Adaptive Schedulers for Deadline-Constrained Content Upload from Mobile Multihomed Vehicles

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

Abstract— **We consider the practical problem of video surveillance in public transport systems, where security videos are stored onboard, and a central operator occasionally needs to access portions of the recordings. When this happens, the selected video must be uploaded within a deadline, possibly using multiple parallel wireless interfaces. Interfaces have different associated costs, related to tariffs charged by Mobile Network Operators (MNOs), energy consumption, data quotas, system load. Our goal is to choose which interfaces to use, and when, so as to minimize the cost of the upload while meeting the deadline, despite the unknown short-term variations in throughput. To achieve this goal, we first collect real traces of mobile uploads from vehicles for different MNOs. Examination of these traces confirms the unpredictability of the short-term throughput of wireless connections, and motivates the adoption of adaptive schedulers with limited a-priori knowledge of the system status.**

To effectively solve our problem, we devised a family of adaptive algorithms, that we thoroughly evaluated using a trace-driven approach. Results show that our adaptive approach can effectively leverage the fundamental tradeoff between the total cost and the delivery time of content upload, despite unknown short-term variations in throughput.

I. 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 capacity offered by wireless links. Consider for instance public transport vehicles (like buses or trains) equipped with multiple Mobile BroadBand (MBB) high speed interfaces, where on-board security cameras record videos that must be uploaded to a security center where an operator occasionally requests to watch selected portions of a video. In this scenario, continuous real-time video uploading is too expensive. Even if MBB networks can offer capacities up to 100 Mb/s, the number of videos, the limited data quota, the performance variability along the path, and the need to check only portions of the videos, call for ingenious upload policies. Hence, videos are stored on board, and, only when an alarm is triggered, the Security Operator (SO) on duty requests a specific portion of the video that should be uploaded before a specified deadline.

This work is funded by the European Union's Horizon 2020 research and innovation program under grant agreement No. 644399 (MONROE).

[1]Politecnico di Torino, Italy - name.lastname@polito.it

[2]University of Oregon, USA - reza@cs.uoregon.edu

[3]Institute Imdea Networks, Spain

978-1-5386-0728-2/17/$31.00

We model this problem as the scheduling of content upload from multihomed mobile nodes, where the content must be delivered within a given deadline, while the cost must be minimized. This differs from the classic problem of content upload using multihomed nodes, where upload delay has typically to be minimized. Similarly, no real time constraint is posed, thus making the problem different from video streaming, and partly similar to a delay tolerant scenario, albeit the hard deadline for delivery of the entire content (rather than individual packets) must be met. 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 and 4G 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, at the expense of ignoring opportunities for cheaper networks to become available, thus increasing upload cost. There is a clear tradeoff between minimizing the total transmission cost and upload completion time.

In our previous work [1], we formulated the problem as a centralized scheduling problem, where an oracle has the perfect knowledge of the upload rate of 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. In this paper, we instead assume only a coarse knowledge of available capacity, and we design online, adaptive schedulers to explore the tradeoff between cost and delivery time. We use a trace driven approach to run a thorough performance evaluation in realistic scenarios, and gain insight about the tradeoff between cost and delivery time over wireless channels with unpredictable short-term variations in throughput. The actual difference with respect to the optimal solution is likely to depend on the specific traces/setting/etc. Therefore, the exploration of the tradeoff is important, and shows the ability of the scheduler to operate in this challenging environment, which is likely in practical settings.

The rest of this paper is organized as follows. Section II briefly overviews related work. Section III introduces the problem formulation and our proposed algorithm. Section IV describes the setup used for simulations, and section V discusses the numerical results obtained from the simulations. Finally, Section VI concludes the paper.

II. Related work

Mobile devices allow users to connect to multiple wireless networks with possibly different technologies, obtaining

throughput values which depend on the node position, the network coverage, the traffic load, the weather conditions, etc. This makes the problem of scheduling transmissions over multiple wireless interfaces both challenging and relevant.

