Practical challenges of network optimized stored video delivery

Foivos Michelinakis
Institute IMDEA Networks, Madrid, Spain
Universidad Carlos III de Madrid (UC3M), Madrid, Spain
Email: foivos.michelinakis@imdea.org

***Abstract*—The scope of this thesis is to investigate the challenges and their possible solutions of applying a theoretical model of optimized stored video delivery to an LTE network. Recent paradigm shifts in content consumption, caused by the ever increasing popularity of online social networks, have allowed researchers to predict future content requests. In addition, models that predict user mobility as well as user activity can be used to infer the capacity that users of radio access networks will have in the future. Our work tries to combine the above, in order to optimize the delivery of video content that is predicted to be consumed by a user, whose future channel capacity is perfectly known, in a way that is transparent to him and require as few radio resources as possible. Thus, both increasing the number of users that can be served in a cell and the quality of service that they enjoy. The simulations have shown that our solution achieves a good balance between robustness and low channel utilization, while significantly reducing the cost compared to the benchmark application.**

***Index Terms*—Pre-fetching, mobile networks, Network Optimization, Content Distribution**

I. Introduction

Since the boom of the "World Wide Web" and up until recently, multimedia consumption over the Internet followed a very simple model. A user would use his desktop machine (or a little bit later, laptop) to connect to a server of a well known content provider (e.g. a big news corporation), browse the available content and finally download his choice directly from that same server, or another server that belonged to the same company. The last few years, emerging new technologies and devices have altered this traditional scheme of media discovery and delivery. The gigantic growth in popularity of online Social Networks (OSNs), Content Distribution Networks (CDNs), video sharing websites, hand-held devices with multimedia capabilities and access to fast 4G/3G networks and finally peer 2 peer applications have shape-shifted the old centralized media dissemination model to a more distributed one, that often makes heavy use of wireless technologies.

In addition, the availability of flat rate Internet access encourages people to watch a vast number of videos online, often in HD quality. Also, an increasing percentage of this content is consumed on handheld devices. Thus, the Internet has transformed into a primary means to access media content. A recent trend, that is expected to continue, is the explosion of the global Internet consumer traffic, mostly caused by the ever increasing video traffic. This phenomenon is even stronger in mobile traffic. According to Cisco, a major network equipment vendor, average smartphone usage grew 81 percent in 2012. The average amount of traffic per smartphone in 2012 was 342 MB per month, up from 189 MB per month in 2011[1]. These numbers are even bigger in countries that have deployed LTE, a recent very fast mobile communication standard. For example in South Korea, the average data consumption of LTE users is already 2.2GB[2]. As a consequence, service providers have to face the huge costs related to updating their network and paying for traffic in order to cope with the demands of their customers. mobile service providers further have to overcome the saturation of their wireless resources in the "last mile". To this end, researchers are focusing on developing techniques that will either reduce traffic or distribute it more evenly throughout the day, reducing the peak demand. The latter is very important since pricing of bandwidth between service providers is based on the peak traffic bandwidth demand (95th percentile[3]). Also, the quality of experience the users enjoy diminishes significantly as the links become saturated (delays, lower quality of content in the case of adaptive bit rate, etc.).

The key concept behind reshaping traffic lies in the fact that a big portion of it can be predicted with relatively good accuracy and thus pre-fetched at a time that minimizes its impact on the network. The widespread adoption of OSNs has drastically changed the way content is consumed in the Internet. Today, a big portion of content is being discovered via OSNs such as Facebook or Twitter, through recommendations and sharing. Content consumption is nowadays highly impacted by the information shared by users through OSNs and the popularity of a given content is most often dictated by its "social" success. The separation of content discovery (OSNs) and content delivery (commercial CDNs, video sharing websites and lately OSNs) allows to obtain timely information regarding by whom, where and when a piece of content will likely be accessed. In the case of popular content, it is possible to obtain fine-grained information about the locality and quantity of traffic this content is expected to generate. Further, it is the only method that can provide some information regarding the future consumption of less popular content, like home videos that are intended to be watched only by close friends of a user.

[1] http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html

[2] http://thenextweb.com/mobile/2013/01/03/lte-3g-data-comparison-mobidia/

[3] http://en.wikipedia.org/wiki/Burstable_billing#95th_percentile

For example, if a link is posted to the wall of an OSN user, it will probably be accessed when that user logs in to that OSN. In the time between the posting of the link and the user log in (i.e., the time when the link is uploaded, but the user is not yet aware of it), the content can be pre-fetched to the users' device or a nearby cache [1].

The present work is a subproblem of an attempt to propose a complete solution that will predict, prefetch to a nearby cache and deliver to the User Equipment (UE)[4] of an LTE network content in a way that is both efficient for the network and transparent to the user. In this paper we focus on how to deliver the multimedia content which is predicted to be requested by a user, by consuming as few resources as possible at the "last mile" (ie. the radio Access Network - RAN). We assume that the content is placed at an entity that belongs to the network operator, so the operator has access to the complete file at the moment when the user requests it and we also assume that we have complete knowledge of the context of the user and the eNodeBs[5] he is associated to. This entity can either be the origin video server, or in case there are prediction and prefetching mechanisms in place, a cache close to the user. The theoretical foundations of the proposed model can be found in [2]. In this work we will try to explore the challenges associated with using the offline approach of that theoretical model in real scenarios via simulations, analyzing them and finally applying and testing modifications that would make it viable in practice.

The contributions of this paper are as follows: first we analyze the practical problems that arise when we attempt to use the theoretical model of [2] and discuss possible solutions for each problem. Second, we propose the modifications that have to be made to both the model and the LTE architecture in order to ensure its robust application in practice.

The remainder of this paper is structured as follows. Section II discusses the state of the art. In section III, we describe our proposed architecture. Section IV presents the formal statement of the problem we are attempting to solve. Section V describes the remarks we made through our simulation testing of the theoretical algorithm, our proposed modifications to it, as well as the validation, through simulations, that our solution functions as intended. Finally, section VI concludes the present work and gives elements for future directions.

II. State of the Art approaches to Traffic Optimization & Prediction

In this section we will try to present a selection of the different approaches to network optimization in the "last mile". There is a surprising big variety in the techniques utilized to achieve the same goal and each one of them seem to perform better than others in certain scenarios and vice versa.

The most notable approaches that take advantage of the buffer of the video application are [2] and [3]. Both papers try to optimize stored video delivery (the whole file is available at the start of the playback, in contrast to live video) and assume that the future channel capacity allocated to the user is accurately known. They aim at transferring the video by utilizing as few radio resources as possible and since channel capacity is known they achieve it by sending more data to the UE when it has good channel conditions and avoid sending data to it when it has unfavorable channel conditions (being at the edge of the cell with low SINR, performing handover etc.). The amount of radio resources required to send the same amount of data under good conditions is significantly lower, compared to the bad ones, because the UE can make use of Modulation and Coding Schemes (MCS) that are more efficient (than robust). The abundant data sent during the good channel conditions are stored in the buffer of the video application, so the main constraint of this approach is the finite size of the buffer.

The following four approaches try to perform traffic tailoring in order to avoid non-necessary transitions of the UE to resource utilizing states and non-necessary data transfers.

Sanadhya et al. [4] propose the use of network deduplication with asymmetric caching in order to reduce unencrypted redundant traffic at the bottlenecked link between the UE and the serving GPRS support node. Their model, which is based on layer 2.5, has a network cache intercept the downlink traffic of the UE, segment it and compute the hash of each segment. If the hash identifies data that are present at the cache of the UE, the network cache just forwards the hash of the segment instead of the actual data.

Quian et al. [5] explore methods that can optimize periodic transfers of cell phones. A lot of applications send periodic updates to servers, which usually do not transfer any useful data, but force the UE to get to and remain in a resource utilizing state, without improving the user experience. Their proposed methods include piggybacking periodic transfers to non periodic transfers, batching and fast dormancy, which forces the UE to release resources earlier that what is dictated by its state machine.

Huang et al.[6] suppose that the traffic generated while the screen is off is much more delay tolerant than the traffic generated while the screen is on, due to the lack of user interaction. This enables them to apply more aggressively traffic optimization techniques, like fast dormancy and batching, without significantly affecting the user experience.

Han et al. [7] propose the model of "mobile social networks", which are the combination of traditional online social networks and opportunistic networks, in order to explore content dissemination opportunities. They focus on free opportunistic networking for the delivery of non time critical data, like weather forecasts, in order to offload the cellular network. The mobile operators would send data to just a subset of subscribers who would in turn propagate the information to the rest of the users. Once a user receives the data, he can further propagate them to other users. If a subscriber has not received data after a time threshold, the operator transmits them directly to him. The initial set of users who get the data first is selected based on past mobility traces and an information dissemination function. In [8], they go on to implement the above theoretical platform. For the

[4]LTE terminology that describes the mobile device that a user is using in order to enjoy high speed mobile Internet

[5]LTE terminology that describes the base station part of the LTE architecture

opportunistic implementation they use Bluetooth in order to avoid the extensive battery drainage that accompanies WiFi.

These last approaches explore the feasibility of traffic offloading to WiFi Access Points.

Balasubramanian et al. [9] propose a model which both A) predicts when the UE will have WiFi connectivity and delays traffic that can be classified as delay tolerant, in order to be offloaded to the WiFi interface and B) switches fast to the 3G interface in case WiFi is unable to meet the delay requirements. The estimation of when WiFi connectivity will be available again is based on the inter-meeting time of the previous N encounters. The fast switching is being achieved by utilizing low-level link layer information, which though causes a slight increase in the 3G load.

Ristanovic et al. [10] use the "call detail records" of mobile operators for each individual user to predict his mobility, based on the fact that, especially during week days, users tend to visit the same cells at the same time. based on that mobility prediction, delay tolerant traffic can be offloaded by either using opportunistic means (other devices at the area of the cell that have the requested data), or by using WiFi APs located at the most popular cells.

Siris et al. [11] optimize traffic offloading in a vehicular scenario. They take advantage of the GPS sensors present in all modern smartphones in order to send geolocation information to a server. This information combined with real time road traffic data, enable the prediction of the future positions of the UE as well as the amount of time the UE will stay in each position. This enables their platform to pre-fetch traffic to WiFi APs along the predicted path, which will deliver the data when the UE comes within range.

