Deadline-Constrained Content Upload from Multihomed Devices: Formulations and Algorithms

Deadline-Constrained Content Upload from Multihomed Devices: Formulations and Algorithms

Ali Safari Khatouni[1*], Marco Ajmone Marsan[1,3], Marco Mellia[1], Reza Rejaie[2]

1 - Politecnico di Torino, Italy
2 - University of Oregon, USA
3- Institute Imdea Networks, Spain

Abstract

This work originates from the practical requirements of video surveillance in public transport systems, where security cameras store video onboard, and a central operator occasionally needs to access portions of the recordings. When this happens, the selected video portions must be uploaded within a given deadline, using (multiple) wireless interfaces, with different costs (which correspond to, e.g., tariffs). We study this video upload problem as a scheduling problem with deadline, where our goal is to choose which interfaces to use and when, so as to minimize the cost of the upload while meeting the given deadline. Our study gives rise to adaptive schedulers that require only a very coarse knowledge of the wireless interfaces bandwidth.

In this paper, we first assume an oracle has perfect knowledge about the available bandwidth of wireless interfaces at each time, and we formulate an optimization problem to minimize the upload cost within the given deadline. Second, we propose greedy oracle-based heuristics that perform very close to optimal, and that can provide a simple baseline for performance. Third, we formulate a stochastic optimization problem, assuming only the knowledge of the distribution of available bandwidth, and, fourth, we propose adaptive schedulers, that we simulate and also implement and test in a real testbed.

Simulation results demonstrate that our adaptive solutions can effectively leverage the fundamental trade-off between upload cost and completion time, despite unpredictable variations in the available bandwidth of wireless interfaces. Experiments with real mobile nodes provided by the MONROE platform confirm these findings.

Keywords: Adaptive scheduling; wireless network; multihomed devices; content upload; smart city

1. Introduction

Wireless technologies such as WiFi, 3G, 4G, and soon-to-come 5G, provide access capacities up to hundreds of Mb/s. Multihomed devices are commonly available, offering the chance to transmit over different technologies and networks at the same time. Yet, there are scenarios in which the amount of data being produced and consumed challenges the bandwidth offered by wireless networks.

In this paper, we look at one of those scenarios. Our interest is motivated and inspired by the real needs of public transport operators. Public transport vehicles (like buses or trains) are equipped with multiple Mobile BroadBand (MBB) [1] interfaces, several onboard security cameras record videos. Those must be uploaded to a security center where an operator occasionally requests to watch selected portions of the videos. In this scenario, continuous real-time video uploading is too expensive. Even if current MBB networks can offer capacities up to 100 Mb/s, the number of vehicles and videos, the limited data quota, the performance variability along the route, and the need to check only parts of the videos, call for ingenious upload strategies. Hence, videos are stored onboard, and, only when an alarm is triggered, the security operator on duty requests the specific portion of the video that must be uploaded before a specified short deadline. The deadline normally is of the order of a few minutes, depending on the urgency of the incident. For instance, pickpocketing events can wait till the vehicle returns to deposit. Instead, in case of health problems of passengers, the security officer needs to access the video within a short time, after obtaining all the required authorizations.

We model this problem as the scheduling of content upload from multihomed mobile devices, where the content must be delivered within a given deadline, while the cost must be minimized. The cost associated with each interface is defined according to the nature of the problem. For example cost can correspond to tariffs, energy consumption, data quota, or system load. In our problem definition, cost is related to the monetary cost of the data transmission on each technology.

Our problem differs from the classic problem of content upload using multihomed nodes [2, 3], where upload delay has typically to be minimized, i.e., throughput maximized. Also, no real time constraint exists in our case, thus making our problem different from video streaming, and somehow similar to a delay tolerant scenario, albeit the hard deadline for delivery

*Corresponding author
Email addresses: ali.safari@polito.it (Ali Safari Khatouni[1]), marco.ajmone@polito.it (Marco Ajmone Marsan[1,3]), marco.mellia@polito.it (Marco Mellia[1]), reza@cs.uoregon.edu (Reza Rejaie[2])

of the entire content (rather than individual packets) must be met [4].

We assume that the mobile node is equipped with several MBB interfaces, with different technologies, e.g., cheap but occasionally available WiFi, more ubiquitous, but more expensive, 3G, 4G, and soon-to-come 5G subscriptions, possibly offered by different operators. The system has to decide i) which interface(s) to use, ii) when to upload from such interface(s), and iii) at which rate to upload (if there is available bandwidth). Our goal is to minimize the total cost of the upload, while meeting the deadline. A greedy solution that immediately starts uploading from all interfaces minimizes the upload time, ignoring opportunities for cheap interfaces to become available in the future, thus increasing upload cost. A trade-off clearly exists between minimizing the total transmission cost or minimizing the upload completion time.

In this paper, we propose and analyze a family of adaptive schedulers that require only a very coarse knowledge of the available bandwidth on wireless interfaces. We extend our work in [5] by defining a more refined scheduler, and evaluating our solution using a larger and recently collected dataset to evaluate the dynamic algorithm proposed in [6]. In addition, We discuss how to carefully tune the dynamic algorithm parameters, and we provide a more extensive evaluation. Finally, we implement and test the algorithms in a real testbed, provided by the MONROE[1] platform [7].

The main contributions of this paper are:

• Devising mathematical formulations of the deadline constrained content upload problem from multihomed terminals, under different assumptions.

• Reporting extensive evaluations of the proposed solutions, based on trace-driven simulations using recently collected traces.

• Designing, implementing, testing, and evaluating a real implementation of the proposed dynamic algorithm on deployed mobile multihomed nodes.

The rest of this paper is structured as follows. We first overview related works and we position our work with respect to state-of-the-art solutions for similar problems (Sec. 2). Then, we report on the collection of traces of content upload data rates from mobile multihomed terminals, showing the unpredictability of short-term variations in available bandwidth (Sec. 3) to gain insight about the trade-off between cost and delivery time over wireless channels. Next, we formulate and solve an idealized version of the problem, where an oracle has perfect knowledge of the upload rate on each interface at each time. The oracle can then schedule the upload in those time slots when cheap connectivity is (expected to be) available, thus minimizing total cost (Sec. 4.1). We also introduce three simple greedy heuristics, to show the effectiveness of intuitive approaches to solve this problem (Sec. 4.2). Then, we formulate the video upload problem as a centralized scheduling problem, where the upload rates of the available interfaces are random variables with known distribution. Solving such problem is computationally impractical (Sec. 4.3). Thus, we aim for a practical solution that requires only a coarse knowledge of the available bandwidth, and we design online, adaptive schedulers to explore the trade-off between cost and delivery time (Sec. 4.4). Afterwards, we test the proposed algorithms in the real MBB platform provided by the MONROE H2020 project (Sec. 6). Finally, we conclude the paper (Sec. 7).

[1] https://www.monroe-project.eu/

2. Related work

Mobile devices allow users to connect to multiple wireless networks with possibly different technologies [8, 9, 10], obtaining throughput values which depend on the terminal position, the network coverage, the traffic load, the weather conditions, etc. This makes the problem of scheduling transmissions over multihomed [11] wireless interfaces both relevant and challenging. Several authors already published works which are close to what we discuss in this paper. For example, Higgins et al. [12] were among the first to face this problem. They tackle a problem in this domain, and propose Intentional Networking, that lets the application choose opportunistically the interfaces, based on a label that expresses the application requirements. They however target different problems such as real-time communications, with no support for deadline.

We discuss related work by looking at three main dimensions, which correspond to topics which have been widely investigated in the past: predictability of wireless network performance, multipath TCP, and delay tolerant networks.

• **Predictability of MBB performance:** The performance of wireless network services under uncertain network availability has been previously investigated by several authors. Deng et al. [13] investigate the characterization of multihomed systems considering WiFi vs. LTE in a controlled experiment. They show that LTE can provide better performance than WiFi, also exhibiting large variability on both short and long time scales. Rahmati et al. [14] present a technique for estimating and learning the WiFi network conditions from a fixed node. Rathnayake et al. [15] demonstrate how a prediction engine can be capable of forecasting future network and bandwidth availability, and propose a utility-based scheduling algorithm which uses the predicted throughput to schedule the data transfer over multiple interfaces from fixed nodes. These works heavily rely on channel performance predictions, and consider scheduling at the packet-level, i.e., choosing which packet to send through which interface, to maximize the total throughput.

Our work differs from those, since we deal with moving vehicles, and this exacerbates the unpredictability of the network performance, as shown by several authors. For instance, Riiser et al. [16] collected 3G mobile network traces from terminals onboard public transport vehicles around the city of Oslo (Norway). Similarly, Chen et al. [17] measured the throughput of both single-path and multi-path data transport in 3G, 4G, and WiFi networks. In both cases, variability is much higher than for fixed nodes. Lee et al. [18] showed that mobile data offloading through WiFi can reduce the energy consumption of

the mobile device, Bychkovsky et al. [19] presented the connectivity characteristic of WiFi for a mobile device traveling in a city. Similarly, Safari et al. [20] ran large scale download measurements in MBB networks. They show MBB networks are much more complex than wired networks, because of many factors which clutter the picture.

Given this difficulty in predicting the characteristics of MBB networks, we collected traces, and we used them to run trace-driven evaluations in realistic scenarios.

• **Multi-path TCP:** A number of recent works focus on multi-path TCP (MPTCP) [2, 21], and look at the design of packet schedulers and congestion control algorithms. The goal of the authors normally is to maximize throughput, or equivalently to minimize upload time (rather than to minimize the total cost of uploading a given content within a specified deadline, as we do). Nikravesh et al. [3] thoroughly investigate the performance gains and the costs of mobile MPTCP by means of traffic measurements, and present the MPFLEX software architecture. Wu et al. [22] propose a framework for video streaming, but do not consider the cost associated with interfaces or a deadline for video upload, see also [23, 24, 25]. Lim et al. [26, 27] introduce an energy-aware variant of MPTCP which aims to reduce energy consumption with respect to standard MPTCP. They however consider download transfers, do not consider any deadline, which makes their work different from ours. Han et al. [28] show the fact that MPTCP implies unnecessary use of cellular interfaces in case of available bandwidth on WiFi in multihomed system. They present MP-DASH, a network interface preference-aware multi-path framework for DASH video streaming. Its performance is highly dependent on the choice of the DASH algorithm. This work is different from ours because it focuses on real-time streaming solution.

• **Delay tolerant networks:** Delay Tolerant Network (DTN) solutions for content upload try to find the way to deliver the content by maximizing the device-to-device transmission [29, 30, 31] or maximizing the use of WiFi [32, 33]. The data delivery has no deadline, and the main problem is the creation of the time-varying network topology to guarantee the delivery. Yetim et al. [32] illustrate the benefit of the delay tolerant approach to save energy by sending more data over WiFi interface. However, their approach does not support change in cost and deadline. On the contrary, we rely on MBB to offer connectivity with associated cost, and devise approaches to use interfaces so as to minimize cost while delivering the content before the deadline.

• **Deadline scheduling:** Previous works did consider scheduling under a fixed deadline, but they assumed that network performance is perfectly known. Zaharia et al. [34] presented an optimal scheduler over multiple network interfaces, and proposed approximations which can be implemented with limited resources in mobile phones, or PDAs. They assumed the cost and bandwidth of each interface to be constant. Moo-Ryong et al. [35] also proposed an algorithm for video upload from smartphones with two MBB interfaces. They focus on energy-delay trade-off. These works differ from ours, since they assume MBB interface availability and capacity are known (which we consider not realistic). We do not assume any a priori knowledge of available capacity.

