Delay tolerant video upload from public vehicles

Delay Tolerant Video Upload from Public Vehicles

Ali Safari Khatouni[1], Marco Ajmone Marsan[1,2], Marco Mellia[1]
[1]Politecnico di Torino, Italy - name.lastname@polito.it
[2]Institute Imdea Networks, Spain

***Abstract*— In this paper we study a surveillance system for public transport vehicles, which is based on the collection of on-board videos, and the upload via wireless transmission to a central security system of video segments corresponding to those cameras and time intervals involved in an accident. We assume that vehicles are connected to several wireless interfaces, provided by different Mobile Network Operators (MNOs), each charging a different cost. Both the cost and the upload rate for each network interface change over time, according to the network load and the position of the vehicle. When a video must be uploaded to the central security, the system has to complete the upload within a deadline, deciding i) which interface(s) to use, ii) when to upload from that interface(s) and iii) at which rate to upload. The goal is to minimize the total cost of the upload, which we assume to be proportional to the data volume being transmitted and to the cost of using a given interface. We formalize the optimization problem and propose greedy heuristics. Results are generated, using real wireless bandwidth traces, showing that one of the proposed greedy heuristics comes very close to the optimal solution.**

***Keywords*—Smart city, public transport, security, video upload, wireless network, scheduling.**

I. Introduction

Smart cities are emerging as an extremely promising and challenging scenario for the application of ICT technologies, and they aim to improve a wide range of aspects of our lives. Among those, particularly relevant are urban transports, security and safety. In this paper we investigate a problem which is at the intersection of these three domains, and relates to the use of wireless communications to monitor public transport vehicles, such as buses, trams or metro trains.

In particular, we study a video surveillance system for public transport vehicles, which is based on the collection of on-board videos and their wireless transmission to a central security system. Our interest is motivated and inspired by the real needs of public transport operators. On public transport vehicles, already now, several video cameras are installed, each producing a video stream with rate from 1 Mb/s to 10 Mb/s. Continuous real-time video streaming from vehicles to the central security system is considered too expensive in data volume and in cost, and largely useless, because nothing relevant happens on the vehicles most of the time. Videos are thus stored on board, and when an alarm is triggered (e.g., when a customer or a driver reports a problem, or after a complaint is filed), the Security Operator (SO) on duty in the central security control station needs to access the portion of the on-board videos which refers to the period of time of the accident.

In traditional systems, videos are uploaded to the central security system when the vehicle enters the depot, where cheap and high-speed wireless connectivity is available. This forces the SO to wait a long time before being able to investigate the accident.

In this paper, we study a novel solution, which provides the SO with near-real-time access to videos corresponding to those cameras and time intervals involved in the accident. We assume that the vehicle is connected to the network by means of different wireless interfaces, through different Mobile Network Operators (MNOs), each charging a different cost, from cheap WiFi, to 3G/4G interfaces, or satellite links. Both the cost and the upload rate for each network interface change over time, according to the network load and the position of the vehicle. We assume that, thanks to the repetitiveness of the public vehicles routes, the system has created a performance map to collect information about the expected network connectivity performance along the route. (The creation of such map is outside the scope of this paper.)

Once the SO requests a video, the system has to complete the upload from the vehicle storage system within a given deadline. The system has to decide i) which interface(s) to use, ii) when to upload from that interface(s), and iii) at which rate to upload. The goal is to minimize the total cost of the upload, which we assume to be proportional to the data volume being transmitted and to the cost of using a given interface. For instance, assume that a video must be uploaded with a deadline of 5 minutes, and that the cost of using a given operator (slow and expensive) 3G interface is higher than the cost of using a (fast and cheap) WiFi interface of a second operator. However, the bus will enter the coverage area of the latter only in 3 minutes. In this context, is it better to wait entering under WiFi coverage, or to start uploading the video now?

The video upload problem can be seen as an optimization problem for which it is possible to obtain different formulations, depending on the assumptions. We consider two different approaches, which lead to two distinct and fundamentally different models. In both cases we assume time is slotted. If a network interface, at each time slot, can be only fully devoted to the upload of a single video, the problem belongs to the *bin packing problem* (BPP) family, which is a well-known combinatorial and NP-hard problem [1]. If, instead, the bandwidth offered by a network interface can be shared between multiple videos, and the upload rate can be controlled freely, the problem can be mapped to a *Minimum Cost Flow Problem* (MCFP) which can be solved very efficiently [2].

