CACHING AND SCHEDULING MECHANISMS FOR H.264 VIDEO FLOW DELIVERY OVER DISCONTINUOUS COVERAGE WIRELESS NETWORKS

Azza Jedidi, Frédéric Weis
IRISA/INRIA, Rennes, France

Sylvaine Kerboeuf, Marie-Line Alberi Morel
Alcatel-Lucent Bell Labs, France

Keywords: Mobility, logical discontinuous coverage wireless network, caching, scheduling, scalability, start-up delay, resource management, layer encoded flow.

Abstract: The past few years have witnessed the deployment of a wide variety of multimedia applications over wireless networks. As a consequence, mobile users are demanding fast and efficient connectivity, especially concerning start-up delays and flow quality. Wireless networks aim at providing an everywhere coverage without prohibitive deployment costs. Thus, the coverage of each access point is extended with detriment to the mean throughput of the radio cell. The throughput of each zone decreases non-uniformely from the cell center to the cell edge. The coverage is continuous, but radio conditions met by users are not homogeneous. In fact, the data rate varies according to user mobility. In the context of streaming applications, intermittent high rate availability may lead to service disruption, especially when user number's increases. Thus, it is essential to design a network architecture able to efficiently deliver video flows to mobile users. In our work, a new equipment, called **network cache**, is introduced. The latter performs efficient flow caching and scheduling, in order to guarantee service continuity. The simulation shows that the proposed architecture gives satisfying service start-up delays and improves system scalability.

1 INTRODUCTION

Wireless [1] data transfers play an important role in current multimedia communications, especially thanks to the increase of available bandwidth and the growing needs for mobility. As a consequence, users are demanding fast and efficient ubiquitous connectivity. They require ¡¡any-time any-where¿¿ services, without delays or disruptions. Therefore, new network infrastructures had to be designed in order to guarantee ubiquitous coverage and to face wireless network constraints, in terms of bandwidth, error rates and other perturbations linked to environment heterogeneity.

In this context, the logical discontinuous coverage wireless network model has been proposed. This new model consists of a set of access points, *i.e.* antennas around which are defined the radio cells. Those antennas are discontinuously spread on the network area, thus providing a ¡¡many-time many-where¿¿ service. Actually, the idea of coverage discontinuity brings two major advantages. First, as it implies the use of a fewer number of access points, the architecture deployment will be cheaper. Second, radio cells disjunction hypothesis simplifies the radio frequency band management and avoids interference problems. The mechanisms defined in the framework of discontinuous radio coverage network model can be easily applied to the context of continuous radio coverage networks, where bandwidth is highly varying. High bandwidth areas are similar to coverage areas, and low bandwidth areas are similar to out of coverage areas. The main idea is to discriminate terminals by their radio conditions in order to select the data to be sent. In fact, terminals are supplied with data when they cross high bandwidth areas (*transfer areas*). And, low bandwidth areas are dedicated to detecting terminals presence (*presence areas*). Our research team has proved through simulations that avoiding data transfer in low bandwidth

[1] This paper presents results from a collaboration between INRIA ACES research team and the MAG project from Alcatel-Lucent Bell Labs

Jedidi A., Weis F., Kerboeuf S. and Alberi Morel M. (2008).
CACHING AND SCHEDULING MECHANISMS FOR H.264 VIDEO FLOW DELIVERY OVER DISCONTINUOUS COVERAGE WIRELESS NETWORKS.
In *Proceedings of the International Conference on Wireless Information Networks and Systems*, pages 66-73
DOI: 10.5220/0002022200660073

areas enhances the throughput use over the network and improves system scalability(Luu et al., 2007).
This new logically discontinuous vision of the network makes this model fit to 3rd generation networks, where coverage is continuous but with important bandwidth fluctuations, as well as 4th generation networks, where radio coverage could be physically discontinuous. Even if this model simplifies network deployment, the connectivity intermittence induces important challenges in order to avoid service disruption. Thus, terminals have to take advantage of the high bandwidth available when crossing a transfer area in order to store some part of the multimedia flow on their own caches, and to consume it when crossing presence areas.
Our works focus on logical discontinuous coverage wireless networks, and aim at providing a multitude of services. In this paper, our main goal consists in designing a scalable unicast streaming service, while reducing the start-up delays.
On the one hand, streaming servers are usually located in ¡¡classical¿¿ IP networks. Such networks are submitted to many constraints, like bandwidth variations. On the other hand, mobile terminals evolve in wireless logical discontinuous coverage areas, constituted of several access points. Each access point defines a high bandwidth area in each radio cell. The main idea we propose in the context of such networks, is to design an intermediate equipment that ¡¡catches¿¿ the video flows sent by the content server, and then efficiently distributes them to mobile terminals in the network. This equipment is called the **network cache** (see figure 1).
In a previous version of our work, the network cache