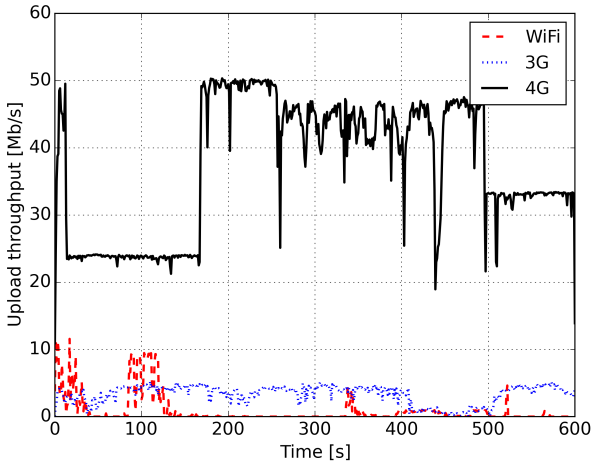


Figure 1: A sample of collected traces for each technology

3. Characterization of mobile traces

We first present some recent mobile traces that we collected from vehicles, with the dual purpose to show how unpredictable the available bandwidth is, and to run realistic performance evaluation using trace-driven simulations; indeed, a credible evaluation of the proposed algorithms calls for realistic data about available upload bandwidth from public transport vehicles. Previous studies collected traces of MBB network data rates, e.g., Chen et al. [17] and Riiser et al. [16]. However, these traces are not very recent, hence they exhibit lower bandwidth values than the ones that we commonly experience over today's wireless networks. Lutu et al. [36] collected a large set of recent traces for MBB from public transport vehicles. However, they primarily focus on the transfer of relatively short files (4 MB) that is less likely to effectively utilize the available bandwidth, and to represent its variations over a longer time scale (minutes vs. seconds). We thus resolved to collect our own traces by using mobile terminals onboard private and public vehicles, or carried by walking users.

3.1. Trace collection methodology

All traces were collected in the city of Torino in Italy in 2016, and refer to three technologies (WiFi, 3G, and 4G), and different mobile network operators. During trace collection, the MBB networks were in normal operating conditions (and unaware of our tests). Our terminals (both Android and iOS smartphones) accessed the mobile networks to upload data to a server on campus. We used both TCP and UDP.

We used a hybrid method in the trace collection process: during each experiment, the mobile terminal runs `iperf2` [2] in the

[2] https://iperf.fr/iperf-doc.php

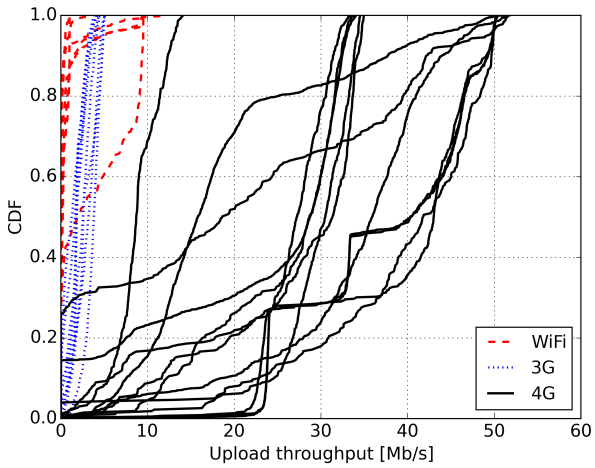


Figure 2: CDF of throughput for multiple traces and technologies

upload direction for 600 seconds while `tcpdump` [3] captures packets at the server. Using the packet trace, we compute the throughput in each second of the experiment. The number of repetitions of active measurements is critical to make sure that enough samples are collected for a sound estimation of the distribution of the throughput of each technology. It is important to note that repetitions cover different times of the day and different days of the week. We repeated the experiment on the same driving routes for at least 5 times, during different days.

In total, we collected 40 traces for each of the three different mobile network operators in Italy (namely TIM, Wind, and Vodafone), with the objective of obtaining multiple samples of the upload throughput in MBB networks. Traces are collected in mobile scenario, the speed of vehicles were in range of [0,70]km/h, most of them collected in three routes with length [2,5]km. For WiFi, we considered the open WiFi community "WoW-Fi" offered by Fastweb customers that share their ADSL or FTTH home networks via the access gateway.[4] We make the collected traces available for researchers[5].

3.2. Trace characterization

We now present evidences of unpredictability of throughput, which makes the scheduling of content upload a challenging task. Fig. 1 shows a sample of the temporal evolution (x-axis) of the upload rate (y-axis) for WiFi (red), 3G (blue), and 4G (black) interfaces, collected when using UDP and TCP. For each trace, we compute the upload throughput considering time intervals of 1 second. Starting times of traces have been realigned for ease of visualization. Fig. 2 presents the Cumulative Distribution Functions (CDF) of the per second upload rate for multiple randomly selected traces. Figures 1 and 2 indicate that WiFi offers upload throughputs which are very variable in time, with a behavior that is almost ON-OFF. Upload rates are limited to less than 10 Mb/s. This is due to the limited coverage of the WiFi network, and to the upload bottleneck of ADSL or FTTH access technologies. The 3G technology provides upload rate values which are invariably lower than 5 Mb/s, with significant short-term variability, but, thanks to the extensive coverage, no long periods of close-to-zero available bandwidth were observed. The behavior using the 4G interfaces exhibits even higher variability, with rates one order of magnitude higher than 3G (up to 50 Mb/s). We do not observe any significant differences when using TCP or UDP, since the unpredictable changes in the rate are mostly due to sudden changes in the access link than to congestion along the path.

Table 1: Throughput statistics

interface	mean	standard deviation	80-th percentile	max	min
WiFi	0.77 Mb/s	2.06	0.57	11.56	0
3G	2.23 Mb/s	1.29	3.44	5.23	0
4G	26.92 Mb/s	13.50	39.47	51.74	0

Table 2: $|Th_t - Th_{t-1}|$ statistics

interface	mean	standard deviation	80-th percentile	max	min
WiFi	0.30 Mb/s	0.89	0.24	9.45	0
3G	0.39 Mb/s	0.44	0.63	3.84	0
4G	2.02 Mb/s	2.98	2.93	47.14	0

Table 1 shows, as a summary of the statistics for the three technologies, the average, standard deviation, 80-th percentile, maximum, and minimum of the observed per second upload rate. Table 2 reports the same statistics, but considering the absolute difference of throughput in two consecutive time slots. In a nutshell, measurements indicate that it is not realistic to assume the exact value of the future available bandwidth, as also claimed by Nikravesh et al. [37].

4. Problem formulations and algorithms

Although we just claimed that assuming to know the future available bandwidth is not realistic, we start by considering this case, since it provides a baseline for performance evaluation. As a second step, we present a stochastic formulation of the problem, that assume only the probability distribution of the available bandwidth is known. Finally, we propose dynamic schedulers that adapt their choices based only on the past values of available bandwidth.

4.1. Optimal solution with perfect bandwidth knowledge

We assume an oracle has perfect knowledge about the bandwidth of all interfaces at all times. We consider time slots, with slots of duration ΔT. The time slot duration is such that the available bandwidth over all interfaces can be assumed constant for one time slot.

We model the scheduling problem using a directed graph $G_1 = (N, E)$, where N is the set of nodes and $E = \{(i, j) \mid i, j \in N\}$ is the set of edges. Referring to Fig. 3, the leftmost node in G_1 represents the video source, i.e., the vehicle. The second group of nodes represents the video files to be uploaded. Each video $k = 1, \ldots, K$ (2 videos in the example) is of volume V_k, and can be uploaded through different interfaces. The system

[3] http://www.tcpdump.org/

[4] `http://www.fastweb.it/adsl-fibra-ottica/dettagli/wow-fi/`. Mobile phones automatically authenticate using IEEE 802.1x with no action needed from the user.

[5] `http://tstat.polito.it/traces-MBB-speedtest.shtml`

has T time slots to complete the upload, represented by the third group of nodes. Each node in this group represents a given interface and time slot. For ease of visualization, nodes referring to the same interface (2 interfaces in the example) are grouped in a box. The number of available time slots (5 in the example) represents the deadline to be met. The rightmost node represents the sink, i.e., the server receiving the videos.

Edges in E are labeled by a cost $c_{i,j}$ and a bandwidth $r_{i,j}$. The label of edge (i, j) is denoted $(c_{i,j}, r_{i,j})$. The source node is connected to each video node. Edges exiting from the source node have zero cost, and bandwidth equal to the video file size in bits. Edges from video nodes to time slot and interface nodes are characterized by the cost per bit of using such time slot and interface ($c_{i,j}$), and the maximum flow that can be supported by such time slot and interface ($r_{i,j}$) in bits/s. This model allows videos to have different deadlines. Indeed, each video is connected only to the slots it can use. nodes representing time slots and interfaces are connected to the sink with an edge with zero cost, and bandwidth equal to the time slot bandwidth.

We assume that there is enough bandwidth to successfully upload all videos, and that any interface can be shared between any video at any time slot, i.e., video content is fluid, and can be split arbitrarily.

4.1.1. Minimum cost flow problem model

We model this problem as a Minimum Cost Flow Problem (MCFP) [38], in which we look for the maximum flow that the network can carry, with the minimum total cost. The objective function in (1) presents the total upload cost, which must be minimized. Expression (2) forces flow conservation constraints. It states that the sum of incoming flows at all nodes (except source and sink) is equal to the sum of outgoing flows, i.e., flow cannot be created or disappear at intermediate nodes. The flow on every edge is non-negative, and it cannot exceed the rate $r_{i,j}$, see (3). Expression (4) forces the total flow exiting from the source node to be greater or equal to the sum of all requested videos, i.e., all videos must leave the source.

Table 3: Variables definition for MCFP

variable	definition
t	Time slot index $\in [0 \dots T]$
ΔT	Duration of time slot
T	Number of time slots before the deadline
K	Number of videos
I	Number of interfaces
N	Total number of nodes in the graph
$r_{i,j}$	Available bandwidth on edge from node i to node j
$f_{i,j}$	Amount of data scheduled from node i to node j
$c_{i,j}$	Cost associated to edge from node i to node j

$$\textbf{min} \sum_{(i,j)\in E} c_{i,j} f_{i,j} \Delta T \tag{1}$$

$$\sum_{(i,j)\in E} f_{i,j} = \sum_{(l,i)\in E} f_{l,i} \;\; \forall i \in Nandi, j, l \neq Source, Sink \tag{2}$$

$$0 \leq f_{i,j} \leq r_{i,j} \;\; \forall (i,j) \in E \tag{3}$$

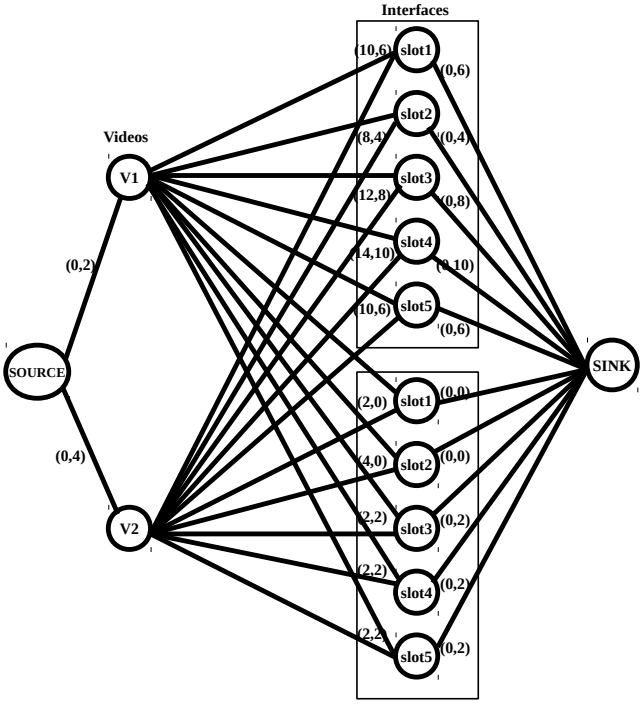


Figure 3: An example to represent the MCFP model with 2 videos, deadline equal to 5 time slots, and 2 interfaces

$$\sum_{(Source,j)\in E} f_{Source,j} \geq \sum_{i} V_i \tag{4}$$

The MCFP problem can be solved using well known and efficient approaches, like the one in [39], with complexity of $O(m \log n(m + n \log n))$ on networks with n nodes and m arcs. As depicted from graph G_1 n and m are equal to $n = K + (I * T) + 2$ and $m = (K + 1)(I * T) + K$. In this work, we use the CPLEX [40] solver.