Finally, in [12], authors attempt to extend the traditional WiFi offloading schemes by taking into account the actual quality of the WiFi APs that a mobile is estimated to encounter in regard to throughput and energy consumption. If it is predicted that the performance of the WiFi will be significantly lower than that of 3G, the WiFi link is not preferred.

III. Architecture

In this section we will provide a high level overview of our proposed architecture. Our architecture is based on the LTE communication standard, but can be easily extended to any other mobile communication standard. We made the choice to focus on LTE, because the RRC states of the various standards are a bit different and we would like to make our approach future proof. The implications of the different RRC states will not be discussed in the present paper, but are important to the bigger project that the current work is part of.

In brief, the main components of the core network of LTE are:

- **Packet Gateway (PGW).** It is the connection point between the Internet and the network of the operator.
- **Serving Gateway (SGW).** It is the regional entity of LTE that forwards data packets to the eNodeBs where the destination UEs are. It does so by keeping track of the mobility of the UEs that are associated to the eNodeBs that it is connected to. Also, it is the anchor point of the UEs who perform handover between two eNodeBs that it is connected to. It is worth noting that the majority of the handovers takes place between eNodeBs that belong to the same SGW.
- **eNodeB.** It is the base station of the LTE network. The SGW forwards to it the packets that are destined to the UEs that are connected to it. Upon reception of a packet, it is temporarily stored and a scheduler regulates when it will be transmitted to the target UE. Some schedulers try to achieve a good tradeoff between fairness and optimal allocation of resources from a network perspective. In the sequel we assume that such a scheduler is being used.
- **UE.** The mobile device of the user (for simplicity, in our scenarios we will assume that the only application running that consumes data is a video application).

As we have stated before, we assume that we have an accurate prediction of a video that a user is about to consume and have prefetched it to a cache close to him. Given the present LTE architecture, a good potential location for storing the video is the SGW. It is the closest point to the UE where we can add storage capabilities and also in case the UE performs a handover during the transmission of the video, the data path will not have to change significantly, because the SGW will just forward the data of to the new eNodeB. Since SGW is aware of the mobility of the UE, it can stop sending data to the old eNodeB at just the right time and start sending them to the new eNodeB, when the UE will have finalized the handover. That way we avoid the problem of having sent packets to the old base station, where we would have to either drop them and transmit them again to the new base station, in case of a seamless handover, or forward them to the new base station from the old one, adding a bit of delay, in case of a lossless handover.

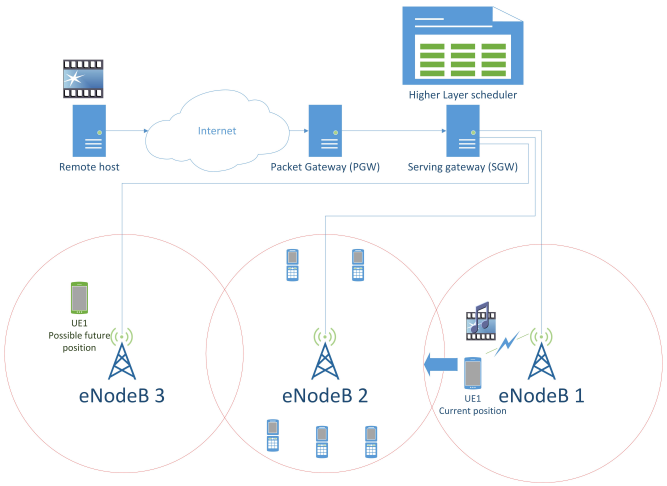


Fig. 1: Example of how mobility and interest prediction can enhance the utilization of the finite resources of radio access networks.

To this end we add a higher layer scheduler to the SGW. The perfect knowledge of the future location and channel conditions of the UE, as well as of the other UEs within its vicinity can allow it to make optimal decisions regarding when

to transmit data. The mobility, maximum channel capacity that can be allocated to the UE at any given time, the buffer size of its video application and the deadlines for the delivery of each data segment that is part of the video will be provided as input to the higher layer scheduler. based on these data, it will provide the rates with which the video should be transmitted in specific time intervals, in order to achieve better fairness, higher utilization of the radio resources and at the same time guaranteeing a good user experience.

For example, let's investigate the simple scenario presented in figure 1. Initially, we prefetch the video to the SGW from the origin video server, before the user requests it. When the user requests it, we assume that he is in a vehicle that has a trajectory that passes across several base stations. Note that the first and the third base station have very few users, whereas the second one is congested. The higher layer scheduler is aware of the context of all users, so it will try to avoid sending data to the UE while it is at eNodeB 2, because the UE will both have to face bigger competition in the resource allocation procedure, which might cause insufficient rate to retain smooth playback and will also deprive the static users of eNodeB 2 from resources.

Moreover, note that two handovers will take place as it moves to different cells. A context oblivious sender, would send Constant bitrate Traffic (CBR), throughout the duration of the video, with possibly a bit of buffering at the beginning. The result would probably be long pauses during his traversing of the second cell. His actual rate would drop significantly right after the handover, because the handover itself consists of 20 ms where almost no data can be delivered at the UE and then the UE would appear at the edge (at best average channel conditions), of a congested cell, thus most probably having an actual rate below the rate of the application. It is possible that his rate could improve as he moves towards the center of the cell, because of the more efficient modulation and coding scheme that would be used, but then he would face another drop in the rate as he would move towards the other edge of the cell and eventually perform another handover. After the handover, he would enter the third uncongested cell where his allocated rate would again be able to support the uninterrupted playback of the video.

On the other hand, our approach would try to fill up the buffer as much as possible before the first handover. Then, after the handover, the UE would rely on the buffer to provide uninterrupted playback. When the UE would approach the center of cell 2 and depending on whether the buffer has enough data to support playback until the next handover, some data might be sent to the UE in order to guarantee smooth playback. From this point and until the next handover no video data would be sent to the UE. After the next handover is over, the data transmission would be identical to the CBR transmission. As we will present in the sequel, the resource savings of applying this approach are significant.

The main weak point of this approach is the possibility of a handover between eNodeBs that belong to different SGWs. Although it remains a possibility, when designing an LTE network, care is taken to ensure that this kind of handovers are very few, so for the time being we will not focus on such events.

IV. Formal statement of the problem and algorithm (analytic considerations)

Throughout this work, we will assume perfect knowledge of the channel capacity that can be allocated to a user in the near future. Essentially, the channel capacity that a user can enjoy is related to the following parameters:

- **User position in the cell and speed.** The Channel Quality Indicators (CQI), like RSRP (Reference Signal Received Power), RSRQ (Reference Signal Received Quality) and SINR (Signal to Interference & Noise Ratio), depend heavily on the location of the user. based on the CQI, LTE subsequently determines which Modulation and Coding Scheme (MCS) achieves the best trade-off between efficiency and robustness[6].
- **Presence and activity of other users in the cell.** The amount of radio resources of the eNodeB are finite. Thus, if a base station does not have enough capacity to serve all the users (it is congested) at the rate that they initially request, then all of them are allocated a fraction of the available resources based on a fairness scheme. The scheduler of the base station determines the fairness scheme and for the rest of the paper we will use the proportional fairness scheme[7]. This scheme tries to maintain a good balance between the overall throughput of the cell and a minimum service guarantee for all the users.

Thus, if one can predict the future values of the above, it is trivial to predict the per user maximum allocated capacity. As presented in section II, there are various techniques that can predict the mobility and activity of the users, so it is possible to have a good estimation for the near future capacity within an acceptable error margin. For the remainder of this work, this scenario will be called online case. In the present work, we will investigate though the simplified offline case, where we assume the future capacity of each user is perfectly known. The future is divided into slots of fixed time duration and we assume knowledge of the average capacity allocation of every user per slot. In [2], the time duration of a slot is set to 1 second.

Usually, capacity is measured in bits per second, but in the case of the LTE radio access network, a more representative unit would be the amount of radio resources required to carry the data (measured in bits) transmitted to the users. In LTE, radio frequencies are organized in groups of 14, called resource blocks and resource blocks in turn are organized in groups of varying size (2, 3 or 4), forming a unit called Resource Block Group (RBG). RBGs are the quantum of resources that can be allocated to the users. The amount of bytes that each RBG can "carry" depends on the modulation and coding scheme used for it. Thus, if we allocate resources to a user when he has good channel conditions and avoid

[6]The mapping of CQI to MCS in LTE can be found at table 7.1.7.1-1 of 3GPP TS 36.213 V9.2.0 (2010-06) found at http://www.3gpp.org/ftp/Specs/html-info/36213.htm

[7]http://en.wikipedia.org/wiki/Proportionally_fair

allocating him resources when he has bad channel conditions we will be able to transfer the same amount of data (bits), using significantly fewer RBGs. Minimizing the number of RBGs per user can boost the number of users being served in a congested cell. Our goal is to transfer as much data as possible to the UEs when they are enjoying good conditions and rely on the buffer to guarantee uninterrupted video playback when they are facing bad channel conditions. When a proportional fairness scheme is used in a congested LTE cell (all users are generating saturation traffic, so they are allocated less rate than they request), we can infer the channel quality of each user by comparing the size of the Transport blocks (TB)[8]. A big TB leads to big capacity and reflects good channel conditions. A small TB leads to low capacity and reflects bad channel conditions.

Based on that observation we will propose a practical implementation of [2], aiming at delivering stored video, using as few radio resources as possible. The objective of [2] is to deliver stored video by transmitting data, only when the capacity of the user is above a certain threshold and rely on the buffer otherwise. This objective can be mathematically expressed as the minimization of $\frac{1}{T}\int_0^\infty \frac{r(t)}{c(t)}dt$, where $c(t)$ is the capacity of the user, $r(t)$ is the allocated rate to that user by the algorithm and T is the duration of the video. It is further assumed that $r(t) = 0\ \forall t > T$, thus the integral is finite.