The performance of wireless service under uncertain network availability has been previously investigated by several authors. Deng et al. [2] investigate the characterization of multihomed systems considering WiFi vs. LTE in a controlled experiment. They show that LTE can have a better performance than WiFi, also demonstrating the variability of performance on both short and long time scales. Rahmati et al. [3] present a technique for estimating and learning the Wi-Fi network conditions from a fixed node. Rathnayake et al. [4] demonstrate how a prediction engine may 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., which packet to send through which interface, to maximize the total throughput. Along this line, recent works focus on multipath TCP (MPTCP), where the design of packet schedulers and congestion control algorithms are studied. The goal is to maximize throughput, or equivalently to minimize upload time, e.g., see the work [5]. Our work differs, since we are dealing with moving vehicles that exacerbate the unpredictability of the network performance. For instance, Riiser et al. [6] collected 3G mobile network traces from onboard public transport vehicles around the city of Oslo (Norway). Similarly, Chen et al. [7] measured the throughput of both single-path and multipath data transport in 3G, 4G, and Wi-Fi networks. In both cases, variability is much higher than for fixed nodes. We too collect traces to gauge unpredictability, and we use them to run trace-driven evaluations in realistic scenarios. More important, the presence of a deadline makes our work totally different, i.e., the goal is not any more to maximize the instantaneous upload rate, rather, to minimize the total cost of uploading the content within the deadline. Delay tolerant networks [8] face a similar problem, but the data delivery problem has typically no deadline, and the main problem is the creation of the time varying network topology to guarantee the data delivery. Here, we relay on MBB to offer connectivity.

Previous works focused on deadline scheduling under the assumption that network performance is perfectly known. Zaharia et al. [9] present an optimal scheduler over multiple network interfaces, and propose approximations which can be implemented with limited resources in mobile phones, or PDAs. They assume the cost and capacity of each interface to be constant. In [1] we also assume perfect knowledge of capacity. Here, we relax this assumption, requiring the scheduler to cope with unknown short-term variations in throughput of individual interfaces.

III. Problem Formulation and Algorithms

Our adaptive algorithm is inspired by an adaptive scheduler for P2P video streaming proposed by Magharei et al. [10]. However, the dynamics of variations for individual connections as well as the design goals in our problem are different. Our only assumption is that the long-term average throughput on each interface is known. This assumption only provides a base line (or a reference) to assess the feasibility and 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 knowledge of what happened during the past slots. It updates the expected rate on each interface based on the overall pace of upload progress during the past recent slots, and schedules the transmission of a portion of the data, giving preference to cheaper interfaces. During the slot, data is transmitted at the chosen rate, according to 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 missing rate, denoted $LeftB$, is greater than zero. In this case, the system is behind schedule, and the scheduler needs to recover in the future. We consider two policies for recovery: i) aggressively recovering during the immediate next slot, ii) conservatively (i.e. optimistically) recovering across all the remaining slots before the deadline.

Let i be the current time slot. B_i represents the expected data rate at which the system should transmit during slot i. This rate must be separately calculated for each slot. At the upload start, we estimate $B_0 = V/S$, being V the total data volume size, and S 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 over all interfaces, e.g., due to a lack of capacity in that slot. To help the scheduling, each interface j maintains and updates the estimated expected rate $\widehat{R}_{ij}$ based on the actual transmission rate R_{ij}. If the interface j was active in period i ($R_{ij} > 0$) and congested ($LeftB_j > 0$), $\widehat{R}_{ij}$ is updated using an Exponentially Weighted Moving Average (EWMA) algorithm with α coefficient:

$$\widehat{R}_{i+1j} = \begin{cases} \alpha\widehat{R}_{ij} + (1-\alpha)R_{ij} \text{ if } R_{ij} > 0 \; and \; LeftB_j > 0 \\ Max(\widehat{R}_{ij}, R_{ij}) \text{ if } R_{ij} > 0 \; and \; LeftB_j = 0 \\ \widehat{R}_{ij} \text{ otherwise} \end{cases} \tag{1}$$