4.2. Heuristic approaches with full knowledge

In order to have simpler alternatives for the computation of the (quasi) optimal solution in the case of perfect bandwidth knowledge, we consider three simple and intuitive greedy heuristics:

i) Greedy-in-time (GT) - This algorithm uploads all videos through all interfaces as soon as possible. It minimizes the upload time, greedily uploading as much data as possible through all available interfaces. The video with closest deadline is transmitted first, as soon as any interface has an available slot to upload (part of) the video.

ii) Greedy-in-rate (GR) - This algorithm sorts time slots according to decreasing transmission rate, and schedules transmission through the highest-rate time slots. If rates are equal, earlier time slots are preferred.

iii) Greedy-in-cost (GC) - This algorithm sorts time slots according to increasing cost, and schedules transmission

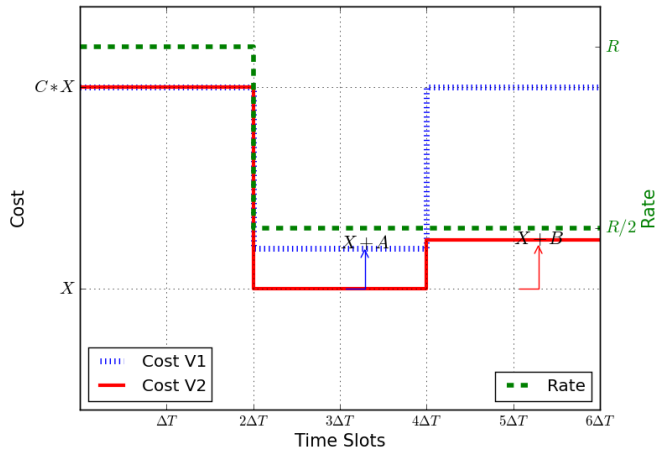


Figure 4: Example to show how greedy approaches can perform worse than the optimal solution

through the cheapest time slots. If costs are equal, earlier time slots are preferred.

All heuristics stop when expression (4) is met, i.e., all videos are uploaded. The first greedy algorithm guarantees that the transfer is completed as soon as possible (this makes it similar to MPTCP solutions), while the second one minimizes the number of time slots to use. Both approaches disregard the upload cost. Only the third algorithm explicitly considers the cost of using different interfaces at different times.

Assume T is the number of time slots, I is the number of interfaces, and K is the number of videos. The GT algorithm only needs the temporal ordering of slots, which is given, so that complexity is $O(1)$. The GC algorithm needs the ordering of time slots according to cost, so that complexity is $O(T*I*K)$ (which depends on the video and the interface), and then according to time. The GR algorithm needs the ordering according to slot bandwidth, and time, so that complexity is $O(T*I)$.

To show that greedy approaches can produce a solution which has suboptimal cost, we use a very simple example with only 1 available interface, and 2 videos to be uploaded, of sizes $V_1 = V_2 = R\Delta T$. The slot costs are presented in Fig. 4, with blue dotted and red solid lines for V_1 and V_2, respectively. The costs of V_1 and V_2 are different (e.g., to represent priorities), and change over time (e.g., due to different tariffs at different time of day). This can describe a scenario with congestion-based pricing. Available bandwidth varies according to the green dashed line. The two videos have the same deadline equal to $6\Delta T$. Each video can be uploaded in one time slot with rate equal to R bits/s, or 2 time slots with rate $R/2$. We can easily compute the total cost, considering the greedy heuristics and the optimal solution. By assuming that $A < B$ and $C > 2$, we have:

1. *GT* - The two videos are uploaded in the first two time slots (either one slot per video, or sharing the slot bandwidth), without considering the cost of slots. The total cost is $2CXR\Delta T$.
2. *GR* - The two videos are uploaded in the first two time slots, which have highest rate, without considering the cost of slots. The total cost is $2CXR\Delta T$.
3. *GC* - Since the upload of V_2 has the lowest cost in the third and fourth time slots, which have rate $R/2$, the upload of V_2 is scheduled in those slots. The cost for the upload of V_1 is equal to CX in all slots except the ones that are allocated to the upload of V_2. The first slot is chosen because of the high rate. The total cost is $CXR\Delta T + 2XR\Delta T/2 = (C+1)XR\Delta T$.
4. *Optimal solution* - The solution based on MCFP schedules the upload of V_1 in the third and fourth time slots, and the upload of V_2 in the fifth and sixth time slots. The total cost is $(X+A)R\Delta T + (X+B)R\Delta T = (2X+A+B)R\Delta T$.

We conclude that the total cost for GT and GR is C times higher with respect to the optimal solution. Instead, the cost for GC is $(C+1)/2$ times higher with respect to the optimal solution where $(A+B) \to 0$. By means of this example we have shown that greedy algorithms can generate solutions with possibly much higher cost than the optimal solution, even in very simple cases.

4.3. Multistage stochastic model

Since assuming the perfect knowledge of the available bandwidth on all interfaces in all time slots is not realistic, we next look at a case with reduced information. We consider the available bandwidth of interface i in slot t, $R_{i,t}$ as a random variable, denoting the realization of $R_{i,t}$ with $r_{i,t}$. From frequent large-scale measurements, it can be possible to estimate the probability distribution of $R_{i,t}$, although this requires great effort. Naturally, the values $r_{i,t}$ are known for past slots, as data are transmitted over the interfaces. Uncertainty in data can be modeled through multistage stochastic models, as follows:

Given the distribution of $R_{i,t}$, we model the scheduling problem using a series of directed graphs $G_2(t) = (N, E)$, where N is the set of nodes and $E = \{(i,j) \mid i,j \in N\}$ is the set of edges. As before, we assume time is slotted, with slot duration ΔT. K videos, each of volume V_k, $k = 1, \cdots, K$, have to be uploaded through I interfaces. The number T of available slots represents the deadline. Edge (i,j) in E is labeled by two values: a cost, and a bandwidth, denoted $c_{ij,t}$ and $r_{ij,t}$ at time t, respectively. Fig. 5 illustrates the graph $G_2(t)$ at time $t = 0$. A source node is connected to the K nodes representing videos. Edges exiting from the source node have zero cost, and bandwidth equal to the video file size in bits. Each video node is then connected by a directed edge to I nodes, each representing an interface. These edges are labeled by the cost per bit of using an interface ($c_{ij,t}$) at time t, and the maximum bandwidth that can be supported at time slot t ($r_{ij,t}$) in bit/s.

Last, interface nodes are connected to a sink node, with an edge with zero cost, and bandwidth equal to infinity. Only after making a decision for the data to send over an interface at time t, the actual bandwidth $r_{ij,t}$ is known. The quantity $b_{i,t}$ represents the amount data in buffer at node i because the bandwidth was lower than expected at previous time slots. Therefore, the

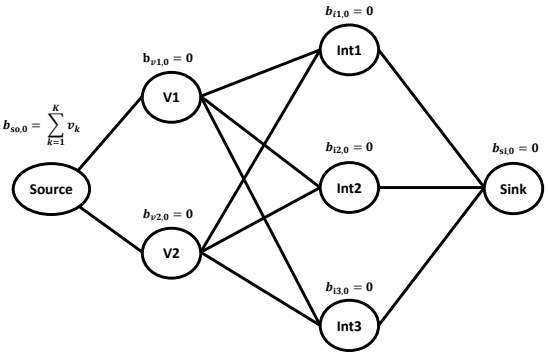


Figure 5: Stochastic model representation at time $t = 0$

buffer occupancy evolves over time based on decisions made in previous time slots.

Table 4: Variables definition for MSMCFP model

variable	definition
t	Time slot index $\in [0 \dots T]$
T	Number of time slots
K	Number of videos
I	Number of interfaces
$b_{i,t}$	Amount of data in buffer at node i at time t
$r_{ij,t}$	Available rate from node i to node j at time t
$f_{ij,t}$	Scheduled data from node i to node j at time t
$x_{ij,t}$	Transmitted data from node i to node j at time t

The quantity $x_{ij,t}$ represents the amount of successfully transmitted data at time slot t on edge ij. Table 4 summarizes the variables used to formulate the problem. Using the graph $G_2(t)$, the problem of minimizing the total cost C to deliver all videos can be solved as a Multistage Minimum Cost Flow Problem (MSMCFP), in which we look for the maximum flow that the network can carry, with the minimum total cost.

The objective function in (5) presents the expected value of the total cost over the deadline, which must be minimized, while (6) forces the real flow passing from node i to node j to be the minimum between what is scheduled $f_{ij,t}$ and the actual bandwidth of the interface $r_{ij,t}$ at time t. Expression (7) states that flow cannot appear/disappear at intermediate nodes at any time, the data being either transmitted or stored in buffers. The flow on every edge must be non-negative, and it cannot exceed rate $r_{ij,t}$, as dictated by (8). Expressions (9) and (10) force the total flow exiting from the source node and entering in the sink node to be equal to the sum of all video sizes. Expressions (11), (12), and (13) indicate the state of buffers at any time t.

$$\min C = \sum_{t=0}^{T} \mathbb{E}\Big\{ \sum_{(ij)\in E} x_{ij,t} c_{ij,t} \Delta T \Big\} \tag{5}$$

$$x_{ij,t} = \min(f_{ij,t}, r_{ij,t}) \;\; \forall ij, t \tag{6}$$

$$b_{i,t+1} = \max\left(b_{i,t} + \sum_{j:(j,i)\in E} x_{ji,t} - \sum_{j:(i,j)\in E} x_{ij,t}, 0 \right) \forall ij, t \tag{7}$$

$$0 \le f_{ij,t} \le r_{ij,t} \qquad \forall (ij) \in E,\ t \in \{0 \dots T\} \tag{8}$$

$$\sum_{t=0}^{T} \sum_{j:(Source,j)\in E} x_{Source,j,t} = \sum_{i=1}^{K} V_i \tag{9}$$

$$\sum_{t=0}^{T} \sum_{i:(i,Sink)\in E} x_{i,Sink,t} = \sum_{i=1}^{K} V_i \tag{10}$$

$$b_{i,T} = 0 \; \forall i, i \neq Sink \; and \; b_{Sink,T} = \sum_{i=1}^{K} V_i \tag{11}$$

$$b_{i,0} = 0 \; \forall i, i \neq Source \; and \; b_{Source,T} = \sum_{i=1}^{K} V_i \tag{12}$$

$$b_{i,t} \ge 0 \; \forall i, t \tag{13}$$

To solve the MSMCFP problem, we need to obtain accurate estimates of the probability distribution for the bandwidth of each interface. This requires, as we already noted in Sec. 3, large-scale measurements at different times in all possible places, which is unrealistic. In addition, no off-the-shelf solver is available to solve multistage stochastic problems; the problem can be solved only by searching through all possible scenarios, which requires the exhaustive exploration of a tree of realizations of depth T, with nodes of degree I, i.e., a complexity $O(I^T)$. This means that this more realistic option, which only assumes the availability of probabilistic information about the bandwidth available on interfaces, is not viable because of the solution complexity. This means that the only feasible option to solve the video upload problem is to design adaptive heuristics.