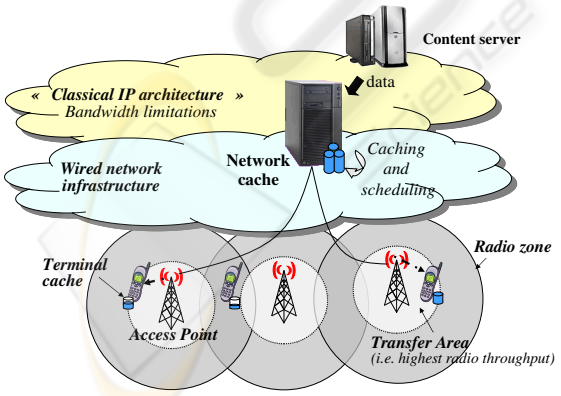

Figure 1: Logical discontinuous coverage wireless network.

acted as a temporary storage memory. In fact, it temporarily stored the part of the flow to be delivered to the mobile terminal, while the latter crossed a presence area. Then, when the terminal entered a transfer area, the network cache rapidely delivered the stored flow to him, taking advantage of the high bandwidth available. More precisely, the network cache was provided with a set of buffers. Each buffer was related to a single streaming request and contained the part of the flow to be sent to the corresponding mobile terminal(Luu et al., 2007). The use of buffers considerably enhanced the network scalability. In fact, it allowed users, crossing transfer areas, to take advantage of the high bandwidth available, in order to rapidly access the requested flow. Flow delivery was no longer subject to the bandwidth limitations at the content server level, as the flows were already stored in the network cache.
The terminal was also provided with buffers to store some parts of the flow in advance, when it crosses transfer area, taking benefit of the high bandwidth available. The terminal consumes this prefetched data while crossing out of coverage areas and, thus, avoids service disruption. The flow playback is smooth (low jitter), as the terminal cache prevented delay variation problems (see figure 2).
Moreover, the network cache played a scheduling

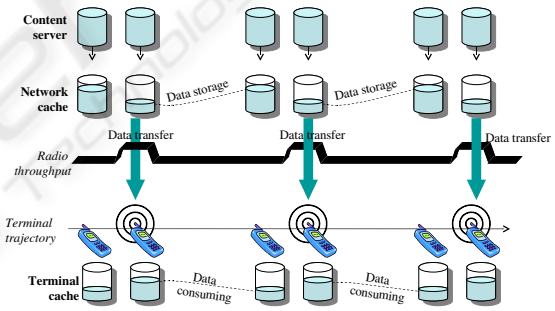

Figure 2: Caching mechanisms.

role. It classified flows in priority queues, depending (1) on terminal positions in the network and (2) on terminal buffers content. The scheduling policy favored terminals which were in transfer areas, as they will rapidly receive their flows thanks to the high bandwidth available. It also favored terminals whose buffer level was under a certain threshold, in order to avoid service disruption.
We proved through simulations that, thanks to the network cache buffers and the scheduling policy, the scalability of the network was considerably enhanced as the service disruption was avoided for a larger terminal density (Luu et al., 2007).
Nevertheless, buffers are temporary storage memories. So, the flow stored in a buffer is used by only one user, and could not benefit to users who ask for the same file later. In this paper, we propose to re-

place the buffers in the network cache with caching mechanisms. Actually, a copy of any requested video flow will be kept in the network cache, after the end of the streaming request. Later requests to the same video flow will be accessed faster as the flow is already stored in the cache network, nearer to the terminal than if it was in the content server. The delivery is no more subject to ¡¡classical¿¿ IP network bandwidth constraints (*i.e.* bandwidth constraints in the path between the content server and the network cache). Buffers, in the terminal, will also be replaced by a cache structure in order to store sufficient parts of the flow and to avoid service disruption.
Moreover, we propose to bring a more efficient scheduling mechanism in the network cache, taking into account some properties of the video flow in addition to terminal position and cache level. As H.264 flows are more and more used for nowadays mobile video applications, we chosed to consider H.264 video flows for our streaming service.
The remainder of this paper is organized as follows: first we make a brief study of the main video flow caching and scheduling mechanisms. Then, we present the mechanisms we chose in the context of our network. After that, the targeted architecture is detailed. Finally, the performances of our solution are evaluated.