The rationale of the expression above is to avoid the estimated rate to converge to small values (practically zero) when an interface is not being used, or used at a rate lower than the maximum available rate, i.e., when the interface bandwidth is not fully utilized. Indeed, if $LeftB_j = 0$, the available capacity of the interfaces allowed to transmit all data using at least the expected rate. However, the actual available rate of the interface is unknown. Thus, we avoid decreasing the estimated rate of these interfaces that are not fully utilized. This could create problems if the algorithm falls behind the schedule, and needs to use this interface at the highest possible bit rate, because $\widehat{R}_{ij}$ would provide a gross underestimation. This happens because the algorithm is not greedy, and more expensive interfaces are partially used in a demand-driven fashion. Alg. 1 presents an overview of our proposed algorithm for adaptive scheduling. After initialization, the algorithm loops for each time slot until the deadline is reached, or all the data have been uploaded. At the beginning of each time slot i, the system has to schedule data for transmission (line 6),

Algorithm 1 Adaptive Scheduler

```
1:  procedure AdaptiveScheduler(α, β, policy)
2:      R̂_0j ← Interface average rates
3:      B_0 ← V/S
4:      for (i = 0; i < S && V > 0; i++) do
5:          SortInterfaceByCost()
6:          procedure Push data to buffers
7:              for (j = 1; j ≤ I && V_i > 0; j++) do
8:                  if cost(j) < maxcost then
9:                      V_i = (β + 1)B_i ΔT
10:                 else
11:                     V_i = B_i ΔT
12:                 V_ij = min(V_i, R̂_ij ΔT)
13:                 V_i = max(V_i − V_ij, 0)
14:         UploadAndWaitForSlotEnd()
15:         procedure Check data in buffers
16:             V = V − Σ_j R_ij ΔT
17:             for (j = 1; j ≤ I; j++) do
18:                 LeftB_j = V_ij − R_ij ΔT
19:                 if R_ij > 0 && LeftB_j > 0 then
20:                     R̂_{i+1 j} ← α R̂_ij + (1 − α) R_ij
21:                 else if R_ij > 0 && LeftB_j = 0 then
22:                     R̂_{i+1 j} ← Max(R̂_ij, R_ij)
23:                 else
24:                     R̂_{i+1,j} ← R̂_ij
25:         LeftB = Σ_j LeftB_j
26:         if LeftB > 0 then
27:             if policy == Aggressive then
28:                 B_{i+1} = B_0 + LeftB
29:             else
30:                 B_{i+1} = B_i + LeftB/(S − i)
```

for an amount equal to $V_i = B_i \Delta T$ if it is considering the most expensive interface, otherwise, $V_i = (\beta + 1) B_i \Delta T$. $\beta \in \mathbb{R}$ 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, the scheduler utilizes any excess bandwidth on the cheap interface to deliver more data in the next slot and stay ahead of the schedule. This in turn increases the chance of completing the upload before the deadline even if available bandwidth drops below the expected value in the remaining slots. The parameter β allows pushing extra data into the interface buffer to efficiently use the bandwidth of cheap interfaces. By setting $\beta = 0$ for the most expensive interface, we avoid any extra load on that interface in order to minimize its use and thus the overall cost of upload (line 8). Since the algorithm aims to minimize cost, the scheduler splits the data into segments that are stored in the buffers of individual interfaces according to their expected available rate (line 12). To minimize cost, interfaces are prioritized for data transmission from the least to most expensive one (line 5).

Data is then transmitted over all interfaces whose buffer is not empty. At the end of each slot, the algorithm updates

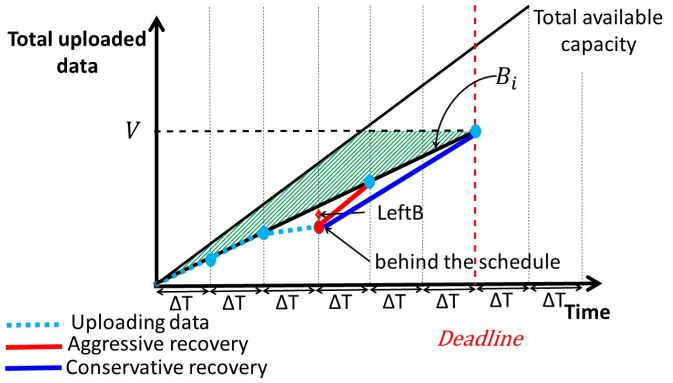


Fig. 1: Scheduling process over time and recovery strategies

the amount of remaining data that should still be transmitted (line 16), and may update the transmission rate (line 18), and the expected rate (line 19-24).