4.4. Dynamic heuristic

Given the complexity of solving the MSMCFP problem, we designed an adaptive algorithm that is inspired by schedulers for P2P video streaming proposed by Magharei et al. [41]. The dynamic of variations for individual connection as well as the design goals in our problem are however different from those in [41]. We only assume the knowledge of the long-term average throughput of each interface. This information serves as a reference to assess the feasibility and the pace of progress for meeting the specified deadline.

We consider slotted time, where ΔT denotes the duration of a single slot. At the beginning of each slot, the scheduler computes the amount of data to transmit on each interface using the observed throughput in recent past slots. It updates the expected rate on each interface based on the overall pace of upload progress during the recent slots, and schedules the transmission of a portion of the data, giving preference to cheaper interfaces. During the slot, data is transmitted according to the actual network state. At the end of the slot, the scheduler checks whether the amount of transmitted data is smaller than expected. If this happens, the unsent data, denoted by $LeftB$, is greater than zero. $LeftB$ is the sum of the amounts of data remaining in all the interfaces' buffers at the end of the time slot. When $LeftB < 0$, the system is behind the expected schedule, and the scheduler needs to recover in the future. We consider two

policies for recovery: i) aggressively recovering during the next slot, ii) conservatively (i.e., optimistically) recovering across all the remaining slots before the deadline. Let t be the current time slot. B_t represents the expected data rate at which the system should transmit during slot t. At the upload start, we estimate $B_0 = V/T$, being $V = \sum_{k=1}^{K} v_k$ the total data volume size, and T the deadline. At the end of each time slot, the system computes $LeftB$, the total amount of scheduled data that was not possible to transmit due to a lack of bandwidth in that slot.

To help the scheduler, each interface i maintains the expected rate $\widehat{r}_{ti}$ based on the actual transmission bandwidth r_{ti}. If the interface i was active in period t ($r_{ti} > 0$) and congested ($LeftB_i > 0$), $\widehat{r}_{ti}$ is updated using an Exponentially Weighted Moving Average (EWMA) algorithm with α coefficient:

$$\widehat{r}_{t+1i} = \begin{cases} \alpha\widehat{r}_{ti} + (1-\alpha)r_{ti} \text{ if } r_{ti} > 0 \; and \; LeftB_i > 0 \\ Max(\widehat{r}_{ti}, r_{ti}) \text{ if } r_{ti} > 0 \; and \; LeftB_i = 0 \\ \widehat{r}_{ti} \text{ otherwise} \end{cases} \quad (14)$$

The rationale of expression (14) is to avoid the estimated bandwidth to converge to small values when an interface is not being used, or used at a rate lower than the maximum available bandwidth, i.e., when the interface bandwidth is not fully utilized. Indeed, data is transmitted at the expected rate and the actual available bandwidth of the interface is unknown. Thus, we avoid decreasing the estimated rate of those interfaces that are not fully utilized. This happens because the algorithm is not greedy, and interfaces are partially used in a demand-driven fashion.

Algorithm 1 presents the pseudo code for adaptive scheduling. After initialization, the algorithm loops over time slots until the deadline is reached, or all the data have been uploaded (line 4). To minimize cost, interfaces are prioritized for data transmission from the least to the most expensive (line 5). At the beginning of each time slot t, the system has to schedule data for transmission (line 6). V_t represents the amount of video data to transmit at time t. $V_t = B_t\Delta T$ when considering the most expensive interface, otherwise, $V_t = (\beta + 1)B_t\Delta T$. $\beta \in \mathbb{R}^+$ is a parameter that controls the optimism of the scheduler. When $\beta = 0$, the scheduler tries to upload the content at the minimum rate which guarantees to complete the upload within the deadline. When $\beta > 0$, the system is more optimistic, and the scheduler tries to utilize any excess bandwidth, so as to deliver more data, and stay ahead of schedule. This increases the chance of completing the upload before the deadline, even if the available bandwidth drops below the expected value in the future. In other words, the parameter β allows pushing extra data into the interface buffer to efficiently use the excess bandwidth of cheap interfaces. By forcing $\beta = 0$ for the most expensive interface[6], we avoid any extra load on that interface, in order to minimize its use and the overall cost of upload (line 8). The amount of data scheduled on interface i at time t is the minimum between V_t and the estimated expected rate $\widehat{r}_{ti}\Delta T$ (line 13). line 14 computes any leftover of video still to schedule on more expensive interfaces.

[6]When costs change over time, the algorithm adapts β consequently.

Algorithm 1 Adaptive Scheduler

```
1:  procedure AdaptiveScheduler(α, β, policy)
2:      r̂_0i ← Interface average rates
3:      B_0 ← V/T # minimum rate to meet the deadline at time t=0
4:      for (t = 0; t < T && V > 0; t++) do
5:          SortInterfaceByCost()
6:          procedure Push data to buffers
7:              for (i = 1; i ≤ I && V_t > 0; i++) do
8:                  if cost(i) < maxcost then
9:                      V_t = (β + 1)B_tΔT
10:                 else
11:                     V_t = B_tΔT
12:                 end if
13:                 V_ti = min(V_t, r̂_tiΔT)
14:                 V_t = max(V_t − V_ti, 0)
15:             end for
16:         end procedure
17:         UploadAndWaitForSlotEnd()
18:         procedure Check data in buffers
19:             V = V − Σ_i^I r_tiΔT
20:             for (i = 1; i ≤ I; i++) do
21:                 LeftB_i = V_ti − r_tiΔT
22:                 if r_ti > 0 && LeftB_i > 0 then
23:                     r̂_t+1i ← αr̂_ti + (1 − α)r_ti
24:                 else if r_ti > 0 && LeftB_i = 0 then
25:                     r̂_t+1i ← Max(r̂_ti, r_ti)
26:                 else
27:                     r̂_t+1i ← r̂_ti
28:                 end if
29:             end for
30:         end procedure
31:         LeftB = Σ_i^I LeftB_i
32:         if LeftB > 0 then
33:             if policy == Aggressive then
34:                 # update minimum rate to meet the deadline at time t=t+1
35:                 B_t+1 = B_0 + LeftB
36:             else
37:                 B_t+1 = B_t + LeftB/(T − t)
38:             end if
39:         end if
40:     end for
41: end procedure
```

The data in the buffer is transmitted over all interfaces (line 17). At the end of each slot, the algorithm updates the amount of remaining data that must still be transmitted (line 19), and updates the transmission rate (line 21), and the expected rate (line 22-27).

If the aggregate transmission rate is smaller than expected, data is accumulated in $LeftB$. When this happens, the system has to recover by increasing the amount of data to schedule for transmission B_{t+1}. If the *aggressive* recovery policy is selected, the scheduler tries to recover in the immediately following slot

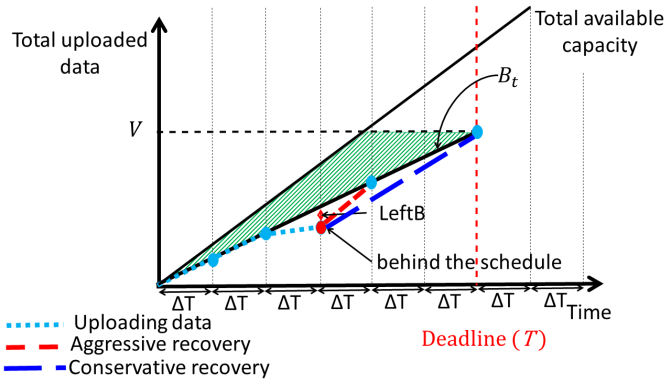


Figure 6: Scheduling process over time and recovery strategies

(line 35). If instead the *conservative* recovery policy is selected, the scheduler spreads the left-over data over all the remaining slots until the deadline, which leads to a higher average transmission rate in remaining slots (line 37).

Fig. 6 illustrates the evolution of the upload process over time. At each ΔT, the algorithm schedules the amount of data to be uploaded B_t. At the end of the third time slot, the actual amount of transmitted data is lower than expected, due to a drop in bandwidth. The system reacts by updating the expected rate for future slots, and trying to either recover in the immediately upcoming slot (aggressive policy, red line), or in the remaining slots before the deadline (conservative policy, blue line).

5. Trace-driven simulations

5.1. Simulation setup

We first describe the simulation setup that we used to run experiments, aiming at the performance evaluation of different schedulers, at the identification of suitable parameter values, and at the exploration of trade-offs between performance and complexity. We consider a scenario where 2 videos must be uploaded from a vehicle equipped with one node with 3 different network interfaces, each one using a different technology: 3G, 4G, and WiFi. As we already mentioned, we base our experiments on the traces presented in Sec. 3. Since public transport vehicles repeatedly follow a fixed path, we loop the traces as many times as necessary to reach the deadline. To allow for some randomness (inherent in wireless bandwidth availability, due to varying network conditions), we select a random combination of traces for each simulation run, and we choose a random starting point for each trace.

The video upload deadline T is chosen in the order of a few minutes. The time slot duration is the only parameter for which domain knowledge can offer a compelling choice: ΔT must be coherent with the time scale of changes in bandwidth at the different interfaces. Using large values for ΔT decreases the ability of the scheduler to adapt to bandwidth changes in a timely manner, whereas having very small values results in unnecessary oscillations and lower bandwidth utilization. The traces collected for WiFi, 3G, and 4G show bandwidth changes on a scale of seconds. For this reason, we choose for ΔT a value equal to 1 second. [7]

The cost associated with each interface is an input to the scheduler, and can be chosen according to the end user constraints and the specific context of the application; it can be derived from either tariffs, or energy consumption, or data quota, or system load, or a combination of those. In our context, the node is onboard a bus, and always plugged; therefore, we assume that energy consumption is not crucial. Considering that the proposed adaptive scheduler is a lightweight application, we do not consider system load to define the cost. We thus use the monetary cost associated to the cost per bit on each interface (hence with each technology). Considering the end user fees for data transmission over different technologies (and including flat as well as variable fees). Arbitrarily, we assume the cost assigned with each interface to be 2, 4, and 8 $(Mb)^{-1}$, respectively, for WiFi, 3G, and 4G. Note that we assume that costs are the same for all videos, but the scheduler can cope with costs that are different for each video. This feature can be exploited, together with the selection of different deadlines, when videos have different urgency. In our previous work [5] we also considered a larger scenario with 10 interfaces and 5 videos, but we omit it here for the sake of brevity.

Videos have size equal to 62.5 MB (V = 125 MB in total), corresponding to about 5 minutes of 1080p video. The simulation time (which corresponds to the deadline T) varies in the range of [100,1000] seconds. We repeat each experiment 100 times with a random combination of traces as input, and we measure the average and the confidence interval for the following two metrics: i) time to complete the upload; and ii) cost of the upload.