2 VIDEO FLOW CACHING MECHANISMS

In the context of streaming applications, many caching algorithms could be adopted in order to design the network cache. For instance, prefix caching approach previledges caching the prefix of video flow in order to fasten the service start (Sen et al., 1999). Segment-based distance-sensitive caching approach groups the frames of the media object into variable-sized segments (the size of segment i is 2^{i-1}). Actually, segments size increases exponentially. As a consequence, the last segment length is half the cached portion length (Wu et al., 2001), which is especially interesting regarding the cache replacement policy. Indeed the latter will delete segments starting from the end of the cached flow. So, it will rapidly discard as big chunks as needed. A more efficient approach is the layered encoding and transmission method. Layer coders compress the flow in several layers. The base layer contains the data representing the most important elements of the flow. It represents the video flow in its minimal quality. Thus, it is the most significant layer. The other layers are hierarchically added to refine progressively the flow quality (Kangasharju et al., 2002). Depending on its capabilities, the terminal will subscribe to a set of cumulative layers to reconstruct the stream.

In our study, we consider H.264 SVC (scalable video coding) video flows. Those flows are layer encoded, which means that they are constituted of a base layer that represents the flow in a minimal quality, and some enhancement layers that could be hierarchically added to the base layer in order to enhance its quality. In our work, we use a model for the H.264 SVC video flows with a base layer and two enhancement layers, as shown in figure 3. A layered caching approach seems to be the most appropriate caching strategy in the context of such flow model.

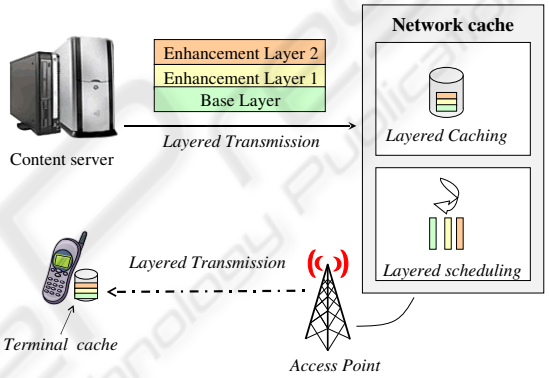

Figure 3: Layer encoded flow model.

3 OVERVIEW OF VIDEO FLOW SCHEDULING MECHANISMS

In this paper, our goal is to deliver H.264 video flow over a logical discontinuous coverage architecture. To aim that goal, we had to design a scheduling algorithm that fits our architecture needs and our video flow properties.
In the context of logical discontinuous coverage networks, video flow scheduling may become a very hard task as it is subject to a multitude of constraints, such as bandwidth limitation and asymmetry, important error rates and the high probability of service disruption due to client mobility.

Multirate Wireless Fair Queuing (MR-FQ) Algorithm. Consists in transmitting the video flow at different rates depending on the channel conditions(Wang et al.,). This algorithm takes into consideration both time and service fairness. This algorithm increases the overall system throughput and guarantees fairness and bounded delays.

Layered Quality Adaptation Algorithm. Consists in adding and dropping layers of a layer encoded video stream to perform smooth flow adaptation(REJAIE and REIBMAN, 2001). The main goal of the algorithm is to fit available bandwidth changes, without rapid and disturbing variations in quality. Actually, congestion control changes the transmission rate rapidly, over several round-trip time, whereas, hierarchical encoding permits smooth video quality adjustment, over long periods of time. The difference between both timescales is obtained thanks to data buffering at the receiver, which smoothes the rapid variations in available bandwidth and allows an almost constant number of layers to be played.