Modern video codecs express frames in relation to their neighboring frames. This is reflected in the manner in which video applications request data from the buffer. Instead of requesting the data on a per frame basis, they request data that refer to a group of frames at a time. Thus, the video application requests data from its buffer a limited number of times every second. In the rest of the paper we will assume that the period between two consecutive requests for data is 100ms. In order to ensure an uninterrupted playback experience all the related data of a group of frames, should be present to the buffer before the video application requests them. If this requirement is not fulfilled, then the playback has to stop until the data arrive in the buffer causing "rebuffering delay". At this point we only investigate scenarios where uninterrupted video delivery is possible (the rate allocation algorithm can provide a feasible solution that does not require rebuffering). If the algorithm can not provide such solution, we assume that the algorithm has failed. In order to force this limitation as well as ensure that the UE does not receive more data than it can store at the buffer the following constraints are defined in [2]:

- *Definition 4.1 (Low constraint):* At any time t, the cumulative amount of data delivered to the buffer of the video application (symbolized β) should be at least equal to the amount of data that the video application has requested up to this point. In addition, the video delivery process should have finished before the last data request.
- *Definition 4.2 (High constraint):* At any time t, the cumulative amount of data delivered to the buffer of the video application (β) should be at most equal to the the amount of data that the video application has requested up to this point plus the size of the buffer at time t.

In the vanilla version of the algorithm these constraints are checked only at times τ, when either the buffer changes size and/or the video application requests data from the buffer. This discrete time approach to the problem holds true in all the steps of the algorithm, even though the process of video transmission is continuous. As we will analyze in the sequel this is one of the main reasons the vanilla version presented in [2] is impossible to be applied in real scenarios. Another serious issue that is visible at this point is that the various variables have different time granularity (the capabilities are calculated on a per second basis, whereas the constrains are calculated on a per 100ms basis).

A rather superficial overview of how the algorithm functions follows:

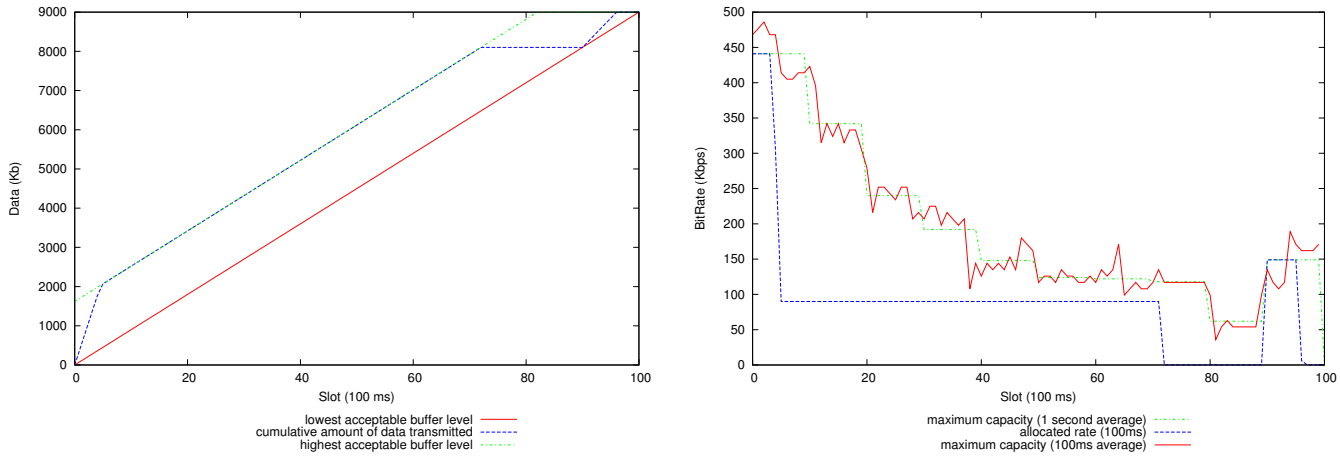
1) Define a time interval that starts at the time of the beginning of the video and ends at the time of the end of the video.
2) Calculate the average future capacity of a user per 1 second slot.
3) Sort the slots according to capacity in a descending order. If two slots have the same value sort first the one that appears earliest.
4) Start allocating rates from the sorted slot list equal to the capacities, until the sum of the transmitted data is at least equal to a target. The assigned rate of all the slots is equal to the capacity, with the exception of the last slot allocated where its rate is equal to the size of the video minus the sum of the capacities of all the previously allocated slots. In the first iteration of the algorithm the target is equal to the size of the video.
5) Check if there are any violations of low and high constraints. If there is no violation, the rates of the previous step represent the optimal feasible solution, for the time interval under examination. If this time interval ends at the end of the video stop and in the opposite case, run again the algorithm starting from the time point t, where the present time interval ended.
6) Given that there are violations of low and high constraints, identify the type of the first violation that occurs and and find the first time point τ, where both constraints are again satisfied.
7) Run again the algorithm from step 3 starting from the beginning of the current time interval until τ and setting a new target of either an empty buffer at τ, in case the low constraint was violated (buffer underflow), or a full buffer at τ, in case the high constraint was violated (buffer overflow).

If the algorithm can not provide a feasible solution for the whole duration of the video for the given constraints we claim that it has failed.

A. A first analysis of the algorithm

At this section we will have a first discussion of some of the reasons that make the current version of the algorithm

[8]LTE terminology to refer to the packets transmitted at the MAC layer of LTE



(a) Buffer (b) Bitrate

Fig. 2: Example run of the vanilla optimization algorithm

impossible to be applied as it is. In order to better illustrate our arguments we have included the output of an example run of the algorithm over a 10 second video with a bitrate of 900kbps, shown in figure 2. In this example we have also assumed that the size of the buffer is constant. At any time instance τ, the value of the low constraint is equal to the amount of data that should have been received, in order to have an uninterrupted playback up to τ and an empty buffer at τ. On the other hand, the value of the high constraint is equal to the amount of data that should have been received, in order to have an uninterrupted playback up to τ and a full buffer at τ. Thus, both constraints any time instance τ, differ by the size of the buffer and as a consequence change at exactly the same time (when a group of packets is removed from the buffer). So whenever a group of frames is consumed, causing the low constraint to jump, the high constraint jumps by the same amount. Figure 2 (a) shows how the cumulative amount of transmitted data increases over time as well as the values of the constraints for the different time instances. In order to have a feasible solution the curve of the cumulative amount of transmitted data should always be within the area defined by the curves of the restrictions. Figure 2 (b), compares the computed 1 second average capacity, the actual rate, and the allocated rate for all the 100ms slots.

The first obvious issue is the inconsistency in the duration of the time intervals over which the different variables are measured. The capacity is measured on a per second basis and the constraints are checked on a per 100ms basis. Since the algorithm checks the constraints on a period equal to their granularity and the β is derived by the allocated rates assigned up to time instance of the check, which in turn is derived by the capacity, we had to divide the 1 second capacity slots to ten consecutive slots of equal capacity and duration of 100ms, in order to make this check possible. This has the effect that step 3 of the algorithm does not change the sequence of the slots, because they have the same capacity and the ones that are earliest are allocated higher in the sorting. In case, the solution dictates that all 10 slots should be utilized, this fix does not pose a problem, because the final allocation would be the same as if we had not make any modifications. If, though the algorithm provides a solution which utilizes some of the slots, because always the earliest slots are ranked higher, the first part of the 1 second slot will be fully utilized and the rest will not be utilized at all, even though if equally spreading the overall target rate for that one second slot to the 100ms slots would provide the same value to the objective function that we want to minimize ($\frac{1}{T}\int_0^\infty \frac{r(t)}{c(t)}dt$). This could be the safe choice in the online scenario, because the prediction of the capacities of the earliest slots is supposed to be more accurate and therefore more trustworthy, but it proves to be too drastic in the offline case. Sending a lot of data in bursts at the beginning of the one second slot, instead of evenly spreading them can cause loss of data, because the actual rate at the beginning of the slot could be significantly lower than the computed one second average. An example of this issue is visible at the last 1 second slot of 2 (b) (from slot 90 to slot 100). Here, the real capacities at the beginning of the slot are significantly lower than at the end of it. This could be caused by the handover of a competing UE to another cell at the middle of the slot, or because the UE under examination moved to a better position which could support a more efficient MCS. In this case, the first 4 slots are allocated more rate than the actual capacity and the last four slots are underutilized, even though their real rate is well above the average. If the difference in capabilities is not significant, the excess data can be stored in the buffer of the eNodeB and transmitted later while when the algorithm does not allocate any rate, or the rate that it allocates is lower the the real one. In case, though the difference is significant, the excess data could cause a buffer overflow at the eNodeB, compromising the on time delivery of the lost data. Especially, in our current approach, where we have not yet implemented retransmission functionality[9], loss of a single packet could cause unpredictable behavior. Retransmissions, utilize capacity that the algorithm assumes is available for the

[9]This functionality is neither included in [2]

normal transmissions, thus the effects of it can range from negligible (a few ms of delay in the delivery of a packet that is not essential to arrive soon) to severe (long delays of groups of subsequent packets, eventually causing buffer underflows). As a final remark on this issue, the authors of [2] assume slow variations in the channel capacity. Therefore the effects of this behavior are significantly diminished, because the values that make up the average do not differ much, but it still remains an open problem.

Another issue that becomes clear is that the constraints are only checked at certain time instances, instead of guaranteeing that they are always valid. Suppose the case where solution dictates that the video application should have a full buffer at two time instances t_i and t_{i+2}, separated by time instance t_{i+1}, where data are removed from the buffer. It is possible that the algorithm will allocate rate to the UE right after the moment t_i, even though the buffer is already full and no data has been removed from the buffer by the video application. The new data would cause a buffer overflow at a time $\tau \in (t_i, t_{i+1})$. This issue is obvious throughout figure 2 (a). Because we have assumed that the constraints change value at the same time $t_{i+1} = t_{i+2}$, so the buffer overflow can happen even if just a single packet is delivered before the next jump point at t_{i+1}. Since packets are transmitted over time, the only way to reach the target cumulative amount of transmitted data is for t_{i+1} is to transmit packets in the time interval (t_i, t_{i+1}), causing the overflow.

As hinted in the first argument of this section, capacity is taken for granted that is equal to goodput. In real scenarios, where data are being lost or corrupt, care should be taken for including the channel utilization caused by retransmissions at the input of the algorithm. A possible solution would be to calculate an error rate and reduce the capacity matrix that is given as an input to the algorithm by that percentage. This though is beyond the scope of this thesis, so we will not discuss it further.