5.2. Perfect knowledge centralized scheduler results

We start by considering the case of perfect knowledge of available bandwidth in future slots, and by comparing the performance of the MCFP to the one of the greedy heuristics. We use the *IBM ILOG CPLEX Optimization Studio 12.6.0.0* [40] Solver Engine to find the optimal solution of the MCFP formulation, while the greedy heuristics are implemented in Python. Experiments were run on the high performance computing cluster *hpc@polito*.[8] Intuition suggests that, for growing values of the deadline, performance should improve, since schedulers have more opportunities to trade cost against delay. In addition, we can expect that schedulers force the upload completion times to be close to the deadline, waiting for any cheap slot appearing toward the end of the available time interval (given the perfect knowledge of available bandwidth in future slots).

Fig. 7 reports results for the average upload cost (together with confidence intervals) versus the upload deadline (expressed in number of time slots) for the greedy heuristics and the optimal solution. As expected, the GR algorithm incurs the highest upload cost, followed by GT. Both algorithms are insensitive to the deadline value, since they do not consider cost. On

[7] We explored ΔT values in {1, 2, 5, 10} s. Results are omitted for the sake of brevity.

[8] http://www.hpc.polito.it

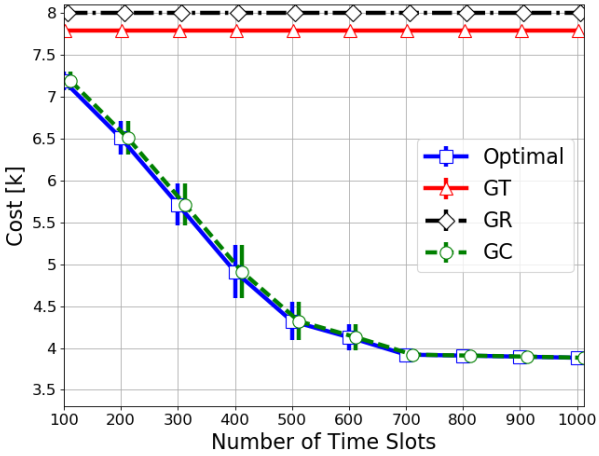


Figure 7: Cost for perfect knowledge centralized results

the contrary, the GC algorithm and the optimal solution provide cost values which decrease for growing deadline, thus meeting our expectation. Quite interesting is the fact that the GC algorithm provides results that are marginally higher than those of the optimal solution. We also analyzed the average total upload time for all algorithms, finding that the optimal solution produces upload times very close to the deadline. The same happens for the GC and GR algorithms, that also leverage the possibility to choose on multiple slots when the deadline is large. On the contrary, the GT algorithm yields very short upload completion times, as expected.

The two main conclusions that we can draw from this first set of results are: i) it is possible to reduce cost by effectively utilizing available bandwidth, despite substantial variations in available wireless bandwidth over time; ii) GR performs worse than GT, since the latter provides lower cost and lower completion times; iii) GC achieves practically the same cost and completion times as the optimal solution.

5.3. Dynamic heuristic scheduler results

We now consider the realistic case where no a priori knowledge of available bandwidth is available. For the scheduler performance analysis, we implemented a custom simulator using Python. The simulator models the upload of K ($K = 2$ in our results) videos from a mobile vehicle equipped with WiFi, 3G, and 4G interfaces (one each in our results). Traces are used to emulate the actual available bandwidth at any time slot.

5.3.1. Dynamic heuristic scheduler parameters

For starters, we investigate the scheduler parameter setting, to better understand the effect of parameter values on performance. The two scheduler parameters are α and β. The parameter α gets values in $[0, 1]$ and drives the EWMA estimation of available bandwidth in future slots. The parameter β controls the amount of extra data (in addition to the estimated available bandwidth) pushed in all interface buffers, except the most expensive one, as described in Sec. 4.4. β takes values in $[0, 30]$.

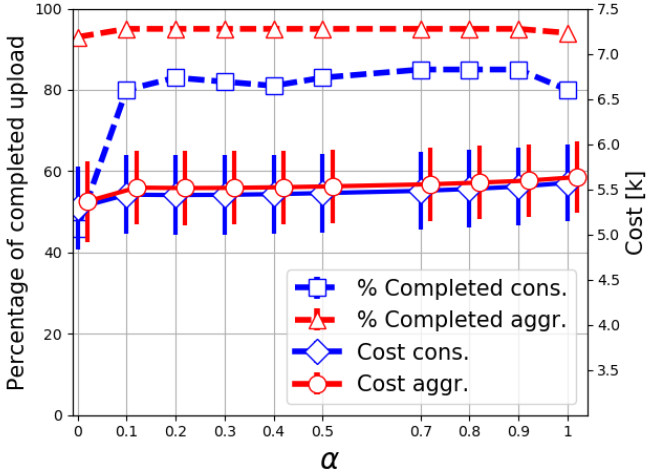


Figure 8: Percentage of completed uploads and cost versus α with $T = 300$ s and $\beta = 1$

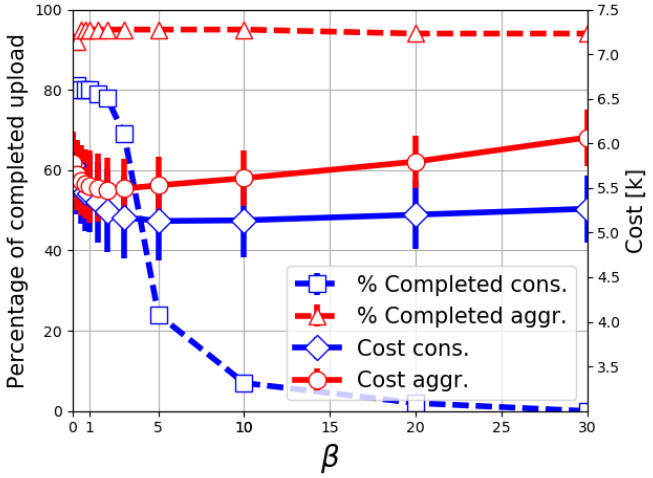
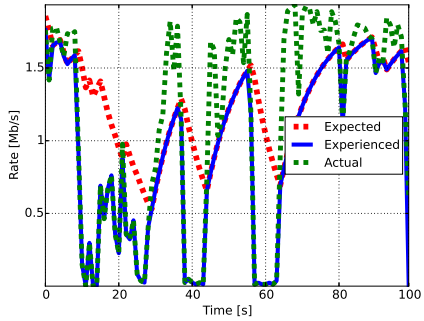


Figure 9: Percentage of completed uploads and cost versus β with $T = 300$ s and $\alpha = 0.1$. Trying to push more data than expected has positive benefits on the aggressive algorithm, but dramatic effects on the conservative algorithm

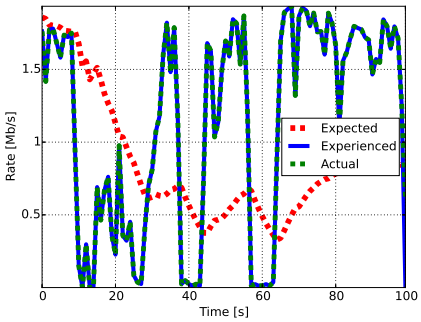
For some combinations of parameters, the upload of the video may not terminate within the specified deadline due to the improper scheduling strategy. We thus use these metrics to evaluate the performance of the scheduler: i) the total cost of the upload, ii) the probability of completing the video upload within the deadline, and iii) the amount of time after the deadline to complete the upload. Our objective is to achieve a very high completion probability with a very low cost.

5.3.2. Parameter setting

We first investigate the impact of α, which determines the timescale of the rate estimation for each interface. Fig. 8 presents the percentage of completed uploads (dotted lines - left y-axis) and their total cost (solid lines - right y-axis) as a function of α for different recovery approaches. Notice how performance is rather insensitive to the value of α. Recalling that α drives the EWMA estimation of the $\widehat{r}_{ti}$, we can conclude that it does not have considerable impact on the proposed adaptive scheduler. This holds true also for all considered β not reported



(a) 3G with $\beta = 0$



(b) 3G with $\beta = 5$

Figure 10: The aggressive scheduler operation with $\beta = 0$ (upper), $\beta = 5$ (lower), and $\alpha = 0.1$

here for the sake of brevity. That is, even a very coarse estimation of the link available bandwidth is sufficient to achieve our goals. In the following, we fix $\alpha = 0.1$.

The choice of β is less intuitive. To illustrate this, Fig. 9 shows the percentage of completed uploads (dotted lines - left y-axis) and their total cost (solid lines - right y-axis) as a function of β. We clearly see that β has a significant impact on performance. Fig. 9 indicates that the aggressive recovery algorithm (red curves) is less sensitive to β (lines are almost flat), which means that aggressively recovering is more important than optimistically spreading extra data across remaining time slots. The blue curves show that the conservative approach does not work properly with large values of β. The rate of delivery required to catch up may never become available, thus missing the deadline. Indeed, increasing β makes the system try to optimistically send more data through cheaper interfaces with the aim to reduce the overall upload cost. However, due to insufficient bandwidth across the remaining slots, the gap between the expected and the actual pace of progress of the upload ($LeftB$) keeps increasing. Therefore, the conservative recovery strategy is unable to catch up and meet the deadline. Notice how the chance to meet the deadline suddenly decreases for increasing β, with most uploads failing for $\beta > 5$. In the following, we fix $\beta = 1$.

To further illustrate the effect of β on the performance of the scheduling algorithm, Fig. 10 shows the evolution of the upload rate over the 3G interface versus time during an experiment, with $\beta = 0$ and $\beta = 5$, respectively. The green (dashed line) is the available bandwidth, the red (dashed line) is the EWMA of available rate, and the blue (solid line) is the experienced rate. The closer the blue curve is to the green curve, the more the system is able to exploit the available bandwidth. Fig. 10 clearly proves that the choice $\beta = 0$ lets a large fractions of the actual available bandwidth on 3G go unused. This forces the scheduler to use the expensive 4G interface. Setting $\beta > 0$ makes the system more optimistic, and prone to send more data than the current bandwidth estimation would allow. This increases the utilization of the 3G interface and hence reduces the load on the (expensive) 4G interface.

5.3.3. *Impact of the deadline*

We now look at the influence of the deadline on performance. Fig. 11 shows the percentage of completed uploads (top plot) and their total cost (bottom plot), for values of $T \in [100, 1000]\,s$, with $\beta = 1$, and $\alpha = 0.1$. As we can expect, longer deadlines imply higher percentages of completed uploads, and lower costs. Looking in more detail at the percentage of completed uploads, we see that the aggressive version of the algorithm (red curve) consistently outperforms the conservative version (blue curve). As previously observed, the latter suffers in scenarios where bandwidth becomes scarce when approaching the deadline.

To appreciate the performance of the adaptive scheduler with respect to the total upload cost, we compare it against the straightforward GT heuristic, that uploads all data as fast as possible, and (as we already saw) provides overall costs very close to the optimal solution. On average, the GT scheduler completes the upload in 33 s, with a cost of 7.5 k units. Adaptive schedulers reduce the cost to about 4 k units with conservative recovery and to about 5 k units with aggressive recovery, with savings of about 46%, and 33%, respectively. In general, the distance of the cost curves from the curve of the perfect knowledge case is quite small for very short deadlines, and remains within about 20-25% for longer deadlines. The former effect is due to the limited choice that short deadlines leave to the scheduler. The latter effect is a clear indicator of the good performance of our adaptive scheduling algorithm.