Buffer Sensitive Bandwidth Allocation Algorithm. Recommends that a client maintains sufficient video frames at the buffer in order to improve the system adaptability and minimize the impact of overloading(Yuen et al., 2002). The video playback starts only after reaching a determined buffer level. Based on the play rate of a video, the system calculates the buffer playback duration. The main idea of the method is to allocate the available bandwidth at a base station, which will serve the concurrent requests in the cell, based on their playback buffer durations.

Discussions. The layer encoded flow model that we adopted suggests the layered quality adaptation as the most appropriate scheduling approach. Moreover, this approach favors system scalability. Our scheduling algorithm, presented in the next section, is also inspired by the Buffer Sensitive Bandwidth Allocation strategy, as we took into account the terminal cache level, when the user enters in the radio cell.

4 H.264 SVC FLOW DELIVERY OVER A LOGICAL DISCONTINUOUS COVERAGE WIRELESS NETWORK

The key element for video flows delivery in the designed network is the network cache. This equipment is mainly made of two modules: the cache and the scheduler.
The cache is divided into three parts, each of them corresponds to the memory storage of one layer. Thus, for each flow, each layer is stored in the corresponding part of the cache. And, these parts are provided with a mapping table in order to match the memory cells with the corresponding flow identifiers. A similar cache structure is adopted for terminal caches.
The goals of the scheduler are (1) to guarantee the continuity of the streaming service and (2) to shorten the start-up delay. To aim that target, it has to efficiently distribute the video flows to the terminals.
In the previous version, the scheduler dynamically classified the video flows in priority queues, taking into account the terminal position (*i.e.* transfer area / presence area) and its buffer filling level. The streaming service was started when the buffer reached a ¡¡start-up buffer level¿¿ that corresponds to a playback duration of 30 seconds, which is a typical size of a terminal internal buffer for a streaming service.
Now, the scheduler classifies the flow layers separately. In fact, it considers the hierarchical order of layers, in addition to terminal position and cache level. As a consequence, the streaming service starting is only linked to base layer caching. Since the equivalent of 30 seconds of playback duration is cached in the terminal, the video playback starts. Moreover, as the scheduling privileges the base layer transmission, enhancement layers could be delayed or even omitted in order to guarantee service continuity. In fact, service continuity in such a flow model is equivalent to base layer streaming continuity. Actually, the level of terminal cache filling-up necessary to streaming start-up is defined by making a compromise between (1) a short start-up delay on the one hand, and (2) caching a part of the flow, sufficient to cross an out of coverage area without service disruption, on the other hand. Depending on the network conditions (i.e. available bandwidth, terminal density, etc.), enhancement layers could be progressively added or ommited, in order to ”smoothly” adapt the quality of the received flows.
Combining those different criteria, we conceived the algorithm detailed below. `BL` refers to the base layer. `EL1` represents the first enhancement layer and `EL2` the second one. `TA` means that the terminal is located in the Transfer Area. `CL` refers to the critical terminal cache filling-up level. The priority queues are numbered from one to six, classified with a decreasing order of priority. Let us consider a flow requested by a terminal. The three layers of that flow will have to be classified by the scheduler. Finally for each layer, the following algorithm will be iterated.

Algorithm 1 : Scheduling policy in the Network Cache.

```
if (Layer = BL) & (TerminalCacheLevel ≤ CL) then
    Priority Queue 1
else
    if (Layer = EL1) & (TerminalCacheLevel ≤CL) & TA then
        Priority Queue 2
    else
        if (Layer = EL2) & (TerminalCacheLevel ≤ CL) & TA then
            Priority Queue 3
        else
            if (Layer = BL) & (TerminalCacheLevel ≥ CL) & TA then
                Priority Queue 4
            else
                if TA then
                    Priority Queue 5
                else
                    Priority Queue 6
                end if
            end if
        end if
    end if
end if
```

This algorithm favors service continuity as it guarantees, with the highest priority, the delivery of the base layer to terminals whose cache is under the critical threshold. Moreover, it clearly appears that it favors flow delivery to terminals which are in transfer area. In fact, they take benefit of the high bandwidth available, wich allow them to store some extra part of the flow in their cache to avoid service disruption while crossing presence areas.