The rest of this paper is organized as follows. In Section II we describe the approach we use in the analysis, and we describe a MCFP model, a BPP model, as well as three greedy heuristics; in addition, we show that optimal solutions, even for toy examples, can be tricky, and escape simple greedy approaches. In Section III we briefly discuss related works.

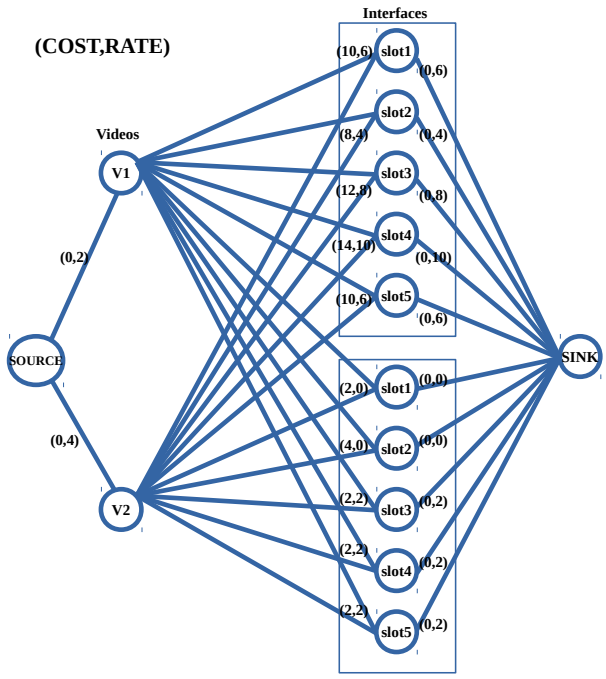


Fig. 1: An example to represent the model.

In Section IV we present some details on the setup considered in the derivation of numerical results. In Section V we present numerical results for realistic cases. Finally, in Section VI we conclude the paper and we discuss future research issues.

II. Model Formulation

We model the scheduling problem using a directed graph $G = (N, E)$, where $N = \{i\}$ is the set of nodes and $E = \{(i, j)\}$ is the set of edges. Referring to Fig. 1, the leftmost node represents the video source, i.e., the vehicle. The second group of nodes represents the video files to be uploaded. Each video k (2 videos in the example) is of volume V_k, and can be uploaded through different interfaces, at different time slots, 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 by a box. The number of available time slots (5 in the example) represents the deadline to meet (recall that we consider slotted time). The rightmost node represents the sink, i.e., the server receiving the videos.

All edges in E have a label containing two values: a cost and a capacity. 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 capacity equal to the video file total size. Each video node is connected by a directed edge to nodes representing a time slot and interface. These edges are characterized by the cost per bit of using such time slot and interface ($c_{i,j}$), and the maximum flow $f_{i,j}$ 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. Each node representing a time slot and interface is connected to the sink with an edge with zero cost, and capacity equal to the time slot capacity.

A. MCFP Model

In this first model, we assume that any interface can be shared between any video at any time slot. We model the problem as a Minimum Cost Flow Problem (MCFP), in which we look for the maximum flow that the network can carry, with the minimum total cost. The objective function in eq. (1) represents the total upload cost, which must be minimized (t is the slot duration). Eq. (2) forces flow conservation constraints. It states that the sum of incoming flows at all nodes (except source and sink) equals the sum of outgoing flows, i.e., flow cannot disappear at intermediate nodes. The flow on every edge is non-negative, and it cannot exceed the rate $r_{i,j}$, see eq. (3). Eq. (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 vehicle.

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

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

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

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

B. BPP Model

With the second model, each interface and time slot can be assigned for transmission to a single video. Variables $f_{i,j}$ become binary variables, equal to 1 if the edge (i, j) is used to transfer data, 0 otherwise. We thus replace eq. (3, 4) with eq. (5, 6) respectively. Since $f_{i,j} \in \{0, 1\}$, only one variable in the sum can take the value 1, i.e., each time slot can be used for the transmission of one video only. Other equations and the objective function are the same as before. This second formulation transforms the problem into a Bin Packing Problem (BPP), which is well-known to be NP-complete, and therefore no polynomial time algorithm can solve it. We leave the analysis and solution of this second problem for future work.

$$\sum_{(i,j)\in E} f_{i,j} \leq 1 \quad \forall j \in N, j \neq Sink \tag{5}$$

$$\sum_{(Source,j)\in E} f_{Source,j} r_{Source,j} \geq \sum_{i} V_i \tag{6}$$

C. Heuristic Approaches

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. In other words, the video with closest deadline is transmitted 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.