If the transmission rate is smaller than expected, i.e., data is removed from each interface buffer, and accumulated in $LeftB > 0$, the system has to recover by pushing more data over interfaces in the upcoming slots. If the *aggressive* recovery policy is selected, it tries to recover in the immediately following slot (line 28). If instead the *conservative* recovery policy is selected, it spreads the left-over data through all the remaining slots till the specified deadline, which leads to a higher average transmission rate in remaining slots (line 30).

Fig. 1 illustrates the evolution of the uploading process over time. At each ΔT, the algorithm schedules the amount of data to be uploaded at the expected rate B_i. 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 long term (conservative policy, blue line).

As a reference, we consider the *Greedy-in-Time* (GT) [1] heuristic, which simply starts uploading content through all interfaces, using the maximum rate in those slots. This would be the solution typically adopted on related works that aims at maximizing the total throughput [3], [4], [5]. We use the solution provided by the Greedy-in-Cost (GC) [1] heuristic to obtain a lower bound for cost to assess the performance of our adaptive scheduling scheme against the GT and GC heuristics.

IV. Simulation Setup

We now describe the simulation setup we use to run performance evaluation. We first present samples of actual mobile traces that we collected from vehicles with the dual purpose to show how unpredictable upload rate is, and to evaluate our proposed scheduling scheme using trace-driven simulations.

A. Collection of mobile traces

A credible evaluation of the proposed algorithms calls for realistic data about upload rate available from public transport vehicles. In our search for usable traces comprising video upload data, we found the traces in [7], [6]. However, these traces exhibit a lower bandwidth values that one experiences over today's wireless networks. Lutu et al. [11] collected a large set of recent traces for MBB on public transport vehicles.

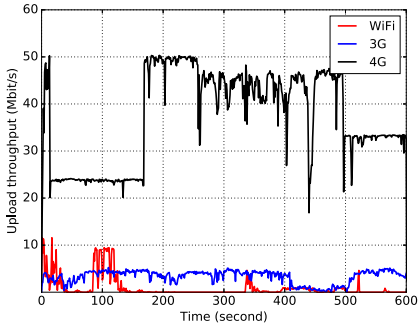


Fig. 2: A sample of collected traces for each technology

However, their traces refer to download, and only consider the transfer of 4MB of data, which is much less than what we need in our video upload problem. We thus resolved to collect our own traces by using mobile terminals on board of private and public vehicles, or carried by users walking.

1) Trace collection methodology: All traces were collected in the city of Torino in Italy, and refer to three technologies (WiFi, 3G, and 4G), and multiple MNOs. The 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, using both TCP and UDP at the transport layer.

We used a hybrid method in the trace collection process: In each experiment, the mobile terminal runs `iperf2` in the upload direction for 600 seconds while `tcpdump` 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 throughput on each technology. Moreover, it is important that repetitions cover different times of the day and different days of the week. We repeated the experiment on the same paths for at least 5 times, during different days.

We collected traces for three different MNOs in Italy (namely Tim, Wind, and Vodafone), with the objective of obtaining a wide sample of the upload throughput in MBB networks. For WiFi, we considered the open WiFi community "WoW-Fi" offered automatically by Fastweb customers that share their DLS or FTTH home network via the access gateway.[1] We make the collected traces available for researchers[2].

2) Trace characterization: While a detailed characterization of these traces are beyond the scope of this paper, we present a few evidences of variability of throughput in such a setting which makes the scheduling of content upload a challenging task. Fig. 2 shows a sample of temporal evolution of (per second) upload rate across the collected traces for each type of interface. Starting times of traces have been re-aligned for ease of visualization.