V. Simulation testing of the rate allocation algorithm

A. Simulation setup

After we implemented the rate allocation algorithm we set up simulations in order to identify possible problems in its applicability in real scenarios. We used a modified version of ns-3.18 simulator[10] and more specifically the LTE module of NS-3 called LENA[11][12].

Instead of placing the higher layer scheduler at the SGW, for simplicity we placed it at a server in the Internet. Then we set the capacity of all the links between the server and the eNodeBs to very high values (practically infinite) and their transmission delay to 0ms. This had the effect that the delay between the transmission of IP packets from the server and their appearance at the eNodeB to be 1ms, which we assume is equivalent of transmitting the data directly from the SGW. A visualization of an example topology can be seen at figure 1. Further, in all simulations we used UDP traffic, because we wanted to test the feasibility of the delivery of the IP packets based on the different allocated data rates, free from the often unexpected behavior of TCP.

Our workflow was the following:

1) We set up a scenario in the NS-3 simulator. The scenario includes various static and moving UEs, some of them performing handovers to neighboring eNodeBs. Then we run the scenario setting all the participating UEs to receive saturation traffic. That way we can infer the worst case maximum capacity for any given time for all the UEs. It is important to note that in order to have consistent results with the rest of the simulations of the workflow, we set the size of the transmitted IP packet in this saturation scenario equal to the size of the IP packet transmitted by the video application. We set the size of the IP packet to be 1125 bytes.
2) We collect the traces of the TBs transmitted from the eNodeBs to the UEs, as well as the IP packets delivered to the buffer of the video application of the UEs.
3) Based on the above traces we calculate the 1 second average capacities of all the participating UEs.
4) Next we follow the approach of the evaluation section of [2]. We feed these capacities to the optimal rate allocation algorithm and we assume that the video that will be transmitted has a 10 second duration and its bitrate is 900kbps (we set the video application to request 90kb of data every 100ms). Thus the constraints change value every 100ms, causing final time duration of the slots in which the algorithm is run to be 100ms. We leave as a variable the size of the buffer of the UE video application. The tested values for the buffer size are 1620kb, 1890kb, 2160kb, 2430kb and 2700kb. Since the simulation is only 10 seconds, the more interesting value for the buffer turned to be the smallest of the above (1620kb). By using this relatively small buffer the problems of the real life application of the algorithm become more evident. So, in the simulations that will be discussed below the value of the buffer is set to 1620kb.
5) If the algorithm fails for one UE, we assume that it is impossible to fulfill the objectives of the experiment and is therefore ignored.
6) If it succeeds, we first calculate the value of $\sum_{i=0}^{99} \frac{r_i(t)}{c_i(t)}$[13], where $r_i(t)$ is the allocated rate for the slot $[t_i, t_{i+1}]$ and $c_i(t)$ is the corresponding capacity, in order to theoretically calculate the cost of this allocation.
7) For every UE that has a successful allocation, we set up a simulation in order to test whether it can unproblematically receive the video. We use the same scenario

[10] http://www.nsnam.org/

[11] http://lena.cttc.es/manual/

[12] The main difference between our modified version and the default version of the simulator is that the proportionally fair scheduler located at the eNodeBs takes into account the whether the UEs that are competing for a RBG have any more data to transmit. In case they do not have any more data destined for them at the related buffers of the eNodeB, they are automatically excluded by the competition for the remaining RBGs of that scheduling period contrary to the default behavior of the simulator which would continue to allocate resources to UE that have nothing more to receive, thus wasting them.

[13] This sum is the equivalent of $\frac{1}{T}\int_0^\infty \frac{r(t)}{c(t)}dt$, in the discrete time. Also, because all the videos tested have the same time duration of $T = 10$ seconds, we can omit the factor $\frac{1}{T}$

as in the first step, and we assign saturation traffic to all the UEs but the target UE. The target UE receives traffic from a video server (in the theoretic approach of our architecture the SGW has this role). This server changes its rate according to the allocation.

8) In the majority of the simulations the UE fails to receive all the IP packets. In such cases, we analyze the traces in order to identify the problems, which are discussed in detail in the next section. Further, we keep track of the utilized RBGs.
9) Finally, for benchmarking purposes we also run the same simulation but we set the video server to to send constant bitrate traffic (CBR) of 900kbps (an IP packet of 1125 bytes is transmitted every 10ms) and we record the number of received RBGs.
10) We calculate the gains of the application against the CBR benchmark, both theoretically by comparing the values of $\sum_{i=0}^{99} \frac{r_i(t)}{c_i(t)}$ of the two simulation runs and practically by comparing the number of received RBGs over time.

B. Practical problems and proposed solutions

In this section we will analyze the problems that we encountered during the practical application of the rate allocation algorithm, as well as propose possible solutions for each one of them.

1) General problems of the optimization algorithm:

a) Buffer overflows between two consecutive states that have as a target a full buffer: As we had predicted in section IV.A, buffer overflows did occur between consecutive time intervals that had as a target a full buffer. Since, the difference between two consecutive target states is exactly one group of frames (the amount of data that the video application requests from the buffer), it is possible to overcome this problem by setting the constraint of the maximum data received at any given time, equal to $\{\beta$ + [size of the buffer] - [size of a group of frames]$\}$. Subsequent simulations proved this approach to be adequate.

b) Averaging the capacity over 1 second time intervals: Our analysis of section IV.A was validated by the simulations. Averaging the capacity over 1 second is not accurate. At the saturation run (which gives the the worst case maximum rate), we follow the recommendation of the algorithm and average the capacity over 1 second. Then in order to be able to run the algorithm at 100ms slots we assume that the 10 consecutive 100ms slots that make a 1-second slot have the same maximum capacity. In reality, when we have to deal with moving nodes and/or changing number of users in a cell and/or users who belong in the same cell but start consuming traffic in random times, the average maximum capacities of the 100 ms slots can deviate a lot from the average (in a lot of cases more than 10% difference). Also because the the optimization algorithm has a clear preference in utilizing the earliest slots (given that they have the same capacity, but this is the case in all consecutive 100ms slots of the 1 second slot), in a lot of occasions the algorithm tries to send data at the beginning of 1 second period, fully utilizing the first 100ms slots and then not using at all the last ones. This is a safe choice from the perspective of the algorithm, since the near future is supposed to be more accurately predicted than the far future, even in the case where the difference is just a some hundreds of ms. In reality, this can cause significant problems.

- In the case where good capacities are present at the beginning of a slot and bad later on (as it is the case with a car that is moving away from a base station-see first 1 second period in figure 2 (b), or when a handover takes place to a congested cell). One possibility is that only the first slots will need to be utilized (because they are so good as it is the case in the figure that fully utilizing a couple of them is enough to reach full buffer), where no problem will surface[14]. Another possibility, is that it is necessary to send data at all (or most of) the slots, including the bad ones. In this case, the first good slots are not fully utilized and at the same time the algorithm tries to send at a higher rate than they can support at the bad ones. Apart from the fact that the algorithm provides a wrong estimation of the score (the good slots are not fully utilized and a better denominator is assumed for the bad ones), there is also the the risk of losing packets, because the excess packets that are sent during the bad slots are accumulated at a (very small) buffer at the eNodeB. If the actual rate of the first bad slots (first 1-2 bad slots) does not deviate much from the average and subsequent slots have higher than average capacity and/or are slots the algorithm chooses to under-utilize (according to its false belief of their actual capacity) or not utilize at all, then it might be possible to buffer the excess packets that are sent during the fully utilized bad slots to the buffer of the eNodeB and have them transmitted when the rate given by the optimization algorithm is lower than the actual rate. It should be noted though that the buffers at eNodeBs are really small and in very few occasions are able to provide this service. If a new packet arrives when the buffer is full, it is dropped and will have to be retransmitted. As discussed before, retransmissions can potentially cause significant delays which in combination to the very strict limits set by the algorithm regarding the amount of data that should have been delivered to the UE at any given time instance, can cause the algorithm to miss a target buffer state, thus greatly affecting its performance.
- In the opposite case (where the good capacities are present at the end of the slot and the bad ones at the beginning), the same buffer issues come to the surface. If the eNodeB buffer is big enough to handle the difference then usually the packets will be delivered over time. If the buffer is not big enough, then the failure described above takes place. In addition, if the capacity is really low, it is possible that a buffer underflow will occur at

[14]Apart from the fact that the cost function of the algorithm will mistakenly give a higher score for these rates, because the denominator of the score is the average rate, which is lower than the actual rate, and the fact that we do not use the slots at their full capacity, achieving even more gains, because we assume that the full rate is the average capacity, so we send at this rate instead of the maximum capacity.

this point. The buffer underflow will occur because the algorithm will allocate just enough capacity to have an empty buffer (low constraint), but it will not able to reach that state.

Some possible solutions are:

- **Increase the buffer at the eNodeB.** The LTE specifications do not indicate the size of the buffer that eNodeBs should dedicate to the UEs, so it is up to the operator to set it. Actually, the size of these buffers is set in an indirect way[15], because they are meant to store packets until they are acknowledged. But the time out time of the acknowledgement is determined by the operator. The combination of the value of the above timer and some other parameters (which can not be fully affected by the operator) dictate the value of the buffer. Most vendors seem to choose a conservative approach and set the value of the buffer to be very small, close to the value of dictated by the above. As a consequence, the size of that buffer is designed only to support specific network operations and is not meant to be used as a temporary storage of packets in periods of congestion. Maybe a different and independent buffer to the one described, dedicated to alleviating the effects of congestion would make more sense in this context. Later, we will try to calculate the size of that buffer by means of simulation, but a first estimation of its upper limit can be derived by using "back of the envelope calculations". The size depends on how big is the deviation of the actual capacity from the average capacity for a sequence of consecutive slots, as well as on how long this sequence slots last. Thus, by calculating the integral in the capacity-time plot between the curves of the real capacity and the allocated rate, when the rate is above, we can derive a worst case estimation for the size of the buffer.
- **Spread the rate over the 1 second slot if possible.** If the algorithm provides an average of 500 in a slot and then does the following allocation [500,500,500,500,200,0,0,0,0,0], it can be spread (with the same theoretical cost) to an allocation like the following [300,300,300,300,300,300,300,100,0,0]. That way, we avoid sending at full rate at any of the slots, thus probably avoiding that problem (unless the deviation from the average is really big, like when another user starts taking a big part of the available resources). On the other hand, this approach might be unreliable at the online case, since it relies a lot on far future capacities for which the uncertainty of their real value is higher (even though they are just some hundreds of ms more into the future), Also there is always the case where the algorithm has provided a solution which fully utilizes all of the slots in the first place.
- **Change the granularity of the average.** Instead of averaging in the order of 1 second, average over periods of smaller time duration. That way outliers will not be able to affect the average a lot and also the predicted 100 ms capacities will reflect better the actual conditions of the channel and the cell at this time and as a consequence be closer to the real values. This has the drawback that both greatly increases the computational cost of the optimization algorithm, (which is already high about a second for a single user offline case of a 10 seconds video) and also might not be that feasible to have such detailed data in the real online case. Essentially the 1 second prediction of capacity is an one second prediction of the following parameters: mobility (which is directly related to CQI/MCS), background traffic (other UEs connected to cell and the kind and amount of traffic they generate, as well as UEs that are about to appear in the user's cell or leave). Making accurate predictions in the order of hundreds of milliseconds for the above is really challenging.