Fig. 11 shows that the conservative recovery algorithm yields lower costs, but higher percentage of missed deadlines, with respect to the aggressive recovery algorithm. It is interesting to see by how much the deadline is missed by the conservative recovery algorithm, i.e., how many more time slots the scheduler needs, to complete the video upload. Fig. 12 reports percentiles of the upload completion time, normalized to the deadline. Box-and-whiskers plots show the 90th, 80th, 20th and 10th percentiles of the upload time, for conservative (blue) and aggressive (red) recovery policies, respectively. Average values are represented by dots. Values larger than 1 show the fraction of extra time slots needed to complete the video upload. We can observe that about 10% more time slots in the case of the conservative recovery policy would allow almost 90% of successful schedulings, even in the case of tight deadlines.

While Fig. 12 considers all simulated schedulings, Fig. 13 consider only those that fail to meet the deadline, and reports the number of time slots needed to complete the upload,

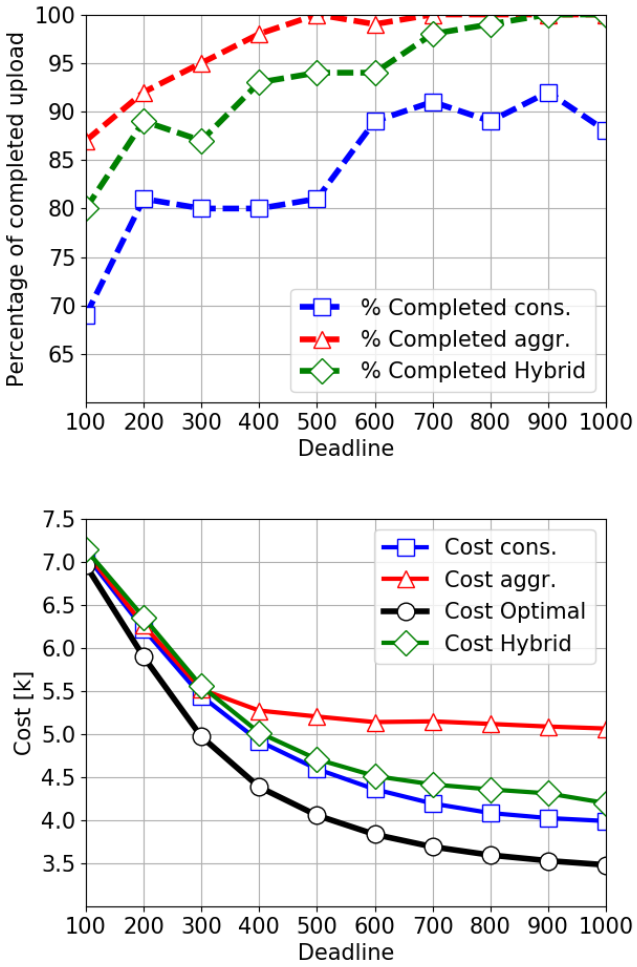


Figure 11: Percentage of completed uploads (top) and cost (bottom) versus deadline T, with $\beta = 1$ and $\alpha = 0.1$

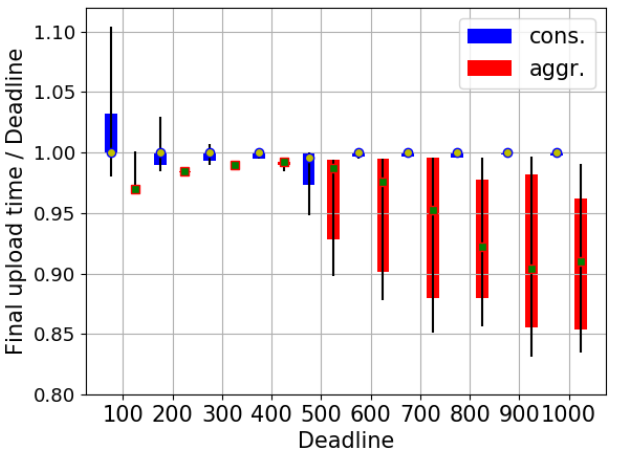


Figure 12: Distribution of final upload time over deadline versus deadline, with $\beta = 1$ and $\alpha = 0.1$

again with box-and-whiskers plots. Results show that for deadlines longer than 5 minutes (300 time slots), about 90% of the schedulings complete within a delay of half a minute (30 slots).

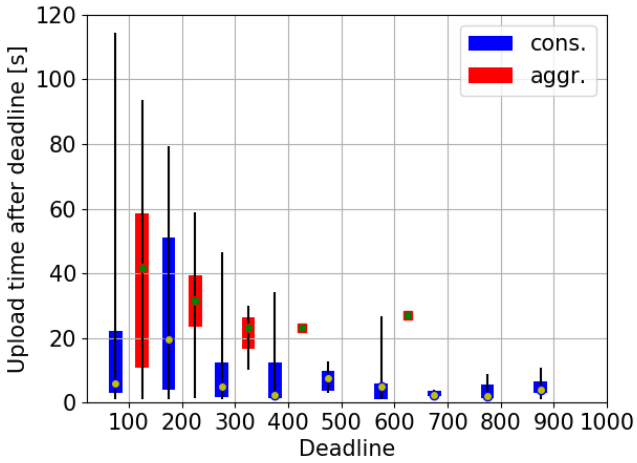


Figure 13: Distribution of final upload time over deadline versus deadline, with $\beta = 1$ and $\alpha = 0.1$ for schedulings that do not meet the deadline.

5.3.4. *Hybrid algorithm*

The fact that i) the conservative recovery algorithm achieves lower costs, and ii) the aggressive algorithm guarantees higher chances to meet the deadline, motivates a hybrid approach that tries to combine the strengths of the two approaches. In particular, we suggest to use the conservative approach in the initial part of the scheduling, while switching to the aggressive approach when getting closer to the deadline. Intuitively, at the beginning of the scheduling, the hybrid algorithm tries to decrease the cost by using a conservative recovery. The choice of $\beta = 1$ allows to push extra data into cheaper interfaces, avoiding the most expensive one. To prevent accumulating too much unsent data, and taking the risk of missing the deadline, the hybrid algorithm switches to the aggressive recovery policy when approaching the deadline.

We evaluate the performance of this hybrid approach, by using the conservative recovery for the first 90% of slots, and then switching to the aggressive recovery algorithm in the remaining 10% of slots. The green line in Fig. 11 shows that this hybrid algorithm achieves very high completion probability with very competitive cost (only about 15% higher than the oracle). This shows that the hybrid approach can properly leverage the tradeoff between cost minimization and and upload time, even under unpredictable variations in available bandwidth.

5.3.5. *Impact of available bandwidth variability*

The proposed adaptive scheduler exploits the long-term average of the available bandwidth, together with an EWMA, to predict the available bandwidth in future slots. The effectiveness of this strategy clearly depends on the variability of the available bandwidth. It is thus important to investigate what is the acceptable variance range for performance to be good. To this end, we add variance to the available bandwidth measured in our traces, by letting each sampled available bandwidth value $r_{t,i}$ become a random variable with uniform distribution in the range $[r_{t,i}(1 - X), r_{t,i}(1 + X)]$, with parameter X in [0,1]. The resulting variance added to the sampled available bandwidth value $r_{t,i}$ is:

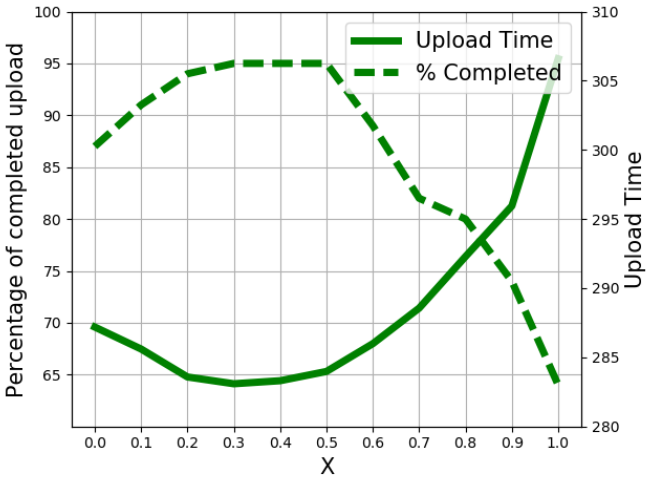


Figure 14: Effect of variation on interface available bandwidth, with deadline = 300, $\beta = 1$, and $\alpha = 0.1$

$$\sigma^2_{r_{t,i}} = \frac{X^2 r^2_{t,i}}{3} \tag{15}$$

Fig. 14 shows the percentage of completed uploads (dashed line - left y-axis) and their upload time (solid line - right y-axis) for the hybrid algorithm, versus X, with deadline 300 s, $\beta = 1$, and $\alpha = 0.1$. The reported values are the averages of 100 repetitions including the ones that failed to meet the deadline. Results indicate that the scheduler behaves well up to $X = 0.5$. A variance increases, the percentage of completed upload within the deadline can drop by 30%. Interestingly, moderate variance helps the algorithm to both reduce the cost and meet the deadline. Because, $\beta > 0$ lets the algorithm to exploit extra bandwidth on cheap interfaces. Conversely, when X increases over 50%, the unpredictability of the system reduce the performance. Notice that the average completion time remains close to the deadline.

6. Experimental evaluation

In this section we describe the design, implementation, and test of our adaptive scheduler in multihomed nodes deployed on vehicles in the framework of the MONROE H2020 project.

6.1. Experimental setup

We first discuss the experimental setup and the engineering choices adopted in the implementation of the adaptive scheduler.

6.1.1. Protocol issues

At the experiment start, the mobile node (the video sender) registers into the central office node (the video receiver). For this, the sender initiates a TCP connection that is used for signaling, and waits for a request from the central office node. In case of disconnection, the sender re-registers itself. This choice simplifies the connection handling, e.g., avoiding NAT (Network Address Translation) traversal issues, commonly encountered in today's MBB networks [20, 42]. For the video upload,

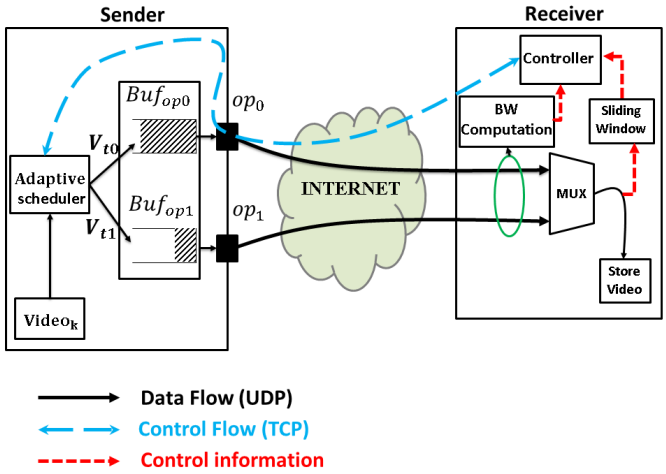


Figure 15: Internal architecture of the adaptive scheduler implementation

the choice of the transport protocol can affect both the performance of the scheduler, and the actual implementation complexity.