TABLE I: Comparing heuristics based on their complexity.

	Time	Capacity	Cost	Complexity
GT	1	-	-	$O(1)$
GC	2	-	1	$O(T*I*N)$
GR	2	1	-	$O(T*I)$

iii) Greedy-in-cost (GC) - This algorithm sorts time slots according to increasing cost, and schedules transmission through the cheapest time slots. If costs are equal, earlier time slots are preferred.

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

Table I shows the priority of the time slot ordering and the complexity of the three greedy heuristics.[1] T is the number of time slots, I is the number of interfaces, and N 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 (which depends on the video and the interface), and then according to time. The GR algorithm needs the ordering according to slot capacity and then time.

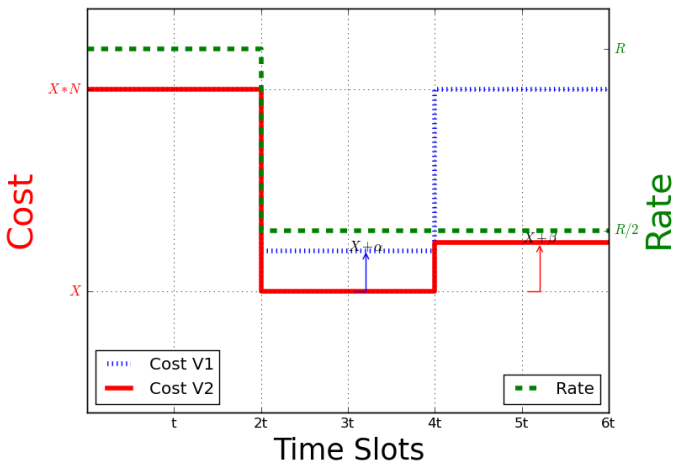


Fig. 2: Example to show how greedy approaches can perform worse than the optimal solution.

In order to show that greedy approaches can produce a solution which has suboptimal cost, we use a very simple example with only one available interface, and 2 videos to be uploaded, of sizes $V_1 = V_2 = Rt$, with the slot costs presented in Fig. 2 with blue dotted and red solid lines, respectively, and slot rates given by the green dashed line. Slots have duration t seconds. The two videos have the same deadline equal to $6t$, and 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 for uploading the two videos, considering the greedy heuristics and the optimal solution. By assuming that $\alpha < \beta$, $(\alpha + \beta) \to 0$, and $N > 2$, we have:

[1]Complexity may be reduced by presorting slots.

1) *Greedy-in-time* - The two videos are uploaded in the first two time slots (either one slot per video, or sharing the slot capacity), without considering the cost of slots. The total cost is $2NXRt$.
2) *Greedy-in-rate* - 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 $2NXRt$.
3) *Greedy-in-cost* - 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 XN 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 $NXRt + 2XRt/2 = (N+1)XRt$.
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 $(X+\alpha)Rt + (X+\beta)Rt = 2XRt + \cancelto{0}{(\alpha+\beta)Rt}$

We conclude that the total cost for Greedy-in-time and Greedy-in-rate is N times higher with respect to the optimal solution. Instead, the cost for Greedy-in-cost is $(N+1)/2$ times higher with respect to the optimal solution. By means of this example we wish to show that greedy algorithms can generate solutions with possibly much higher cost than the optimal solution for the scheduling problem, even in very simple cases.