[15]post by Nidhi at http://www.linkedin.com/groups/LTERLC-buffer-size-operators-discretion-1180727.S.158319679

c) The assumption that capacity is equal to goodput.: As discussed above, the algorithm does not take into account retransmissions or received IP packets that turn out to be corrupt. A viable solution is to apply more strict restrictions on the lowest amount of received data at a given time instance. By setting a higher target, it is possible to have safe minimum amount of data in the buffer, in case of retransmission delays. Of course, the most robust solution would be to always aim for a full buffer, but this would cancel the gains of the algorithm. A safe choice that seems to achieve a good balance is to set a minimum target of an empty buffer plus one group of frames (10 IP packets in our simulation scenarios). In most cases though, aiming for two or three IP packets in excess proves to be adequate.

2) Problems that arise when we measure capacity based on the number of transmitted TBs:

a) Segmentation of IP packets to TBs.: If an IP packet proves to be too big for to be carried in a single TB, then it is segmented and transmitted to the UE over multiple TBs. Upon receiving the TBs, the UE stores them at RLC layer, until all the TBs that carry data of a single IP packets are delivered. Upon delivery of the last TB, the IP packet is reconstructed and forwarded to the higher layers of the UE, where the video application receives it. This process can cause significant delay in the delivery of the IP packets. In order to illustrate this concept better, we can assume an example visualized in figure 3. If we suppose that the UE is in a location with unfavorable conditions, the MCS is by definition low, so the allocation algorithm will try to avoid sending any data, unless it is necessary. If the UE stays in that position for long enough time the buffer will become empty and the algorithm will allocate just enough data to ensure uninterrupted video playback, thus the target buffer state for the next time instances, would be an empty buffer. The IP packets that are sent though, are forced to be segmented because they are bigger than the TB size dictated by the current MCS. In a lot of occasions the amount of data received by the UE in a slot is equal to the amount data scheduled by the algorithm, but because of the segmentation, the last packet might not be received fully and thus not delivered to the application. So, the amount of data received in the application is not enough to decode a group of frames causing a rebuffering delay.

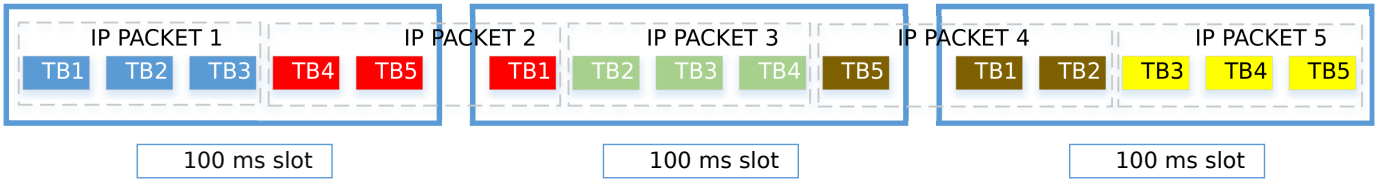


Fig. 3: Segmentation of IP packets to TBs. On some occasions, this can cause the late delivery of data to the buffer application, even though they are present on time at the UE.

This example might be easier to comprehend by using numeric approach. Suppose that in a 100ms time slot 5 TBs of size 500 bytes can be delivered. So, the capacity of this 100ms slot will be calculated as 2500 bytes per 100 ms. Given that capacity, the algorithm assumes that exactly that much data will be delivered to the application and might set a rate for that slot equal to 2500 bytes per 100ms, in order to reach a target for the time instance of the end of the slot equal to an empty buffer. If the IP packet that is transmitted is 1500 bytes, then it will have to be segmented and transmitted in 3 TBs of 500 bytes in order to be sent to the UE (for simplicity we assume that TBs do not have headers). As predicted by the algorithm, all 5 TBs will be delivered to the UE, but only the first 3 will be reconstructed to an IP packet that will be forwarded to the video application on time. The rest will remain to the lower layers of the UE, until the last TB arrives. As a consequence, only 1500 bytes will be delivered to the video application before the deadline.

The amount of delay between the delivery of TBs to a UE, varies depending on the competition in the eNodeB. So, the more congested a cell is, the more UEs will be scheduled before the target UE will be able to be scheduled again. Essentially, this problem becomes more intense in congested cells and it also usually appears alongside other problems discussed in this section, causing severe drawbacks. Some possible solutions are:

- **Increase the low constraint by one IP packet.** This will ensure that the video application is not affecting by any possible delays in the delivery of the last packet. As we mentioned though, this problem appears in locations with bad reception, so it acts on top of other problems. Thus, a more safe choice is to increase the low constraint by 3 or 4 packets.
- **Measure the capacity based on packets delivered to the video application.**
- **Make IP packets tiny, so that they can be transfered by one TB, regardless of the capacity of the TB.** This is the least viable solution of all. Even though, it would solve the problem, it would also greatly increase the overhead. Further, the TBs can get really small when the MCS has low values. The following table presents the size of TBs in bytes for the lowest values of MCS.

Value of MCS	Size of TB (bytes)
4	437
2	269
0 (out of range)	165

TABLE I

b) Transmission delays.: There is a non negligible amount of delay at various stages of the IP packet delivery process.

- After an IP packet is delivered to the UE (delivery of the last TB that carries a segment of an IP packet) there is a further 3ms delay before it is delivered to the application. The algorithm is oblivious of that delay.
- It takes about 1ms for packets to travel from the higher layer scheduler (video server in our simulations) to the eNodeB. We take into consideration this delay when transmitting data, from the video server and adjust the IP packet transmission times accordingly.

The effect of the above is that the last packet transmitted within a 100 ms slot, might arrive on time at the UE (according to the predictions of the algorithm), but not get delivered on time at the video application. Possible solutions to the above could be:

- **Transmit at a slightly higher rate at the beginning of a 100ms and at a slightly reduced rate at its end.** This change should be very small though because drastic changes could cause buffer overflow at the eNodeB. Also, another reason to avoid sending data towards the end of the slot is the segmentation of IP packets, discussed above.
- **Measure the capacity based on packets delivered to the video application.**
- **Increase the low constraint by 2-3 IP packets, depending on the competition on the cell.** As competition in a cell increases, the time between the arrival of two consecutive TBs that carry data of the same IP packet increases and as a consequence the overall delay increases. In order to ensure that the video application will always have enough data in the buffer regardless of the possible delays, we can set the buffer to have at least a couple of packets (or more in a congested cell) in excess, when the algorithm sets the target equal to the low constraint.

3) Problems that arise when we measure capacity based on the number of received IP packets by the application buffer:

a) The video server should take into account the variable delays in the delivery of packets.: Properly syncing the video server, in order to send packets based on when they are expected to be received by the video application proved to be rather tricky. The various transmission delays, as well as delays in the delivery of packets caused by the competition of other UEs in the cell, cause a variable amount of delay between the transmission of a packet from the server and its delivery to the Video application at the UE. The effect is magnified

in the cases where the IP packets have to be segmented. This delay though, should be taken into account, in order to transmit the IP packets by that much time earlier than their expected arrival. The only way to calculate the actual amount of delay for a specific UE in a specific scenario is by "trial and error". A rather positive fact is that that delay remains constant for as long as the competition in the cell remains constant. Thus, knowing the kind of competition in the cell (number of UEs that generate traffic and the traffic patterns they generate), might lead to an accurate prediction of the expected delay, for a given scheduler.

4) Additional considerations and remarks:

a) The algorithm sends more packets overall, compared to the benchmark application.: Thus, the total amount of data sent over the duration of the video is slightly bigger because of the extra headers. The benchmark sends a full packet every predefined period (10 ms in our case). The algorithm quite often sends packets that are not full (at the end of every slot in order to send exactly the amount of data that it has computed to send at the end of the slot, even in the case where it sends at full rate at consecutive slots, since the target rate for all the slots is almost never divided without a remainder by the packet size). Compared to the overall gains of applying the algorithm, this is negligible though. Finally, when the algorithm tries to maintain a state (sends at each slot data equal to the data the UE video application consumes), this effect does not surface, because the transmission rate is equal to the rate of the CBR.

b) High sensitivity to the mechanism of LTE that prevents out of order delivery of RLC PDUs.: This point discusses a special kind of retransmission that severely impacts the practical performance of the rate allocation algorithm. One of the lower layers of LTE, called RLC, is responsible for segmenting the packets that arrive from the upper layers of the stack to packets of its own layer called RLC PDUs. RLC PDUs have a unique sequence number that ensures in order delivery to the receiver. If the RLC layer of the receiver detects that a PDU is missing, it stops forwarding packets to the upper layers until it is received. While it waits for the missing PDU to arrive, the incoming PDUs that have a higher sequence number than the missing one are stored in a buffer. When the missing PDU arrives, it is forwarded to the upper layers along with the rest of the PDUs present in the buffer. LTE relies in a process called HARQ for the retransmission of the missing PDU, which even though has a priority over the normal transmissions at the scheduling process and usually delivers the missing PDU within 10 - 20ms, can not always guarantee a successful fast delivery. The reasons why this may happen are beyond the scope of this thesis. Upon detection of the missing PDU, a 100 ms timer starts. If the timer expires before the PDU is successfully retransmitted, then the missing PDU is transmitted as if it was new data and the PDUs present at the buffer of the UE are forwarded all together at the higher layers.