5 SIMULATION AND MAIN RESULTS

Our solution has been evaluated in the framework of a discrete event modeling based on a Java simulator, which uses DESMO-J (Discrete event simulation framework in Java) library(Des,). The simulated surface is 1 km^2. There are one network cache and six Access Points (AP). The server streaming rate is 1 Mb/s (256 kb/s for the base layer, 256 kb/s for the first enhancement layer and 512 kb/s for the second one). The simulated environment is dense urban environment like Manhattan. Each AP defines a radio cell, with a range of 50 m. A set of terminals move in a Manhattan topology, randomly crossing different radio cells and out of coverage areas, at a speed of 50 km per hour (vehicular model). Their cache content allows them staying out of coverage for a certain duration without service disruption(Luu et al., 2007). This duration is called the Time Out-of-cover (ToC). It corresponds to the longest time for which the service is satisfied, while the user is out of cover. So, this duration corresponds to the amount of data that the terminal must store in its cache to guarantee a continuous data delivery. As a consequence, if the user remains out of cover more than the ToC value, sevice disruptions may appear. In our previous studies, we evaluated the ToC value for our mobility model and access point topology to 240 seconds.

The terminal may ask for any video flow in the streaming server. If this flow is already stored in the network cache, its layers are scheduled and transmitted directly from the network cache to the terminal. Else, the three layers are sent from the server, scheduled by the network cache and transmitted to the terminal, while being simultaneously stored in the network cache. The terminal can start playing the video flow since the base layer reaches the start-up threshold, which is equivalent to a playback duration of 30 seconds.

In order to evaluate our system, it seems important to point out our main goals. In fact, we need to support an important number of terminals while guaranteeing short start up delays, and service continuity. To aim at that target, we adopted a layered flow model, and we took this model into account for the design of the caching and scheduling mechanisms. The main principle was to distribute the layers depending on the terminal position and cache level. We chose to guarantee service continuity, even if we had to temporarily degrade the flow quality, by omitting some enhancement layers.

In the next sections, the performances of the designed system are evaluated, considering the start-up delay (section 5.1) and the system behavior when the terminal density increases (section 5.2).

5.1 Start of the Streaming Application

Let us consider a terminal, alone in the network. First, the terminal requests a video flow which is not already cached in the network cache. The flow is sent to the terminal from the streaming server, and simultaneously stored in the network cache.
At start of our streaming application, the network cache is empty. Here, our main goals are (1) to provide as quickly as possible the amount of data required to start the streaming application (*i.e.* terminal cache level = 30 s * Service Data Rate) and (2) to min-

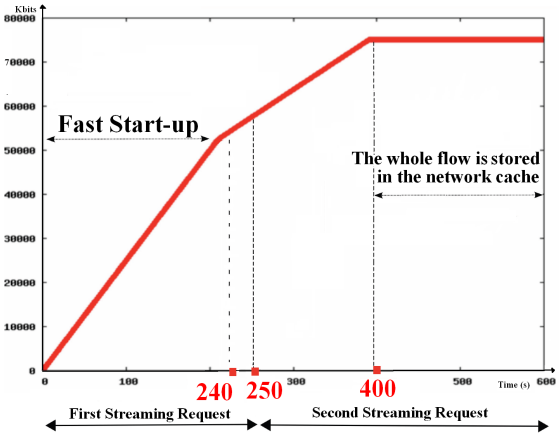

Figure 4: Network cache filling.

imize service disruptions as soon as the application has begun to consume data in the terminal (*i.e.* terminal cache level = ToC * Service data rate). Therefore, we introduced a fast start-up period during which the network cache is able to retrieve the beginning of the stream twice as fast as the normal service rate (Fast start-up rate = 2 Mb/s, normal service rate = 1 Mb/s). As the terminal is provided with a doubled data rate, the time necessary to start the streaming application decreases from 30 s to 15 s (Luu et al., 2007). This fast start-up phase continues after the service start. So the terminal continues storing data in its cache while consuminng. After this fast start up period, the flow starts to be delivered at the normal application data rate (1 Mb/s).