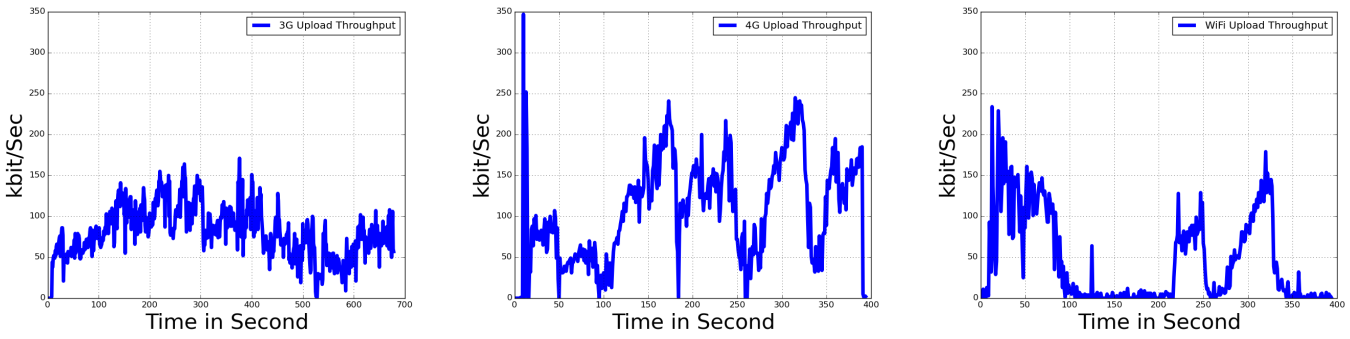
III. Related works

Mobile devices allow users to connect to multiple wireless access networks with possibly different technologies, obtaining throughput values which depend on many factors, such as the user 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, so that several works in this field have been previously published [3].

Stochastic scheduling has been broadly studied in the multi-processor, multi-server domains [4], [5]. This family of works looks at the problem of allocating resources to requests, such as processors to programs, when uncertainties are associated with requests. The network availability in mobile networks brings uncertainty in scheduling the operation of mobile devices with multiple interfaces. This aspect has been investigated recently by several authors. Rahmati et. al. [6] presents a technique for estimating and learning the Wi-Fi network conditions. Rathnayake et. al. [7] demonstrates how a prediction engine is capable of forecasting future network and bandwidth availability, and proposes a utility-based scheduling algorithm which uses the predicted throughput to schedule the data transfer over multiple interfaces.

Other works focused on scheduling under the assumption that network throughput is known. Zaharia et. al. [8] presents a model for the optimal scheduling over multiple network interfaces, and proposes approaches to find the scheduling algorithm which can be implemented with the limited resources available in devices like cellular phones, PDA, etc.

Trace-driven scheduling is a reasonable approach to experiment with algorithms in realistic conditions, since traces provide credible data about the wireless network performance.



(a) 3G (b) 4G (c) WiFi

Fig. 3: Upload throughput based on traces

Riiser et. al. [9] used 3G mobile network traces[2] collected onboard different types of public transport vehicles around the city of Oslo (Norway) to measure the achieved throughput when adaptive HTTP streaming runs in mobile devices equipped with 3G networks. Chen et. al. [10] measured the throughput, as well as several other performance metrics of both single-path and multi-path data transport in cellular networks, examining 3G, 4G, and Wi-Fi networks[3].

Our work differs from previous research in several ways. First, we do not consider stochastic scheduling, because we deal with a trace-driven approach. Second, we consider the delay tolerant upload of videos using different wireless technologies, while previous works mostly focus on download of data while optimizing completion time. Third, our work is different from [8] because they use utility-based scheduling and they assume the cost and capacity of each interface to be constant; instead, in our model we assume that cost and capacity are not constant over time or across interfaces. In addition, their goal is to find a solution which can be implemented directly on the mobile device, while the fact that we work in a near-real-time scenario with deadlines of the order of several minutes, makes our problems solvable by a centralized scheduler.

IV. Experiments

In this section we describe the setup we use to test the performance of the greedy algorithms and compare them against the optimal solution. To use a realistic setting, we adopt a trace-driven approach, based on the traces collected by Chen et al. [10] in 2012. These traces record the upload data rate available to wireless interfaces of different technologies (WiFi, 3G, 4G) on mobile devices in the Boston area. These data are reported in Fig. 3. Each network interface has a different profile, as expected. For instance, as we can see in Fig. 3c, the *WiFi* interface exhibits periods of good coverage interlaced with periods in which the upload data rate is close to zero. On the contrary, the upload data rate offered by the 3G interface is more stable along the whole considered interval. The lengths of traces are 700 s for 3G, and 400 s for both 4G and WiFi. Since public transport vehicles repeatedly follow a fixed path, when a longer time span is necessary, we repeat the traces as many times as necessary to reach the deadline. However, to allow for some randomness, possibly resulting from deviations from the expected behavior, the capacity of each interface at any specific time slot is computed by sampling a Normal distribution with mean and standard deviation equal to the trace value. The case in which prediction errors are accounted for, calls for stochastic programming approaches, but this will be the subject of future work. Since deadlines in real cases are in the order of a few minutes, the solution of the problem through a centralized scheduler seems appropriate. The time slot duration is taken to be 1 second, which should be small enough to account for performance fluctuations. However, in our future work we plan to study the impact of the slot duration. The average video size is chosen to match the deadlines for the upload.