This process can potentially introduce more than 100 ms of delay in the delivery of packets (100ms because of the timer as well as the delay for forwarding the IP packets from the lower layer to the video application). Considering that in our scenarios we assume that the video application consumes data every 100ms, suffering such delays almost always causes a rebuffering pause, when the buffer is low. In our simulations, this problem usually appeared after a handover and when the UE was at locations with very bad CQI. When the UE is in situations like these, the algorithm aims to maintain an empty buffer, because the cost of delivering data is very high. Thus, the problem always occurs when the buffer of the video application is at its lowest levels, resulting in a rebuffering pause.

The only viable solution is to raise the constraint that is related to the minimum buffer level by at least the amount of data that a group of frames is worth. This method has proved to solve the problem, because in all time intervals, the data that will have to be consumed at the ending time instance are present in the buffer at least since the starting time instance. The cost of this solution could potential be very high though. It requires the UE to maintain a minimum amount of data in its buffer even in areas with unfavorable conditions, where the cost of transmission is very high.

c) The algorithm is oblivious of what proportion of the capacity can be used by the application.: The UE might request at any time instance data that are related to other applications. Thus, the capacity estimation is always more optimistic, causing the algorithm to allocate rates that can not be supported. The only solution to this problem would be to measure the capacity based on the number of IP packets over time, that are delivered to the application. Also, it is possible to predict to an extend the amount of data requested by other applications.

C. Proposed modifications to the algorithm and the LTE architecture

Before we discuss possible general solutions, we have to stress that the above problems have a tendency to appear at the same time. When the channel conditions are good and the the cell in not that congested, usually we do not face any these issues. When the channel conditions are bad and/or the cell is congested, we face combination of problems and their effects are additive. It is clear that the proposed solution should be robust enough to ensure proper delivery of IP packets in a significant number of scenarios regardless of the problem combinations that arise in them. One of the methods that we will use to increase the robustness is to apply more strict constraints (the area between the low and high limits in figure 2 (a) becomes tighter). This comes at the cost of reduced applicability of the rate allocation algorithm. Slightly stricter limits greatly reduce the number of feasible solutions and in many occasions might cause the algorithm to fail to provide a solution. Also, stricter limits imply lower flexibility in the possible allocations, so in general the provided solutions have a higher cost in both the total number of utilized RBGs and the value of $\sum_{i=0}^{99} \frac{r_i(t)}{c_i(t)}$. The final proposed solution should take into account this sensitive tradeoff between applicability/increased cost and robustness.

Obviously, the most robust possible solution is to aim for a full buffer at all the time instances. In the offline case that we are currently investigating, there is no need to resort to

such drastic measures, because despite the problems that arise we have a good estimation of the number of packets that can be delivered in the application in all time intervals. In the online case though, where the values of future capacities become more uncertain the more into the future they are, in a number of occasions the only safe choice is to apply greedy transmission, aiming for a full buffer. Also, because in the offline case we try to calculate allocations for the the whole time duration of the video, aiming for having always a full buffer will cause the algorithm to fail for the reasons discussed above. In the online case we apply the algorithm in much shorter time intervals, whose size is dependent on for how far into the future we have accurate predictions, thus the applicability of the algorithm is not affected at all. Further, even if the predicted capacities are not enough to reach a full buffer, in the online case, the algorithm does not fail, but uses the greedy approach to transmit as much as possible.

Our proposed modifications to the vanilla algorithm are the following:

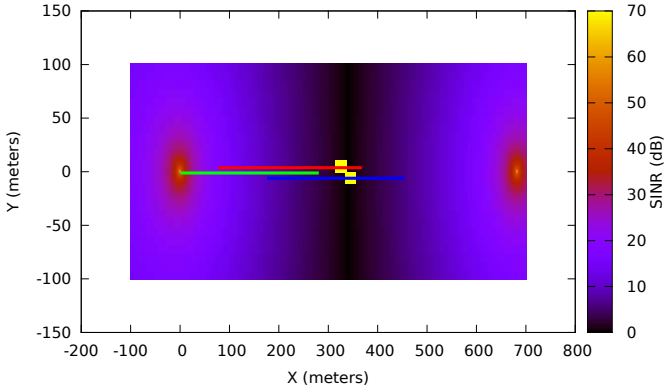
- **Calculation of the capacities at the application layer.** As we discussed before, calculating the capacity based on the bitrate of the mac layer can be misleading, because of the various delays and the side effects of mechanisms present at the layers below the application. On the other hand, assuming knowledge of data that the application actually receives proves to be more reliable, since the only delays that affect the "on time" packet delivery are caused by the background traffic.
- **Lower the constraint for the maximum amount of data that can be received at any time instance, by a group of frames (10 IP packets in our simulations).** This design choice removes potential buffer overflows between two consecutive stages that have as a target a full buffer.
- **Increase the constraint for the minimum amount of data that can be received at any time instance, by a group of frames (10 IP packets in our simulations).** This proves to make our solution more robust to delays, at the cost of utilizing more network resources. In contrast to the constraint modification discussed at the above point, this one is treated as a soft limit. If the algorithm is not able to provide a feasible solution for the current value of that constraint, we lower the constraint by one packet, until we reach the value it originally had before our modifications.
- **Calculation of the average capacity over smaller time intervals.** The default interval of 1 second proved to be inadequate in scenarios where background competitive traffic or CQI exhibits fast variations, like vehicular scenarios. Thus, calculation of the capacity over smaller time intervals provides better estimation of the real capacity at every time instance.
- **The distribution of the calculated capacities within the time interval over which we compute the average is not even.** According to the vanilla algorithm, all the slots that belong to a time interval over which the average capacity is calculated have a capacity equal to that average. Distributing the capacities in a manner that has the capacities of the first slots being higher than the capacities of the last slots proved to be more robust and also alleviate partially the drawbacks caused by the fact that the real capacity is always different than the computed average capacity. More specifically, if we average the capacities over:
  - **1 second.** The first slot is assigned 110% of the capacity of the computed average. The subsequent slots are assigned capacities that are gradually reduced by a factor of 2% of the average capacity. The last slot is assigned 90% of the capacity of the computed average.
  - **500 ms.** The first slot is assigned 108% of the capacity of the computed average. The subsequent slots are assigned capacities that are gradually reduced by a factor of 4% of the average capacity. The last slot is assigned 92% of the capacity of the computed average.
  - **100 ms.** In this case, there is no need to make any adjustment, since the duration of the time interval is equal to the period over which the buffer is reduced.

  In all the above cases, if we detect that a handover is about to take place within a time interval, we do not modify the allocated capacities. All the slots are assumed to have a capacity equal to the computed average.
- **Increase the buffers of the transmission queues of the eNodeBs.** This is a necessary design decision, because the miscalculation of the real capacities can cause packets losses because of buffer overflows at the eNodeBs. This effect is more acute in our solution, because of the artificially increased capacities of the first slots within a time interval.

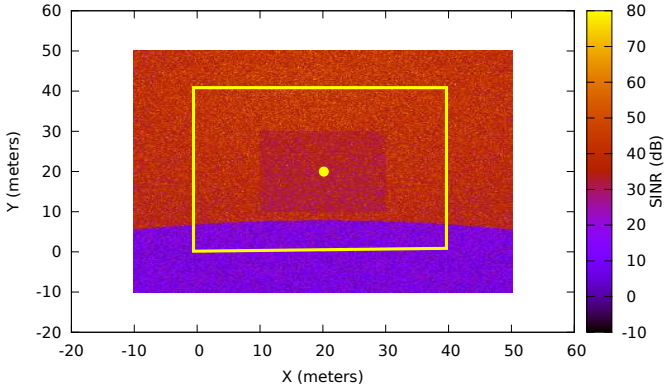
D. Evaluation of our proposed modifications

In this section we will evaluate the robustness and cost of our proposed solution, as well as explore the effects of varying the values of some of the parameters discussed in the above section.

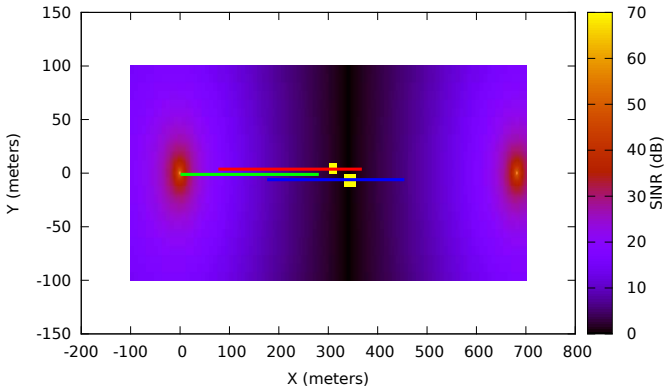
1) Simulation set up and scenarios: We investigated the performance of the algorithm in 3 characteristic LTE scenarios. In the vehicular scenario, depicted in figure 4 (a), three moving nodes with different starting positions move from the cell on the left of the figure to the cell on the right of the figure with a speed of 108 km/h (30m/s). This very fast movement allows for quick variations of the CQI and thus of the capacity. The yellow squares in the figure symbolize the locations of the handover events. The pedestrian scenario, depicted in figure 4 (c), has exactly the same topology and node trajectories as the vehicular scenario, but the nodes move with speed of 1m/s. Also the handover of the node with the blue trajectory takes place 1 second earlier. Finally, the building scenario, depicted in figure 4 (b), has 4 moving nodes wandering around a building with a speed of 30m/s and one stationary node inside the building, shown in the center of the figure. The base station in that scenario is located in the upper part of the topology (not visible in the figure), thus, when the UEs