Figure 4 illustrates the network cache content during the experiment. In this figure, the whole flow (the total of the three layers) is represented. At **240** seconds, the figure illustrates that application data rate decreases after the fast start-up phase (the graph's slope decreases). After that (at **250** seconds), the terminal decides to stop this first streaming request. At that moment, the terminal did not end the whole flow streaming. We may notice that the flow is not totally stored in the network cache at that moment (see figure 4). The terminal cache empties at the end of the streaming request. Nevertheless, in the network cache, the part of the flow stored is kept there. Now (at **250 seconds**), the terminal asks for the same flow, a second time. The second streaming request starts. A part of the flow is already cached in the network cache as shown in figure 4. This figure shows that, during this second streaming request, the part of the flow, remaining at the content server level, is still delivered to the network cache. At **400** seconds, the whole flow is stored in the network cache.

Figure 5 represents the evolution of the terminal cache content. In this figure, the flow layers are repre-

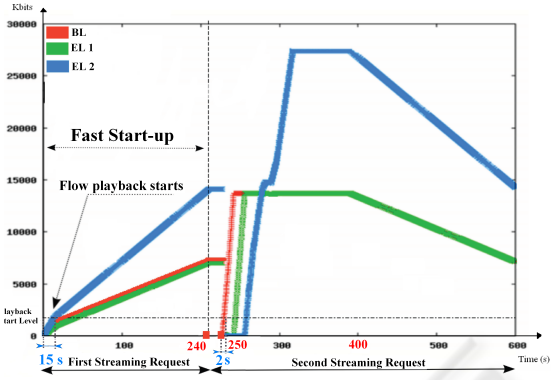

Figure 5: Terminal cache filling.

sented seperately. During the first streaming request, figure 5 shows that the terminal cache starts filling until its content reaches the playback start level (at **15** seconds). Then, the terminal starts playing the video flow at a 1 Mb/s rate, while it continues receiving the flow from the network cache at a 2 Mb/s rate. Since **240** seconds, the fast start-up period lasts and the application rate become equal to 1 Mb/s. The mobile terminal consumes its flow at the same rate it receives it, that is why we observe a plateau during this period. At **250** seconds, the terminal ends the first streaming request and empties its cache. At **250** seconds, the terminal asks for streaming the same video flow a second time. The second streaming request starts. Figure 5 shows that the base layer, and the first enhancement layer are sent at 256 kb/s, while enhancement layer 2 is transmitted at 512 kb/s. The second enhancement layer is thus received more rapidely than both the base layer and the first enhancement layer. The figure also shows that the layers are scheduled in the defined hierarchical order, which highlights the role of the scheduler.

Now, the terminal cache fills-up faster, as the flow is stored at the network cache level. As a consequence, the start-up threshold is reached more rapidly (The start-up threshold is the terminal cache level which corresponds to a playback duration of 30 seconds). Actually, the start-up threshold is reached in about 2 seconds in the second request while it is reached in about 15 seconds for the first one. This demonstrates the network cache impact during service starting.

We may notice that, at the beginning of the first streaming request, the network cache was empty, but when the second request starts, a part of the requested flow is already cached at the network cache. As a consequence, this part of the flow is very rapidely delivered to the terminal thanks to the high bandwidth available. Actually, as the flows are located in the network cache, they are not submitted to the bandwidth

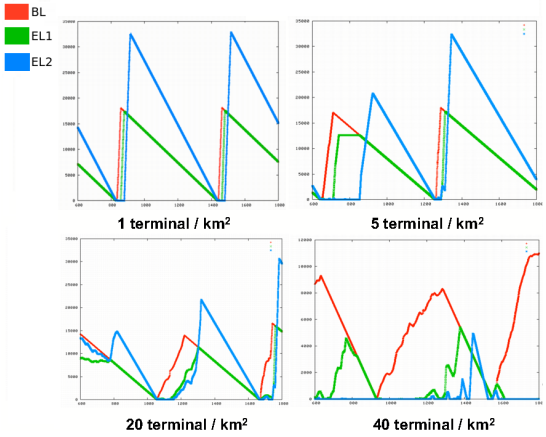

Figure 6: Terminal cache filling.

constraints at the video server level.

During the starting of a streaming request, some images might be played while only one or two layers are received. If only the base layer is received, the image is in base quality. If base layer and enhancement layer 1 are received, the quality is good. Finally, if all the layers are received, the quality is excellent. Table 1 details the percentage of images received, in each quality, by the considered terminal.