To give more details, Table I shows, as a summary of statistics for three technologies, the average, standard deviation, 80-th percentile, maximum, and minimum of the observed upload throughput. Table II reports the same statistics, but considering the absolute change of throughput in two consecutive time slots. In a nutshell, measurements indicate that predicting the exact value of the future available bandwidth is not a trivial task, as claimed by Nikravesh et al. [12]. This provides a clear motivation for the use of adaptive schedulers.

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

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

TABLE I: 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 II: $|Th_t - Th_{t-1}|$ 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

B. Experiments setup

We implemented a custom simulator using Python. It models the upload of $K = 2$ videos from a mobile vehicle equipped with a WiFi, a 3G, and a 4G interface. The cost associated with each interface is arbitrary, assumed constant over time, and taken to be 2, 4, and 8 $(Mb)^{-1}$, respectively. videos have a size equal to 62.5 MB ($V = 125$ MB in total), corresponding to about 5 minutes of 1080p video. The simulator implements the adaptive schedulers, and data is transmitted considering capacity r_{ij} from traces. The simulation time (which corresponds to the upload deadline S) varies in the range of [100,600] seconds.

We investigate parameter setting to better understand their effect on performance. α gets values in $[0,1]$. As described in Sec. III, we push some extra data β into all interface buffers, except the most expensive interface (4G in our case). β takes values in $[0,30]$. The impact of these parameters is investigated in Sec. V-A. We use the collected traces as a starting point to perform a realistic evaluation, but, to make sure that our results are not specific to a particular trace or to its short-term dynamics, we select a random combination of traces for each simulation run, and choose a random starting point for the selected trace. Each simulation is repeated 200 times with randomly selected traces as input to avoid any bias in our results toward a specific trace. We present the average and standard deviation of our performance metrics across all runs.

For some combinations of parameters, the upload of the video may not terminate within the specified deadline due to the lack sufficient bandwidth, or improper scheduling strategy. To assess performance, we thus measure the percentage of successfully completed uploads, as well as the overall upload cost C (considering only successfully completed uploads). After tuning parameters, we compare results against the case of an oracle that knows the upload throughput in future slots. To this end, we also solve the video upload problem with the centralized Greedy-in-Cost heuristic, which gives a total cost extremely close to the theoretical optimal solution [1].

V. Simulation Results

In this section, we describe the results of our simulations, using different algorithmic variations. We first discuss the tuning of the algorithm parameters. Then, we compare the algorithm performance against the optimal solution in case of known bandwidth availability.

A. Parameter setting

Our scheduler has three basic parameters as follows: ΔT, the duration of the time slot, α, the parameter of the EWMA

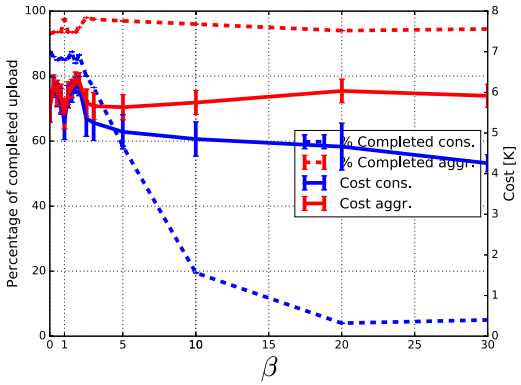


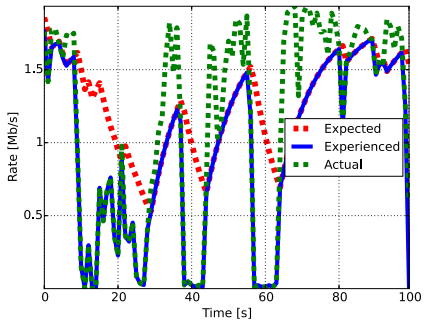
Fig. 3: Percentage of completed uploads and C versus β with $S = 300\,\text{s}$, $\alpha = 0.1$. Trying to push a little more data than expected has positive benefits on the aggressive algorithm, but dramatic effects on the conservative algorithm