We considered two scenarios: 1) a small problem, with 2 videos to upload, using 3 different network interfaces for 3G, 4G, and WiFi; 2) a large problem, with 5 videos to upload, using 10 different network interfaces in total, with 5, 4, and 1 interfaces of type 3G, 4G, and WiFi, respectively.

Each video is specified by two parameters: its size and its deadline. The size of each video is randomly drawn from a Normal distribution with average equal to 3 MB and standard deviation equal to 1 MB. For experiments, we consider the maximum number of available time slots in [200,1000] or [200,10000] for small and large scenarios. Deadlines are chosen in the last 25% of the available time slots, using a uniform distribution. For example, if we have 200 slots, the deadline is selected uniformly in the range [150,200]. The cost associated with each interface at a given time slot is drawn from a Normal distribution with average 3, 6 and, 9 for WiFi, 3G, and 4G, respectively, and with standard deviation equal to 1 for all types of network interfaces.

We used the *IBM ILOG CPLEX Optimization Studio 12.6.0.0* [11] 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*. We repeat each experiment ten times, and measure the average and the confidence interval for the following metrics: i) time to complete the upload; ii) cost of the upload; and iii) CPU time for the computation of the solution.

Intuitively, we can expect that any algorithm may force the upload completion times to be very close to the deadline e.g., because of a very cheap slot appearing very late. This might not be advisable, since network interfaces could experience throughput values smaller than forecasted, so that the upload is

[2]Traces are publicly available on https://heim.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs/. We thank the authors of [9] for allowing us to use them for some experiments.

[3]We thank the authors of [10] for allowing us to use their traces.

not completed within the time limit. To avoid this, we introduce a penalty function for the use of slots close to the deadline. The expression of the penalty function for slot i and video j is the following:

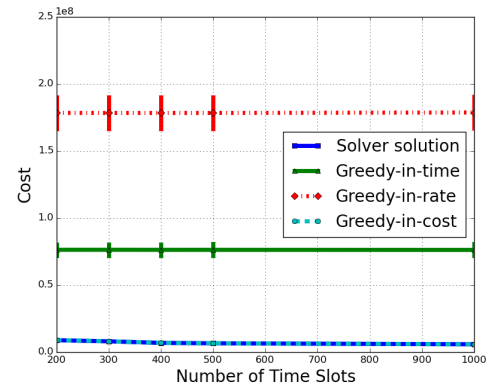
$$cp_{i,j} = c_{i,j} * (P - (D_j - t)/D_j) \qquad \forall t > D_j/2 \qquad (7)$$

where t is the slot index, D_j is the deadline of video j, $c_{i,j}$ is the cost of slot i for video j before the penalty, and $cp_{i,j}$ is the penalized cost of slot i for video j. P is the penalty factor, which we choose in the set $\{1,2,5,10\}$. For $t \leq D_j/2$ the slot costs are not penalized. The penalty function forces the scheduler to find a solution which is more robust to variations from the predicted throughput values. Indeed, it provides a solution that leaves several unused time slots close to the deadline; those slots provide a safety margin, and allow for rescheduling, if necessary.

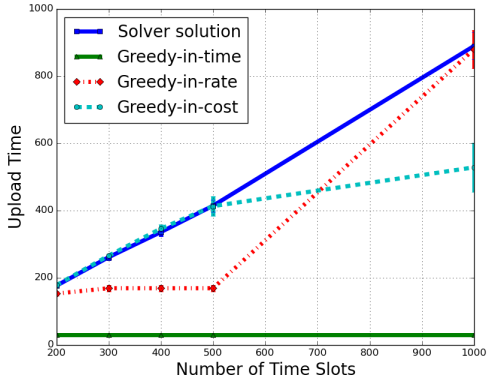
V. Results

In this section we discuss the numerical results obtained for the two scenarios described in Sec. IV.

A. Simple scenario



(a) Cost



(b) Upload Time

Fig. 4: Results for the simple scenario.