5.2 Scalability of the Solution

If a terminal, alone in the network, requests a video flow already stored in the network cache, the three layers are almost immediately received in the terminal cache. As terminal density increases, reception becomes slower, because the available bandwidth is shared between users. Moreover, the flow quality may need to be degraded (by omitting enhancement layers) in order to send base layers to all users without service disruption. Figure 6 represents the filling-up of the cache of one terminal, for different values of terminal densities. The figure shows that the enhancement layers reception is delayed as the number of terminals increases. The idea is that the base layer of the requested flows must be sent to all terminals in order to maintain service continuity. Enhancement layers could be delayed or even omitted in order to avoid service discontinuity. Moreover, acknowledgements sent by terminals allow the network cache to avoid sending enhancement layers of images already played at the client side with base quality. Table 2 illutrates how the system progressively degrades the received flows quality in order to maintain the streaming service continuity. Actually, the table details the evolution of the percentage of images received in base quality, in good quality and in excellent quality by one considered terminal, while terminal density increases.

Table 1: Received images quality.

Received images quality	
Base quality	1,6 %
Good quality	3,9 %
Excellent quality	94,4 %

Table 2: Received images quality.

Received images quality						
terminals density/Km²	10	15	20	25	30	40
Base quality	4,03 %	7,15 %	2,84 %	6,99 %	19,76 %	39,86 %
Good quality	3,15 %	4,26 %	6,54 %	15,04 %	27,09 %	42,68 %
Excellent quality	92,8 %	88,57 %	90,61 %	77,95 %	53,14 %	17,45 %

6 CONCLUSIONS

We work on extending logical discontinuous coverage networks to a multitude of services. This paper adresses the streaming service. We actually tried to deliver unicast video flows to a multitude of mobile terminals, in a logical discontinuous coverage network. The main goal was to guarantee a good quality of service. We especially focused on guaranteeing short start-up delays and system scalability. To achieve this goal, appropriate cache and scheduling modules have been designed. Those mechanisms took benefit from the video flow layered model.

The network cache plays a crucial role. As it is nearer to the terminals than the origin server, start-up delays are considerably reduced. Actually, flow delivery is no more submitted to bandwith limitations at content server side. Terminal caches allow flow storing in order to avoid service disruption while being in out of coverage areas, or low throughput areas. They also allow smooth video playback. The layered flow scheduling guarantees system scalability, thanks to the possibility of quality degradation, in order to maintain the service without any disruption.

Simulation showed that our goals are achieved : Terminals receive their flows with satisfying start-up delays, and service disruptions are avoided, thanks to the caching and sheduling approach adopted. The short periods of flow degradation allowed service continuity, while maintaining a good quality of experience (as the base layer flow contains the main elements of the stream).

Our futur work will concentrate on testing new mobility profiles, like the pedestrian model and on extending the evaluation of scalability performances, by simulating different video flows, available at different bitrates.

REFERENCES

A framework for discrete-event modelling and simulation. http://asi-www.informatik.uni-hamburg.de/desmoj.

Kangasharju, J., Hartanto, F., Reisslein, M., and Ross, K. W. (2002). Distributing layered encoded video through caches. IEEE Transactions on Computers, 51(6), pp. 622-636.).

Luu, A., Morel, M.-L. A., Kerboeuf, S., Ménard, R., Tlais, M., and Weis, F. (2007). Improving Wireless Network Capacity by Introducing Logical Discontinuous Coverage. Report).

REJAIE, R. and REIBMAN, A. (2001). Design issues for layered quality adaptive internet video playback. In *Workshop on Digital Communications (Taormina, Italy)*.

Sen, S., Rexford, J., and Towsley, D. (1999). Proxy prefix caching for multimedia streams. In *IEEE INFOCOM 99, New York, NY*.

Wang, Y.-C., Tseng, Y.-C., and Chen, W.-T. MR-FQ: A Fair Scheduling Algorithm for Wireless Networks with Variable Transmission Rates. Simulation: Transactions of The Society for Modeling and Simulation International, accepted, 2005. (SCI)).

Wu, K. L., Yu, P. S., , and Wolf, J. L. (2001). Segment-based proxy caching of multimedia streams. In *World Wide Web Conference(WWW10), Hong Kong*.

Yuen, J., Lam, K.-Y., and Chan, E. (2002). A fair and adaptive scheduling protocol for video stream transmission in mobile environment. In *IEEE International Conference on Multimedia and Expo, Lausanne, Switzerland*.