bandwidth estimator, and β, the optimism coefficient. 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 long 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 adapting too frequently. The collected traces for WiFi, 3G, and 4G show bandwidth changes on a scale of seconds. For this reason, we chose for ΔT a value equal to 1 second. We next investigate the impact of α, which determines the timescale of the rate estimation for each interface. Results show that performance is rather insensitive to the value of α. Recalling that α drives the EWMA estimation of the $\widehat{R}_{ij}$, we can conclude that this is not a particularly critical aspect of the scheduling problem. That is, even a very coarse estimation of the link throughput is sufficient to achieve our goals. For the derivation of results we fix $\alpha = 0.1$. [3]

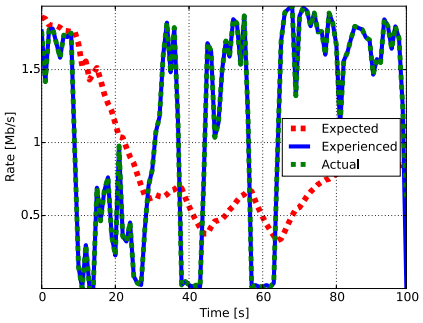
The choice of β is rather less intuitive. To illustrate this point, Fig. 3 presents the percentage of completed uploads and their total cost for different values of β. We clearly see that the value of β has a significant impact on the percentage of completed uploads and on their cost. Fig. 3 indicates that the aggressive algorithm (red curves) is not sensitive to β (line is flat), which means that aggressively adapting is more important than optimistically spreading the extra data across all the remaining intervals. It also shows that the conservative approach does not work with large values of β, since the required rate of delivery to catch up may not become available.

Indeed, increasing β suggests that optimistically sending more data through the cheaper interfaces reduces overall costs. However, due to the insufficient bandwidth across the remaining slots, the gap between the expected and actual pace of progress ($LeftB$) keeps increasing. Therefore, the conservative recovery strategy is unable to catch up and meet the deadline. Notice, when the deadline approaches, the chance the system does not offer enough capacity to complete the transfer becomes high, causing a failure to complete the upload. Notice indeed how the chance to meet the deadline

[3] We explored ΔT and α values in $\{1, 2, 5, 10\}$ and $\{0.1, 0.3, 0.5, 0.7, 0.8, 0.9\}$, respectively. Results are omitted due to lack of space.



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



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

Fig. 4: The aggressive scheduler operation with $\beta = 0$ (upper), $\beta = 5$ (lower), and $\alpha = 0.1$

suddenly decreases for increasing β, with most uploads failing for $\beta > 5$.

To further examine the effect of β on the performance of the scheduling scheme, Fig. 4 shows the evolution of the 3G rate over time during an experiment, with $\beta = 0$ and $\beta = 5$, respectively. The green line is the available bandwidth, the red line is the EWMA of available rate, and the blue line is the experienced rate. The closer the blue curve is to the green curve, the more the system is able to exploit the capacity of a cheap interface. We clearly see that the choice $\beta = 0$ lets a large fractions of the actually available capacity on 3G go unused. This forces the system to use the expensive 4G interface. Setting $\beta = 5$ makes the system more optimistic, and prone to send more data than the current capacity estimation would allow. If successful, this increases the utilization of the cheaper interfaces and reduces the load on the most expensive interface.

B. *Impact of deadline*

We look now at the influence of the deadline on performance. Fig. 5 shows the percentage of completed uploads and their total cost, for values of $S \in [100, 600]\,s$, with $\beta = 0$. As we can expect, longer deadlines imply higher percentages of completed uploads, and lower costs. Considering the percentage of completed uploads, we see that the aggressive version of the algorithm consistently outperforms the conservative version. As previously observed, the latter suffers in scenarios where there is a lack of available capacity when approaching the deadline. To appreciate the benefit of adaptive schedulers, we compare against the straightforward GT heuristic that uploads all data as fast as possible. On average, it would complete the upload in 33 s, with the cost of 7.5 k units. Adaptive schedulers reduce the cost up to 45%.