The results in Fig. 4 refer to the simpler case, in which 2 videos have to be uploaded via 3 network interfaces. Fig. 4a shows the total cost. As can be expected, the GR algorithm incurs the highest upload cost, followed by GT. This is due to the fact that both those heuristics neglect the cost values. The GC algorithm provides cost values which are almost equal to those of the optimal solution. As previously anticipated, the total upload time in Fig. 4b, shows that the optimal solution produces upload times very close to the deadline, as well as GC when the number of slots is small, and GR when the number of slots is large. On the contrary, GT yields very short upload completion times, as expected.

The two main conclusions that we can draw from these results are: i) that GR is dominated by GT, since the latter provides lower cost and lower completion times, ii) that GC achieves practically the same cost (not lower, of course) as the optimal solution, with completion times similar, or even lower, than the optimal solution.

B. Larger scenario with penalty function

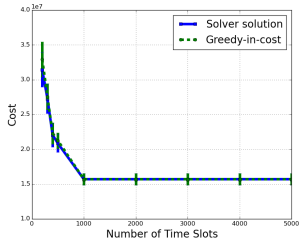
We now consider the case of 5 videos and 10 network interfaces, looking only at the GC algorithm and the optimal solution, but also considering the introduction of the penalty function defined in Sec. IV. Results are presented in Figs. 5, 6, and 7, for the cost, the upload time, and the computation time, respectively. Notice how cost decreases as the number of slots increases (Fig. 5). The cost obtained by the two algorithms is almost the same (of course, values achieved by the optimal solution are never higher than the values obtained by GC). Upload times are often shorter for the GC heuristic. We also see that the introduction of the penalty function is effective in bringing the completion times safely earlier than the deadlines. (Fig. 6a vs. Figs. 6b,c,d)

Finally, we note that the computation times for GC are shorter than those necessary for the solver to compute the optimal solution. These characteristics make GC a very effective algorithm for the generation of simple but effective solutions for the near-real-time video upload problem.

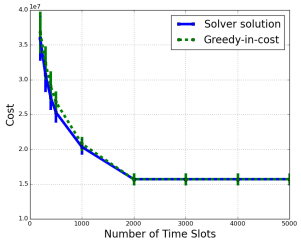
VI. Conclusions and Outlook

In this paper we studied a surveillance system for public transport vehicles, which is based on the near-real-time wireless slotted transmission of videos to a central security system. We assumed that vehicles are connected to several wireless interfaces, with variable, but predictable, cost and rate. When a video must be uploaded, the system has to optimize cost by choosing which interfaces to use, in which slots to upload and at which rate, at the same time meeting a given deadline. Three greedy heuristic were compared to the optimal solution, showing that the greedy algorithm that looks at slot cost provides solutions very close to optimal in terms of cost, and often more advantageous in terms of upload time.

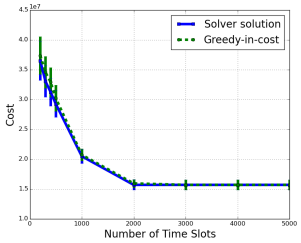
The further steps of the analysis will consider the variable nature of mobile networks, whose performance depends on parameters which cannot be predicted precisely. For instance, social events, congestion, network outages, vehicle changes of path, etc., can affect the actual upload performance. In this case, online algorithms that can dynamically compute the scheduling for the residual upload workload must be devised. Practical issues must also be faced, e.g., the impact of the transport protocol, the choice of video coding approaches, the granularity of time slots, the interference between simultaneous requests, etc. All of these features make this problem quite challenging, and worth further investigation.