(a) vehicular scenario

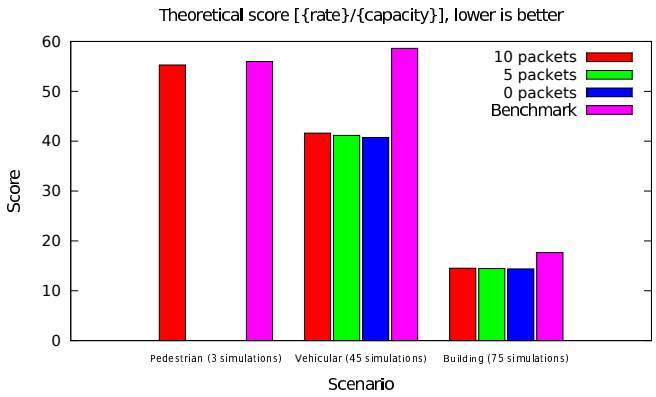


(b) building scenario

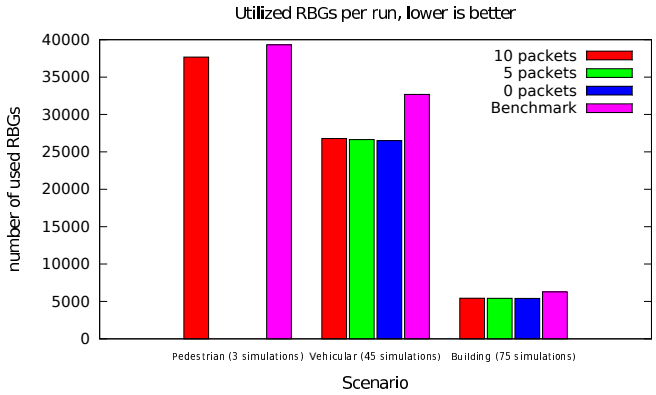


(c) pedestrian scenario

Fig. 4: Radio environment maps of the 3 simulation scenarios, where we tested our model. The yellow boxes represent the locations of the handovers and the colored lines the trajectories of the nodes.



(a) Theoretical score (average over 10 seconds)



(b) Number of used RBGs (average over 10 seconds)

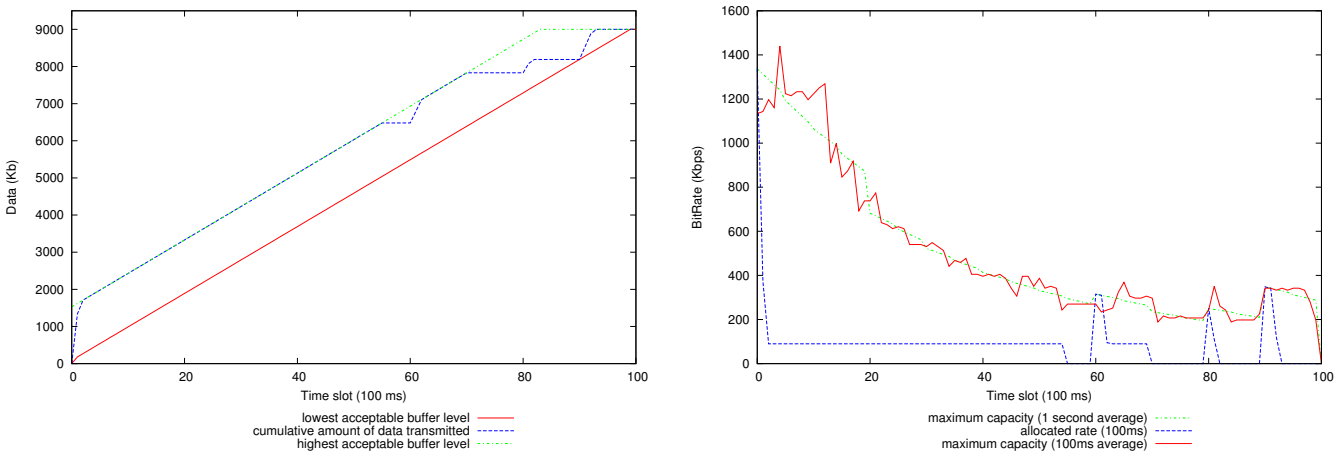
Fig. 5: Comparison between the cost of different solutions in the 3 tested scenarios.

move to the lower region of the topology they experience a significant drop in the SINR, because of the presence of the building. In all the scenarios one node is receiving video traffic with rates pre-calculated by the optimization algorithm while the rest of the nodes generate saturation traffic. In the vehicular and building scenario the video has 10 seconds duration and 900kbps bitrate and in the pedestrian scenario the video has 300 seconds duration and 900kbps bitrate. In every run of the simulation we vary the target UE, the granularity of the time interval over which the average is computed, the buffer size of the video application of the UE and the safety margin by which we raise the constraint for the minimum amount of data that can be received at any time instance. For benchmarking purposes, we also compare our solution to the constant bitrate scenario, where the video server transmits data with a constant rate of 900kbps.

It should be noted that in all the scenarios, we set the value of the RLC reordering timer to 1 ms, in order to exclude that factor from the simulations and we do not force retransmissions of lost or corrupt packets.

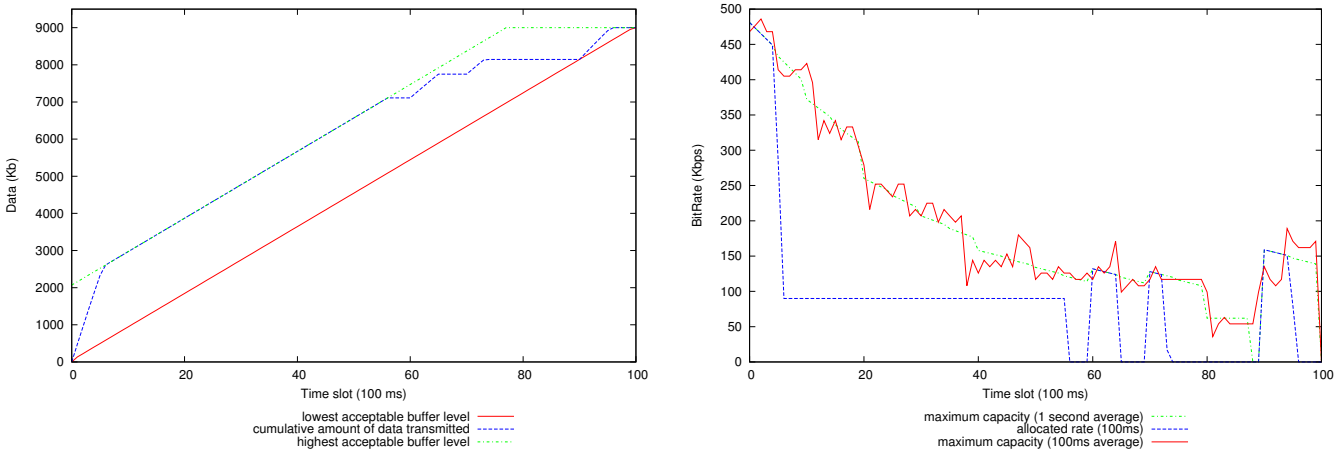
2) Rate usages and buffer states: Initially, figures 6, 7 and 8 show the variations of the buffer levels and rate allocations for the nodes of the vehicular scenario. These figures demonstrate how the rates are adapted to the channel conditions: the algorithm selects to use the full capacity when the conditions are good and relies on the UE buffer when capacity is low.

3) Cost reduction: The aggregated results of both the theoretical cost and the cost in terms of used RBGs, among all the simulation runs is depicted in figure 5. All versions of the algorithm greatly reduce both the theoretical cost and the number of RBGs needed to transmit the video data. As expected, the more robust a solution is, the more resources it requires. The difference for the various values of the safety margin is not very big though, especially in contrast to the cost of the benchmark application.



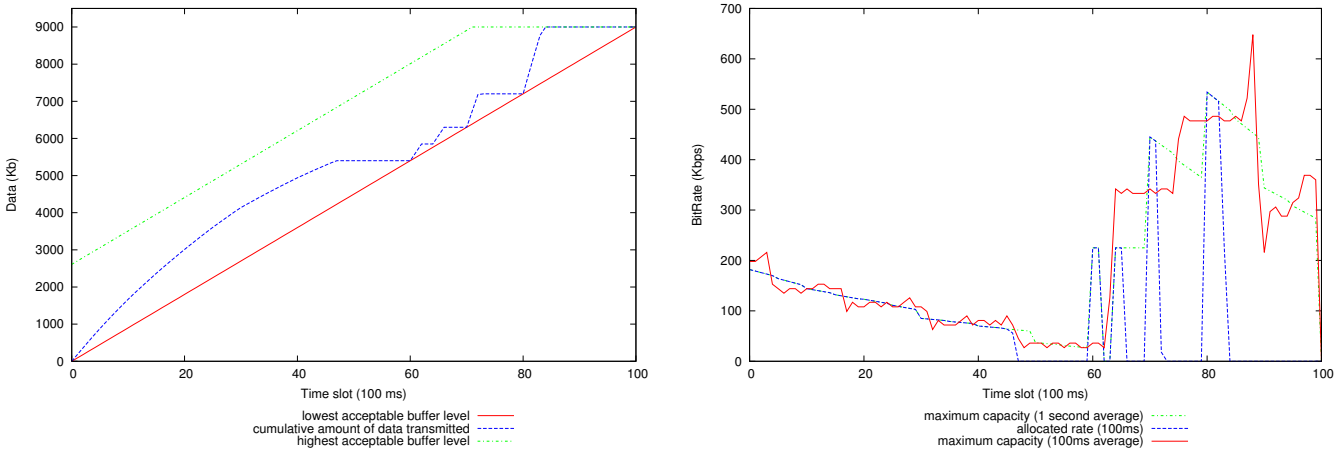
(a) Buffer (b) Bitrate

Fig. 6: Example distribution of rates and buffer occupancy for node 1 of the vehicular scenario



(a) Buffer (b) Bitrate

Fig. 7: Example distribution of rates and buffer occupancy for node 2 of the vehicular scenario



(a) Buffer (b) Bitrate

Fig. 8: Example distribution of rates and buffer occupancy for node 3 of the vehicular scenario

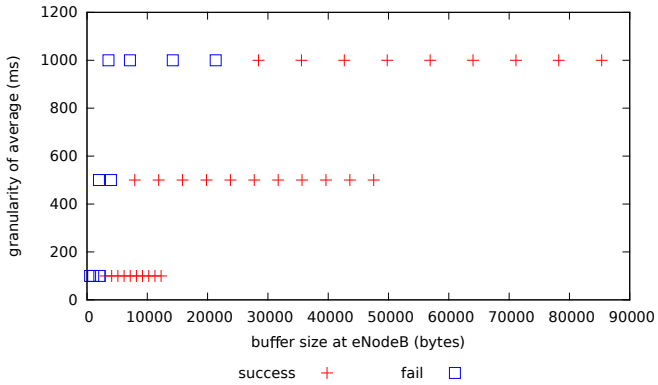
It should be noted that, in the pedestrian scenario, we only tested the approach that requires 10 packets as safety margin for the low constraint. In this scenario we observe minimum gains between our optimized approach and the benchmark. This is caused by the slow variation of the channel. For long periods of time (significantly longer than time for which the buffer can support uninterrupted playback, without receiving any new packets), the CQI that the UE encounters exhibits very small variations. The optimization algorithm reduces the cost by scheduling packets sooner than when the benchmark would, in time intervals where conditions are favorable. The constraints though limit the amount of time in advance each packet can be transmitted. Thus, the cost of transmitting packets in the pedestrian scenario is almost the same regardless of how sooner the optimization algorithm schedules them, because the slots used have almost the same capacity.

