 2007.
51. Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. A methodology for computing end-to-end delay bounds in fifo-multiplexing tandems. *Performance Evaluation*, 65(11-12):922 – 943, 2008. Performance Evaluation Methodologies and Tools: Selected Papers from ValueTools 2007.

52. M. Leoncini, P. Santi, and P. Valente. An stdma-based framework for qos provisioning in wireless mesh networks. In *Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on*, pages 223 –232, October 2008.
53. Miguel Sousa Lobo, Lobo I, Lieyen Vandenberghe, Herv Lebret, and Stephen Boyd. Applications of second-order cone programming. *Linear Algebra and its Applications*, pages 284(1–3):193–228, November 2007.
54. D.H. Lorenz and A. Orda. Optimal partition of qos requirements on unicast paths and multicast trees. *Networking, IEEE/ACM Transactions on*, 10(1):102 –114, February 2002.
55. Qingming Ma and Peter Steenkiste. Quality-of-service routing for traffic with performance guarantees. In *In Proc. IFIP International Workshop on Quality of Service*, pages 115–126, 1997.
56. A. Mehrotra and M. Trick. A column generation approach for graph coloring. *INFORMS Journal On Computing*, 8(4):344–354, 1996.
57. S. Misra, Guoliang Xue, and Dejun Yang. Polynomial time approximations for multi-path routing with bandwidth and delay constraints. In *INFOCOM 2009, IEEE*, pages 558 –566, April 2009.
58. Girija Narlikar, Gordon Wilfong, and Lisa Zhang. Designing multihop wireless backhaul networks with delay guarantees. *Wirel. Netw.*, 16:237–254, Jan 2010.
59. R. Nelson and L. Kleinrock. Spatial tdma: A collision-free multihop channel access protocol. *Communications, IEEE Transactions on [legacy, pre - 1988]*, 33(9):934–944, Sep 1985.
60. The Network Simulator NS-2. `http://www.isi.edu/nsnam/ns/`.
61. A. Orda. Routing with end-to-end qos guarantees in broadband networks. *Networking, IEEE/ACM Transactions on*, 7(3):365 –374, June 1999.
62. Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Trans. Netw.*, 1:344–357, June 1993.
63. Georgios S. Paschos, Ioannis Papapanagiotou, Stavros A. Kotsopoulos, and George K. Karagiannidis. A new mac protocol with pseudo-tdma behavior for supporting quality of service in 802.11 wireless lans. *EURASIP J. Wirel. Commun. Netw.*, 2006:76–84, mar 2006.
64. E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture, 2001.
65. M. Saad, A. Leon-Garcia, and Wei Yu. Optimal network rate allocation under end-to-end quality-of-service requirements. *Network and Service Management, IEEE Transactions on*, 4(3):40 –49, December 2007.
66. N. V. Sahinidis and M. Tawarmalani. *BARON 9.0.4: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2010.
67. SCIP solver, v. 1.2.0. `http://scip.zib.de/`.
68. Harish Shetiya and Vinod Sharma. Algorithms for routing and centralized scheduling to provide qos in ieee 802.16 mesh networks. In *Proceedings of the 1st ACM workshop on Wireless multimedia networking and performance modeling*, WMuNeP '05, pages 140–149, New York, NY, USA, 2005. ACM.
69. M. Shreedhar and G. Varghese. Efficient fair queuing using deficit round-robin. *Networking, IEEE/ACM Transactions on*, 4(3):375 –385, June 1996.
70. D. Stiliadis and A. Varma. Latency-rate servers: a general model for analysis of traffic scheduling algorithms. *Networking, IEEE/ACM Transactions on*, 6(5):611 –624, October 1998.
71. Yu Wang, Weizhao Wang, Xiang-Yang Li, and Wen-Zhan Song. Interference-aware joint routing and tdma link scheduling for static wireless networks. *Parallel and Distributed Systems, IEEE Transactions on*, 19(12):1709 –1726, December 2008.
72. L. A. Wolsey. *Integer programming*. Wiley-Interscience, New York, NY, USA, 1998.
73. Wen-Lin Yang. Optimal and heuristic algorithms for quality-of-service routing with multiple constraints. *Perform. Eval.*, 57:261–278, July 2004.

74. Hui Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83(10):1374 –1396, October 1995.
75. Jun Zou and Dongmei Zhao. Connection-based scheduling for supporting real-time traffic in wireless mesh networks. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1 –6, December 2008.