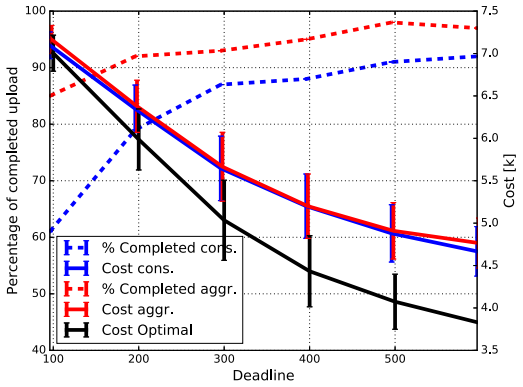


Fig. 5: Percentage of completed uploads and C versus deadline S, with $\beta = 0$ and $\alpha = 0.1$

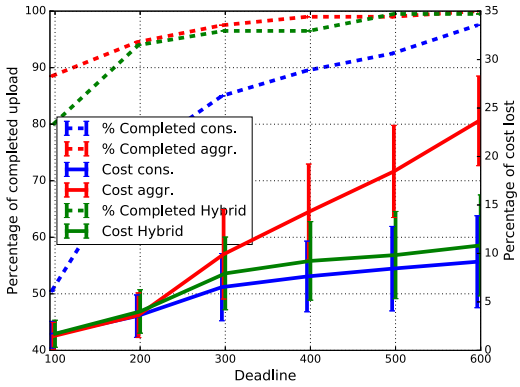


Fig. 6: Percentage of completed uploads and C versus S with $\alpha = 0.1$ and $\beta = 1$ for hybrid recovery

In general, the distance of the cost curves from the oracle case is quite small for very short deadlines, and remains within about 20-25% for longer deadlines. The fact that the difference is small for short deadlines is due to the limited freedom in scheduling. The fact that with longer deadlines the difference remains within acceptable limits is a clear indicator of the good performance of our algorithm. Interestingly, the conservative algorithm provides smaller cost.

C. Hybrid algorithm

The fact that the conservative algorithm offers lower costs, and the aggressive algorithm guarantees higher chances to meet the upload deadline, motivating a hybrid approach. In particular, we use the conservative approach in the initial part of the scheduling, while moving to the aggressive approach when getting closer to the end of the scheduling. Intuitively, at the beginning of the scheduling the algorithm tries to decrease the cost by using a conservative recovery. $0 < \beta < 5$ guarantees to push extra data into the cheap interfaces, avoiding the most expensive one. To then prevent accumulating too much data and taking the risk of missing the deadline, the algorithm switches to the aggressive recovery policy at certain point.

We tested this hybrid approach, by using the conservative recovery for the first 90% of S, and then switching to the aggressive recovery algorithm in the last 10% of S. To better appreciate the cost, we compute the fraction of the additional cost consumed during scheduling with respect the cost of the oracle. Fig. 6 shows that this hybrid recovery achieves very high completion probability with very competitive cost (in the order of 10% higher than the oracle).

VI. Conclusions and Outlook

In this paper we looked at the deadline-constrained upload of video segments from vehicles equipped with a set of wireless network interfaces, possibly provided by different network operators, with multiple technologies, and different costs. The objective of our work is to identify a good set of interfaces to use, and of time slots when to transmit, so as to meet the deadline with minimum cost.

However, since this is normally not true, due to the difficulty in predicting available wireless bandwidth, we devised and studied adaptive algorithms that only require the information about the average available bandwidth.

Our adaptive algorithms, and in particular the hybrid algorithm that combines the two approaches that we proposed to recover whenever an adaptive algorithm falls behind the expected performance, are capable of coming very close to the performance of the (unfeasible) optimum schedule, while requiring only easily acquirable information about wireless bandwidth availability.

References

[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

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

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

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

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

[11] 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.

[12] 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".