4) Buffer size at eNodeB: Next, we picture the relationship between the granularity of the time interval over which the average capacity is calculated and the buffer size at the eNodeB, in figure 9. We simulated the same scenario, gradually increasing the buffer size of the eNodeB. In all of our simulations, retransmissions are disabled, so if a packet is lost because of a buffer overflow at the eNodeB the total number of received packets is lower than the total number of transmitted packets and we suppose that we have a failure. As expected, higher granularity forces us to use significantly bigger buffers, in order to alleviate the effects of the wrong estimation of the real capacity and the drawbacks of our design decision to assume that the first slots of a time interval have higher capacity than calculated. The greater the error between the real capacity and the rate is, the greater the difference between the rate at which packets arrive at the eNodeB and depart from it, therefore requiring a bigger buffer in order to avoid overflows.

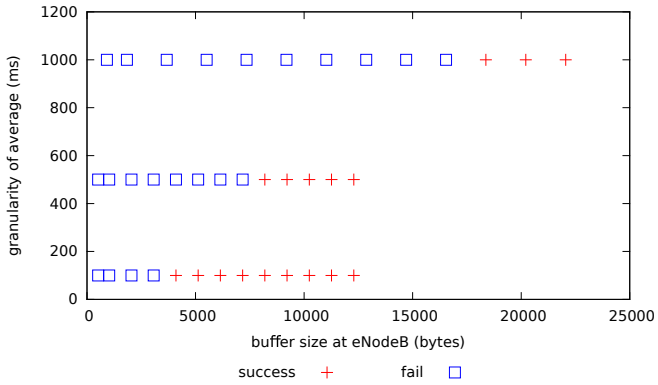
If the time interval is 100ms, the minimum required buffer size is smaller than the size of buffers that are currently present at the eNodeBs, thus in this case the modifications to the LTE architecture would be minimized.

5) Robustness of the different safety margins of the low constraint: In figure 10 and table II we present the details of a scenario where a node suffers from a loss of consecutive packets. This incident takes place right before the handover of the node with the red trajectory in the vehicular scenario. Before the handover, the node is at the edge of a congested cell, suffering from low CQI. After the handover the node arrives in an uncongested cell with similar CQI and as a result his capacity is significantly higher. Because the handover takes place in the middle of a time interval over which the average capacity is calculated, the last high capacity slots raise the average a lot more than the capacity of the first slots. Before the handover, packets are transmitted with the nominal capacity and as a consequence congregate on the buffer. Then, a seamless handover[16] takes place, resulting in loss of all the packets that were in the eNodeB buffer. Since, our scenarios do not support retransmissions, we are interested in detecting how much time is needed until the packet loss causes a buffer underflow. We consider that in case of retransmissions, the time difference between the loss of a packet and the buffer underflow event is equal to the amount of time that the UE would have to retransmit the packet. So, a bigger time interval is desirable.

[16]The packets of the first cell that are destined to the UE that did the handover are dropped instead of being forwarded to the new cell, as it is the case in the lossless handover.



(a) node 1



(b) node 2

Fig. 9: The granularity of the time interval over which the average capacity is calculated, determines the error between the allocated rate and the real capacity and therefore the value of buffer at the eNodeB. Two of the nodes of the vehicular scenario are pictured.

When the granularity of the average is 1 second the first packets are lost at time 6.1 seconds and in the 500 ms scenario the first packets are lost at time 6.2 seconds. In the 100 ms scenario, we do not experience any packet loss, because the rate differs very little from the real capacity. In table II we show when the buffer underflow events take place. It is clear that the 10 packets safety margin grants the system significantly more time to recover from the packet loss.

In figure 10, we depict the theoretical scores achieved in the same simulation runs. We observe that our solution has a slightly higher cost, despite being significantly more robust.

VI. Conclusion and Future Work

In this paper we identified and analyzed a lot of the challenges of applying a theoretical model of optimized stored video delivery over an LTE network. Based on that analysis, we have proposed possible modifications to the algorithm and

Granularity of average \ low constraint increase	0 packets	5 packets	10 packets
1 second	6.1531 Sec.	6.1531 Sec.	6.3531 Sec.
500 ms	6.3531 Sec.	6.4531 Sec.	9.9531 Sec.
100 ms	No loss	No loss	No loss

TABLE II: Time instance when the first buffer underflow occurs.

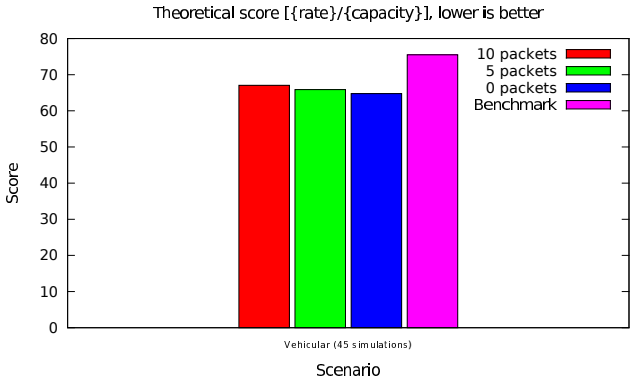


Fig. 10: Average of the theoretic scores achieved for different values of the safety limit of the UE buffer, over all the simulations where packet losses occurred.

to the LTE architecture that enable a trustworthy application of that model. The simulations have shown that our solution achieves a good balance between robustness and low channel utilization, while significantly reducing the cost compared to the benchmark application.

In subsequent works, we plan to explore the implications of retransmissions, rebuffering, as well as apply our solution to the online case, where knowledge of the future is both limited and inaccurate.

REFERENCES

[1] S. Traverso, K. Huguenin, I. Trestian, V. Erramilli, N. Laoutaris, and K. Papagiannaki, "Tailgate: handling long-tail content with a little help from friends," in *Proceedings of the 21st international conference on World Wide Web*, ser. WWW '12. New York, NY, USA: ACM, 2012, pp. 151–160. [Online]. Available: http://doi.acm.org/10.1145/2187836.2187858

[2] Z. Lu and G. de Veciana, "Optimizing stored video delivery for mobile networks: The value of knowing the future," in *Proceedings IEEE INFOCOM*, 2013, pp. 2706–2714.

[3] S. Sadr and S. Valentin, "Anticipatory buffer control and resource allocation for wireless video streaming," *CoRR*, vol. abs/1304.3056, 2013.

[4] S. Sanadhya, R. Sivakumar, K.-H. Kim, P. Congdon, S. Lakshmanan, and J. P. Singh, "Asymmetric caching: improved network deduplication for mobile devices," in *Proceedings of the 18th annual international conference on Mobile computing and networking*, ser. Mobicom '12. New York, NY, USA: ACM, 2012, pp. 161–172. [Online]. Available: http://doi.acm.org/10.1145/2348543.2348565

[5] F. Qian, Z. Wang, Y. Gao, J. Huang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "Periodic transfers in mobile applications: network-wide origin, impact, and optimization," in *Proceedings of the 21st international conference on World Wide Web*, ser. WWW '12. New York, NY, USA: ACM, 2012, pp. 51–60. [Online]. Available: http://doi.acm.org/10.1145/2187836.2187844

[6] J. Huang, F. Qian, Z. M. Mao, S. Sen, and O. Spatscheck, "Screen-off traffic characterization and optimization in 3g/4g networks," in *Proceedings of the 2012 ACM conference on Internet measurement conference*, ser. IMC '12. New York, NY, USA: ACM, 2012, pp. 357–364. [Online]. Available: http://doi.acm.org/10.1145/2398776.2398813

[7] 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

[8] B. Han, P. Hui, V. Kumar, M. Marathe, J. Shao, and A. Srinivasan, "Mobile data offloading through opportunistic communications and social participation," *IEEE Transactions on Mobile Computing*, vol. 11, no. 5, pp. 821–834, 2012.

[9] 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

[10] N. Ristanovic, J.-Y. Le Boudec, A. Chaintreau, and V. Erramilli, "Energy efficient offloading of 3g networks," in *Proceedings of the 2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems*, ser. MASS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 202–211. [Online]. Available: http://dx.doi.org/10.1109/MASS.2011.27

[11] V. A. Siris and D. Kalyvas, "Enhancing mobile data offloading with mobility prediction and prefetching," in *Proceedings of the seventh ACM international workshop on Mobility in the evolving internet architecture*, ser. MobiArch '12. New York, NY, USA: ACM, 2012, pp. 17–22. [Online]. Available: http://doi.acm.org/10.1145/2348676.2348682

[12] A. Y. Ding, P. Hui, M. Kojo, and S. Tarkoma, "Enabling energy-aware mobile data offloading for smartphones through vertical collaboration," in *Proceedings of the 2012 ACM conference on CoNEXT student workshop*, ser. CoNEXT Student '12. New York, NY, USA: ACM, 2012, pp. 27–28. [Online]. Available: http://doi.acm.org/10.1145/2413247.2413264