In the experiment scenario, TCP would need some application mechanisms to manage connections (one for each interface), which may be complex in our mobile scenario, especially for the WiFi interface, which has to reopen a connection each time it joins a new hotspot. TCP also complicates the sender buffer management, since data can be stored in the sender TCP buffer with no knowledge of how much has been actually delivered to the receiver side. This is an issue during transition phases, e.g., when switching the assigned packet from one interface to another one, since an assigned packet to an interface is pushed into its buffer and is not accessible. UDP on the contrary offers a datagram and unreliable service, with accurate delimitation of messages. This gives us the freedom to push data into the interfaces with tighter control, letting the scheduling algorithm decide the sending rate for each interface. However, UDP offers no guarantee on the delivery of messages. For this, we need to add an Automatic Repeat reQuest (ARQ) protocol [43].

In our prototype, we opt for UDP as transport protocol, and we implement a selective repeat ARQ protocol, with acknowledgements that report information on the received messages using a bitmap of 10,000 elements. Acknowledgements are generated every second, and sent on the TCP signaling channel. Acknowledgements also carry information to accurately compute the actual rate at the receiver side for each interface. The receiver calculated the data rate for each interface every second by using the information obtained from packet header. The header contains information about the node, video, interfaces, and data offset.

6.1.2. MONROE platform

MONROE is an H2020 project that has developed an open platform which allows researchers to run experiments on MBB networks in Europe. MONROE nodes are deployed in 4 countries (Italy, Norway, Spain, and Sweden), and include both stationary and mobile nodes, the latter traveling on vehicles like buses, trains, and trucks. MONROE lets users run custom ex-

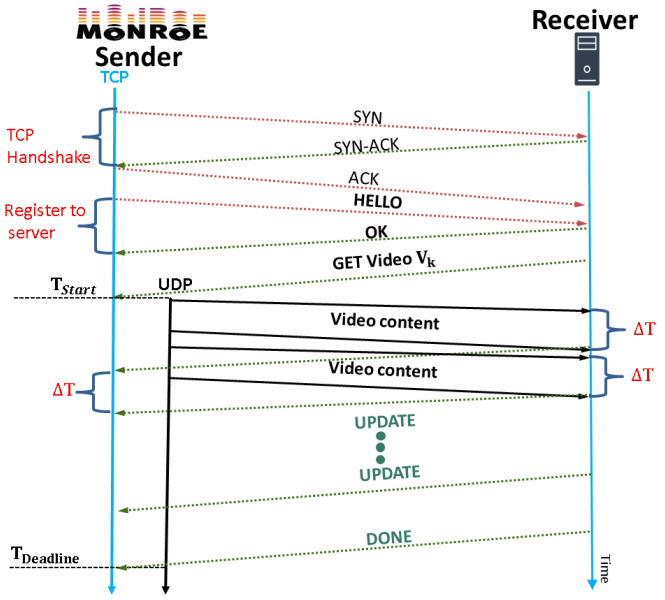


Figure 16: Experiment design

periments, identified through open calls. Each MONROE node has two MBB interfaces with subscriptions which are different in each country, as well as a WiFi interface.

Each node runs a stripped down version of Debian Linux, with a Docker [9] setup that allows experimenters to deploy their experiment by selecting the desired nodes and the experiment time by using a centralized experiment scheduler. The MONROE experiment scheduler automates the container distribution on the selected nodes, runs the experiment, and collects results.

Our video upload scheduler was designed and implemented in a Docker container, deployed by the MONROE experiment scheduler, and run over the selected mobile MONROE nodes.

6.1.3. *Experiment architecture*

Fig. 15 illustrates the internal architecture of the adaptive video upload scheduler implemented in the MONROE platform. The left side represents the sender (the public transport vehicle which uploads the video). The sender is equipped with multiple MBB interfaces. The right side represents the receiver (the security center) that requests the specific content with a given deadline. A TCP connection is used as a control channel to register the node, request the content, and send update and control messages. Update messages contain the upload rate computed at the receiver, as well as acknowledgements carrying the bitmap of the received segments to implement the ARQ protocol. The control connections consume about 1% of the video upload bandwidth.

Fig. 15 illustrates the control flow (TCP) and data flow (UDP) using blue and black arrows, respectively. The adaptive scheduler module runs the video upload scheduling algorithm and pushes data into interface buffers. The MUX module stores packets for each video on disk, while the sliding window module keeps track of all the packets that were already received. Since the adaptive scheduler module at the sender side needs the information about the experienced interface rate, every ΔT the controller module sends update messages and a bitmap of the missing packets (implementing a selective repeat ARQ protocol).

Fig. 16 shows the communication workflow in our experiment. All the connections are initiated from the sender. At the beginning, the sender starts a TCP connection (on a reliable interface - MPTCP could be used here) as control channel. After the TCP three-way handshake, the sender registers itself with an HELLO message. The HELLO message contains information about the available interfaces and videos. If the server recognizes the client, it replies with an OK message. If the TCP connection fails at any time, the client starts a new TCP connection.

The server can request a specific video portion by issuing a GET message with a specific chunk ID. Then, the sender starts sending UDP messages on selected interfaces to upload data according to the hybrid adaptive video upload scheduler. The receiver replies with UPDATE messages to report the last time slot data rate, and the bitmap of received messages. If for any reason the UPDATE message does not arrive at the end of each time slot, the sender assumes all packets of the previous slot are delivered successfully and keeps sending messages. With the next UPDATE message, the bitmap will arrive, possibly missing messages will be retransmitted, and rate adjusted. Messages are 1,000 Byte long. With selective acknowledgements carrying the status of the last 10,000 messages, we have an equivalent window of 10 MB. Since ACKs are sent every second, this lets us reach a data rate equal to 80 Mb/s. [10]

6.2. *Experimental results*

We started our experimental analysis by using stationary nodes with three interfaces: 2 MBB interfaces, named op0 and op1, and 1 Ethernet wired connection, which was used to emulate a WiFi connection [11] by using the Linux traffic control tool[12] to limit the bandwidth and impose random (1%) packet loss. We assumed that the interface costs change over time, to be able to illustrate and evaluate the behavior of the video upload scheduling algorithm.

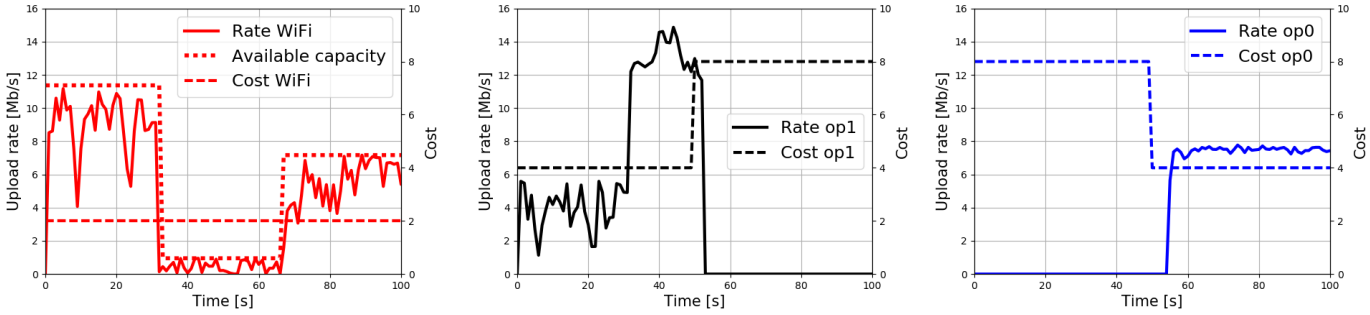
Fig. 17 shows all interface upload rates (solid lines, left y-axis) and costs (dashed lines, right y-axis) versus time in seconds. The dotted line in Fig. 17a shows the available bandwidth of the WiFi interface (emulated on the Ethernet connection). In the interval [35,65] seconds we simulate a period of lack of coverage to see if the scheduler is going to use more expensive interfaces. Fig. 17b shows that op1 is selected to support the transmission of data that cannot make it through the WiFi interface. The cost of the interfaces op0 and op1 changes after 50 seconds, and the adaptive scheduler switches to the now cheaper interface to upload data. This proves that the video upload scheduler operates as expected.

[9] https://www.docker.com/

[10] 8*1000*10000=80Mb/s

[11] At the time of the experiment campaign, the MONROE nodes were not yet incorporating a WiFi interface

[12] http://lartc.org/manpages/tc.txt



(a) WiFi (b) op1 (c) op0

Figure 17: Adaptive scheduler experiment on a stationary MONROE node

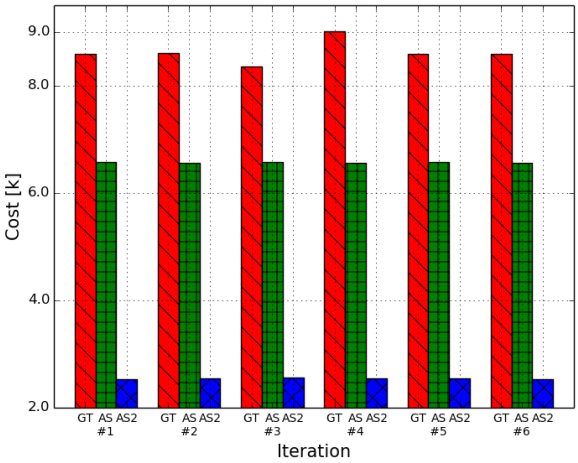


Figure 18: Cost comparison of GT-100 (GT), AS-100 (AS), and AS-200 (AS2) in MONROE node traveling on public transport vehicle in 6 iterations

We next present results obtained by running experiments on mobile MONROE nodes equipped with two 4G MBB interfaces and one WiFi interface. We compare the results obtained with the hybrid video upload scheduler against those obtained with the GT heuristic, which is the only heuristic that can schedule the video content with no a priori knowledge of the interface available bandwidth.

We compare the total cost of the GT (GT-100) algorithm, and of the adaptive scheduler (AS-100) with deadline of 100 s, and of the adaptive scheduler (AS-200) with deadline of 200 s. In order to allow the reader to get a quantitative view of the scenarios experienced by the tests, consider that RTT, available bandwidth, and Received Signal Strength Indicator (RSSI) were measured in the ranges $[40, 120]$ ms, $[0, 80]$ Mb/s, and $[-30, -100]$ dBm, respectively. The tests for GT-100, AS-100, and AS-200 were run back to back with 20 seconds idle time (batch of experiment), repeating each experiment 6 times. Fig. 18 shows the total costs (y-axis) for the three schedulers in 6 batch of experiments (x-axis). We can see that AS-100 (AS) obtains solutions about 30% cheaper than GT-100 (GT), while completing the upload within the deadline. As expected, AS-200 (AS2) uploads the video with 40% lower cost with respect to AS-100 (AS).

7. Conclusions and outlook

In this paper, we considered the practical problem of video surveillance in connected public transport vehicles, where security videos are stored onboard, and a central operator sometimes requests to watch portions of the videos. At these requests, the selected video must be uploaded within a given deadline, by using the wireless network interfaces available on the vehicle, considering interfaces have different associated costs. The video upload goal is to minimize the total cost of the upload while meeting the deadline.

To identify an effective solution to our problem, we explored several aspects. In order to obtain a benchmark, we first considered the case where an oracle has perfect knowledge about available bandwidth of wireless links; we formalized the corresponding optimization problem and proposed greedy heuristics. Second, we looked at the case where the distribution of the available bandwidth on wireless interfaces is known; we defined the corresponding stochastic optimization problem, then we found that its solution is computationally extremely costly. We thus explored a family of adaptive scheduling algorithms that require only a coarse knowledge of the available bandwidth on wireless interfaces. Results show that our adaptive scheduling approach can effectively leverage the fundamental trade-off between the total video upload cost and the video delivery time, despite unknown short-term variations in throughput on wireless links. Finally, we implemented and tested our adaptive algorithm in the platform for wireless network experiments provided by the MONROE H2020 project.