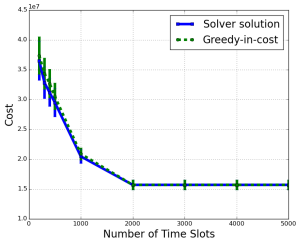
(a) Without penalty



(b) Penalty with factor 2

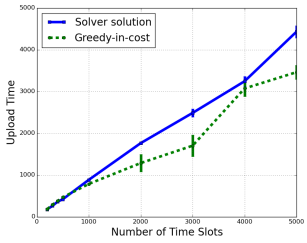


(c) Penalty with factor 5

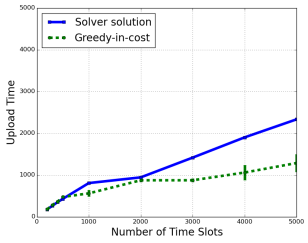


(d) Penalty with factor 10

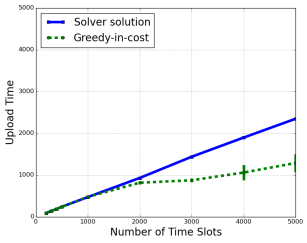
Fig. 5: Cost



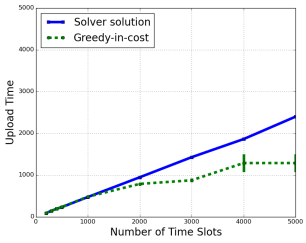
(a) Without penalty



(b) Penalty with factor 2

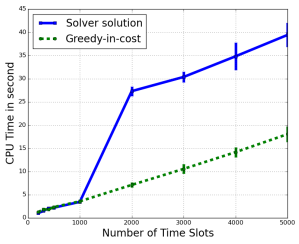


(c) Penalty with factor 5

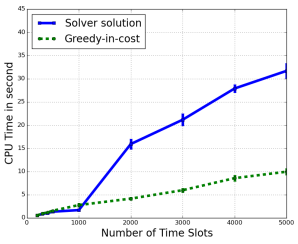


(d) Penalty with factor 10

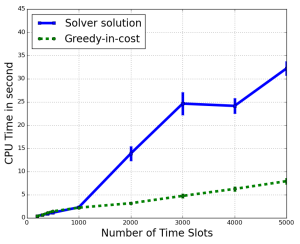
Fig. 6: Upload Time



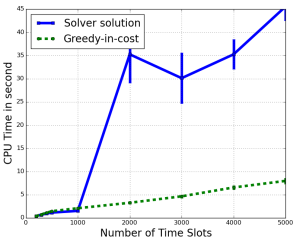
(a) Without penalty



(b) Penalty with factor 2



(c) Penalty with factor 5



(d) Penalty with factor 10

Fig. 7: CPU Time

ACKNOWLEDGEMENTS

This work was funded by the *MONROE* [12] project (grant agreement no. 644399) in the H2020-ICT-11-2014. Computational resources were provided by hpc@polito, which is a project of Academic Computing within the Department of Control and Computer Engineering at the Politecnico di Torino[4].

REFERENCES

[1] Martello, Silvano and Toth, Paolo, *Knapsack Problems: Algorithms and Computer Implementations*. New York, NY, USA: John Wiley & Sons, Inc., 1990.

[2] Ahuja, Ravindra K.; Magnanti, Thomas L. and Orlin, James B., *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[3] Yap, Kok-Kiong; Huang, Te-Yuan; Yiakoumis, Yiannis; Chinchali, Sandeep; McKeown, Nick and Katti, Sachin, "Scheduling packets over multiple interfaces while respecting user preferences," in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '13. New York, NY, USA: ACM, 2013, pp. 109–120.

[4] Yuan, Wanghong and Nahrstedt, Klara, "Energy-efficient soft real-time cpu scheduling for mobile multimedia systems," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 149–163, Oct. 2003.

[5] Al-Zubaidy, Hussein; Lambadaris, Ioannis and Viniotis, Yannis, "Optimal scheduling in multi-server queues with random connectivity and retransmissions," *Comput. Commun.*, vol. 35, no. 13, pp. 1626–1638, Jul. 2012.

[6] Rahmati, Ahmad and Zhong, Lin, "Context-for-wireless: Context-sensitive energy-efficient wireless data transfer," in *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services*, ser. MobiSys '07. New York, NY, USA: ACM, 2007, pp. 165–178.

[7] Rathnayake, Upendra; Petander, Henrik and Ott, Maximilian, "Emune: Architecture for mobile data transfer scheduling with network availability predictions," *Springer US*, 2012-04.

[8] Zaharia, Matei A. and Keshav, Srinivasan, "Fast and optimal scheduling over multiple network interfaces," University of Waterloo, Tech. Rep., 2007.

[9] Riiser, Haakon; Vigmostad, Paul; Griwodz, Carsten and Halvorsen, Pål, "Commute path bandwidth traces from 3g networks: Analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*, ser. MMSys '13. New York, NY, USA: ACM, 2013, pp. 114–118.

[10] Chen, Yung-Chih; Nahum, Erich M.; Gibbens, Richard J.; Towsley, Don and Lim, Yeon-sup, "Characterizing 4g and 3g networks: Supporting mobility with multi-path tcp," UMass Amherst Technical Report, Tech. Rep., 2012.

[11] "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

[12] [Online]. Available: https://www.monroe-project.eu/

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