We believe, there are particular aspects of the algorithm can be evaluated more deeply. Firstly, we plan to compare the performance of the proposed adaptive scheduler with MP-DASH and other open source video adaptive scheduler. Secondly, the parameters can be define by dynamic approach instead of in advance tuning. Thirdly, the cost also can be modeled for the general multihomed MBB devices.

Acknowledgements

This work was funded by the *MONROE* H2020 project (grant agreement no. 644399). Computational resources were provided by hpc@polito, which is a project of Academic Comput-

ing within the Department of Control and Computer Engineering at the Politecnico di Torino.

References

[1] M. Ergen, *Mobile Broadband - Including WiMAX and LTE*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[2] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Rfc 6182 - architectural guidelines for multipath tcp development," 2011.

[3] Nikravesh, A. and Guo, Y. and Qian, F. and Mao, Z. M. and Sen, S., "An in-depth understanding of multipath tcp on mobile devices: Measurement and system design," ser. MobiCom '16. New York, NY, USA: ACM, 2016, pp. 189–201.

[4] K. Fall, "A delay-tolerant network architecture for challenged internets," ser. SIGCOMM '03. New York, NY, USA: ACM, 2003, pp. 27–34.

[5] Safari Khatouni, A.; Ajmone Marsan, M. and Mellia, M., "Delay tolerant video upload from public vehicles," ser. SmartCity'16. INFOCOM Workshop, 2016, pp. 213–218.

[6] A. S. Khatouni, M. A. Marsan, M. Mellia, and R. Rejaie, "Adaptive schedulers for deadline-constrained content upload from mobile multihomed vehicles," in *2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, June 2017, pp. 1–6.

[7] Alay, Ö; Lutu, A.; Peón-Quirós, M.; Mancuso, V.; Hirsch, T.; Evensen, K.; Hansen, A.; Alfredsson, S.; Karlsson, J.; Brunstrom, A.; Safari Khatouni, A.; Mellia, M.; Ajmone Marsan, M.;, "Experience: An open platform for experimentation with commercial mobile broadband networks," in *The 23rd Annual International Conference on Mobile Computing and Networking (MobiCom 2017)*, October 2017.

[8] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4g lte networks," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '12. New York, NY, USA: ACM, 2012, pp. 225–238. [Online]. Available: http://doi.acm.org/10.1145/2307636.2307658

[9] Huang, Junxian and Qian, Feng and Gerber, Alexandre and Mao, Z. Morley and Sen, Subhabrata and Spatscheck, Oliver, "A close examination of performance and power characteristics of 4g lte networks," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '12. New York, NY, USA: ACM, 2012, pp. 225–238. [Online]. Available: http://doi.acm.org/10.1145/2307636.2307658

[10] J. Huang, Q. Xu, B. Tiwana, Z. M. Mao, M. Zhang, and P. Bahl, "Anatomizing application performance differences on smartphones," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 165–178. [Online]. Available: http://doi.acm.org/10.1145/1814433.1814452

[11] B. M. Sousa, K. Pentikousis, and M. Curado, "Multihoming management for future networks," *Mobile Networks and Applications*, vol. 16, no. 4, pp. 505–517, Aug 2011. [Online]. Available: https://doi.org/10.1007/s11036-011-0323-5

[12] B. D. Higgins, A. Reda, T. Alperovich, J. Flinn, T. J. Giuli, B. Noble, and D. Watson, "Intentional networking: Opportunistic exploitation of mobile network diversity," in *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '10. New York, NY, USA: ACM, 2010, pp. 73–84. [Online]. Available: http://doi.acm.org/10.1145/1859995.1860005

[13] Deng, S. and Netravali, R. and Sivaraman, A. and Balakrishnan, H., "Wifi, lte, or both?: Measuring multi-homed wireless internet performance," ser. IMC '14. New York, NY, USA: ACM, 2014, pp. 181–194.

[14] Rahmati, A. and Zhong, L., "Context-for-wireless: Context-sensitive energy-efficient wireless data transfer," ser. MobiSys '07. New York, NY, USA: ACM, 2007, pp. 165–178.

[15] Rathnayake, U.; Petander, H. and Ott, M., "Emune: Architecture for mobile data transfer scheduling with network availability predictions," *Springer US*, 2012-04.

[16] Riiser, H.; Vigmostad, P.; Griwodz, C. and Halvorsen, P., "Commute path bandwidth traces from 3g networks: Analysis and applications," ser. MMSys '13. New York, NY, USA: ACM, 2013, pp. 114–118.

[17] Chen, Y.; Nahum, E. M.; Gibbens, R. J.; Towsley, D. and Lim, Y., "Characterizing 4g and 3g networks: Supporting mobility with multi-path tcp," UMass Amherst Technical Report, Tech. Rep., 2012.

[18] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong, "Mobile data offloading: How much can wifi deliver?" *IEEE/ACM Transactions on Networking*, vol. 21, no. 2, pp. 536–550, April 2013.

[19] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, and S. Madden, "A measurement study of vehicular internet access using in situ wi-fi networks," in *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '06. New York, NY, USA: ACM, 2006, pp. 50–61. [Online]. Available: http://doi.acm.org/10.1145/1161089.1161097

[20] Safari Khatouni, A.; Mellia, M.; Ajmone Marsan, M.; Alfredsson, S.; Karlsson, J.; Brunstrom, A.; Alay, Ö; Lutu, A.; Midoglu, C.; Mancuso, V.;, "Speedtest-like measurements in 3g/4g networks: the monroe experience," in *29th International Teletraffic Congress - ITC29*, September 2017, pp. 4–8.

[21] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley, "How hard can it be? designing and implementing a deployable multipath TCP," in *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. San Jose, CA: USENIX Association, 2012, pp. 399–412. [Online]. Available: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/raiciu

[22] J. Wu, C. Yuen, B. Cheng, M. Wang, and J. Chen, "Streaming high-quality mobile video with multipath tcp in heterogeneous wireless networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 9, pp. 2345–2361, Sept 2016.

[23] Y.-C. Chen, Y.-s. Lim, R. J. Gibbens, E. M. Nahum, R. Khalili, and D. Towsley, "A measurement-based study of multipath tcp performance over wireless networks," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. New York, NY, USA: ACM, 2013, pp. 455–468. [Online]. Available: http://doi.acm.org/10.1145/2504730.2504751

[24] Q. Peng, M. Chen, A. Walid, and S. Low, "Energy efficient multipath tcp for mobile devices," in *Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '14. New York, NY, USA: ACM, 2014, pp. 257–266. [Online]. Available: http://doi.acm.org/10.1145/2632951.2632971

[25] A. Nika, Y. Zhu, N. Ding, A. Jindal, Y. C. Hu, X. Zhou, B. Y. Zhao, and H. Zheng, "Energy and performance of smartphone radio bundling in outdoor environments," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2015, pp. 809–819. [Online]. Available: https://doi.org/10.1145/2736277.2741635

[26] Y.-s. Lim, Y.-C. Chen, E. M. Nahum, D. Towsley, R. J. Gibbens, and E. Cecchet, "Design, implementation, and evaluation of energy-aware multi-path tcp," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '15. New York, NY, USA: ACM, 2015, pp. 30:1–30:13. [Online]. Available: http://doi.acm.org/10.1145/2716281.2836115

[27] Y.-s. Lim, Y.-C. Chen, E. M. Nahum, D. Towsley, and R. J. Gibbens, "How green is multipath tcp for mobile devices?" in *Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, & Challenges*, ser. AllThingsCellular '14. New York, NY, USA: ACM, 2014, pp. 3–8. [Online]. Available: http://doi.acm.org/10.1145/2627585.2627596

[28] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan, "Mp-dash: Adaptive video streaming over preference-aware multipath," in *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '16. New York, NY, USA: ACM, 2016, pp. 129–143. [Online]. Available: http://doi.acm.org/10.1145/2999572.2999606

[29] B. Han, P. Hui, V. A. Kumar, M. V. Marathe, G. Pei, and A. Srinivasan, "Cellular traffic offloading through opportunistic communications: A case study," in *Proceedings of the 5th ACM Workshop on Challenged Networks*, ser. CHANTS '10. New York, NY, USA: ACM, 2010, pp. 31–38. [Online]. Available: http://doi.acm.org/10.1145/1859934.1859943

[30] B. Han, P. Hui, and A. Srinivasan, "Mobile data offloading in metropolitan area networks," *SIGMOBILE Mob. Comput. Commun.*

Rev., vol. 14, no. 4, pp. 28–30, Nov. 2010. [Online]. Available: http://doi.acm.org/10.1145/1942268.1942279

[31] J. Whitbeck, M. Amorim, Y. Lopez, J. Leguay, and V. Conan, "Relieving the wireless infrastructure: When opportunistic networks meet guaranteed delays," in *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, June 2011, pp. 1–10.

[32] O. B. Yetim and M. Martonosi, "Adaptive delay-tolerant scheduling for efficient cellular and wifi usage," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, June 2014, pp. 1–7.

[33] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3g using wifi," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 209–222. [Online]. Available: http://doi.acm.org/10.1145/1814433.1814456

[34] Zaharia, M. A. and Keshav, S., "Fast and optimal scheduling over multiple network interfaces," University of Waterloo, Tech. Rep., 2007.

[35] M.-R. Ra, J. Paek, A. B. Sharma, R. Govindan, M. H. Krieger, and M. J. Neely, "Energy-delay tradeoffs in smartphone applications," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 255–270. [Online]. Available: http://doi.acm.org/10.1145/1814433.1814459

[36] Lutu, A.; Raj Siwakoti, Y.; Alay, Ö; Baltrūnas, Džiugas and Elmokashfi, Ahmed, "The good, the bad and the implications of profiling mobile broadband coverage," *Computer Networks*, 2016.

[37] Nikravesh, A.; Choffnes, D. R.; Katz-Bassett, E.; Mao, Z. M.; Welsh, M., *Mobile Network Performance from User Devices: A Longitudinal, Multidimensional Analysis*. Springer International Publishing, 2014, pp. 12–22".

[38] Ahuja, R. K.; Magnanti, T. L. and Orlin, J. B., *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[39] J. Orlin, "A faster strongly polynomial minimum cost flow algorithm," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, ser. STOC '88. New York, NY, USA: ACM, 1988, pp. 377–387. [Online]. Available: http://doi.acm.org/10.1145/62212.62249

[40] "IBM ILOG CPLEX Optimization Studio 12.6.0.0." [Online]. Available: http://www-01.ibm.com/support/knowledgecenter/SSSA5P_12.6.0/ilog.odms.studio.help/Optimization_Studio/topics/COS_home.html

[41] Magharei, N. and Rejaie, R., "Adaptive receiver-driven streaming from multiple senders," *Multimedia Systems*, vol. 11, no. 6, pp. 550–567, 2006.

[42] S. Triukose, S. Ardon, A. Mahanti, and A. Seth, "Geolocating ip addresses in cellular data networks," in *Passive and Active Measurement*, N. Taft and F. Ricciato, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 158–167.

[43] L. L. Peterson and B. S. Davie, *Computer Networks, Fifth Edition: A Systems Approach*, 5th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.