Enabling real-time TV services in CCN networks

Giuseppe Piro, Vincenzo Ciancaglini

Enabling real-time TV broadcast services over CCN networks

Giuseppe Piro[1] and Vincenzo Ciancaglini[2]
[1]Politecnico di Bari, Italy, g.piro@poliba.it
[2]INRIA Sophia Antipolis Méditerranée, France, vincenzo.ciancaglini@inria.fr

1. Introduction

Due to the growing importance that content sharing applications are experiencing in our everyday life [1], the emerging Content Centric Networking (CCN) paradigm [2] represents the most attractive solution for driving the current *host-based* Internet system towards a novel architecture focused around the *content-centric* concept. Basically, the communication in CCN requires the adoption of only two types of messages, namely *Interest* and *Data* [2]. A user may ask for a specific content by issuing an *Interest*, which is routed across the network towards nodes in posses of the required information, thus triggering them to reply with *Data* packets. While routing operations are performed by the strategy layer only for *Interest* packets, *Data* messages are sent back to requesting users just following the reverse path of the *Interest*, allowing every intermediate node to cache the forwarded content.

During past few years, CCN obtained a very warm attention in the scientific community. This is testified by the presence of several studies that have already investigated caching policies and data-transfer performances [3][4][5], congestion control issues [6], and routing strategies [7][8]. More recently, instead, thanks to the growing demand for TV services, some research activities are focusing the attention also to the design of sophisticated techniques enabling real-time TV broadcast services over CCN networks. Unlike Video-On-Demand, the real-time video distribution has to deal with a specific class of problems to ensure the timely delivery of an ordered stream of chunks. Moreover, video chunks have to be received within a given time interval (the *playout delay*), before being actually played. A chunk not delivered before such time deadline will result in degradation of the rendered video. Bearing in mind such issues, handling real-time streaming services in CCN networks represents a very difficult challenge. In this regard, only few preliminary works have been proposed in literature. The architecture presented in [9] has been designed for mapping HTTP-based streaming applications in a CCN. A novel cooperative caching strategy enabling time-shifted TV services has been discussed in [10]. A time-based Interest protocol is proposed in [11] in which a user sends a specific *Interest* message asking for a group of contents generated by the server during a specific time interval. Similarly to the previous paper, also in [12] is proposed a mechanism through which a user may request for multiple *Data* packets by issuing one *Interest* message. Unfortunately, all of these works do not evaluate the effectiveness of conceived ideas in complex and realistic environments and, more in general, they do not propose (and test) any sophisticated strategy that adapts real-time services in high loaded networks.

In our very recent contribution [13] we designed a novel architecture, namely CCN-TV, supporting real-time streaming services through the data-centric approach. In this paper, we extend our previous work by testing the CCN-TV system in a more complex network scenario. In addition, a depth analysis of the role that some of the main components of a CCN node, i.e., the cache and the Pending Interest Table (PIT) table, have in the presence of real-time services, as well as the comparison with respect to a *baseline* scenario where these features are not implemented, will be provided too.

2. The CCN-TV Architecture

In CCN-TV we consider a network of nodes requesting different real-time video streams, identified by a *channelID*, served by one or more broadcast servers. Unlike canonical UDP/TCP-based streaming, in CCN-TV each video is divided in consecutive chunks, identified by a progressive *chunk number*, that have to be requested individually, via a dedicated *Interest*. This fundamental aspect naturally supports the implementation of a flow control mechanism through which each user can explicitly request for new chunks just when the old ones have been received (or in the case they are not more useful because out of delay). In line with these premises, a *channel bootstrap phase*, a *flow control strategy*, and an efficient mechanism for retransmitting *Interest* packets have been designed within the CCN-TV architecture. For enabling these functionalities we need to extend the basic structure of the *Interest* packet by introducing an additional *Status* field marking if the *Interest* is related to the *channel bootstrap phase* or to a *retransmission*. In the case it is necessary to be conformed to classical CCN messages, this field can be easily replaced by an additional entry in the content name.

The channel bootstrap phase

Due to video codec requirements, a video stream can be visualized at the user side only once a specific I-Frame has been received. Therefore, to bootstrap a TV channel, a client has to find the closest server and gather from it the chunk (and the corresponding *chunkID*) of the last generated I-Frame. To this end, it sends an *Interest* packet for the URI: [domain]/[channelID], with the *Status* field set to *BOOTSTRAP* and a *Nonce* field containing a uniquely generated value. In this way, the message will travel unblocked until the first good stream repository, that will answer with a *Data* packet providing information about the first chunk of the last generated I-Frame. Once the user received this *Data* packet, it will request subsequent chunks, using a sliding window mechanism detailed in the following.

The flow control mechanism

A sliding window mechanism has been properly designed for enabling the user to request subsequent chunks of a video content. First, let us define *Pending Chunk* and *Pending Window* as the chunk whose *Interest* has been sent by the node and the window containing W different pending chunks not yet received, respectively. In details, together with the *chunkID*, we store in the *Pending Window* the timestamp of the first request and the timestamp of the last retransmission. Hence, whenever a new data message is received, or if the node does not receive any data for at least *windowTimeout* seconds, the following operations are performed: (1) purge the *Pending Window* from all the chunks who are expired, i.e., who have already been played; (2) retransmit all chunks that have not been received within the *windowTimeout*; (3) transmit, for each slot that got freed by the received or expired chunks, the *Interest* for a new one.

Interest routing

Normally, a CCN node does not propagate *Interest* packets related to contents already requested by other users in the past but not yet satisfied with corresponding *Data* packets [2]. The PIT table is used to keep track of *Interest* packets that have been forwarded upstream towards content sources, combining them with the respective arrival faces, thus allowing the properly delivery of backward Data packets sent in response to *Interests*. It is important to note that this mechanism prevents the propagation of retransmitted *Interest* packets, thus compromising the right behavior of CCN-TV. In order to force the propagation of retransmitted *Interests*, the *Status* field is set to *Retransmission*: this configuration would impose nodes along the routing path to propagate it versus the router that can satisfy this request (i.e., by skipping the usual CCN mechanism).

3. Performance evaluation of CCN-TV

We evaluated performances of CCN-TV architecture, through computer simulations carried out with *ccnSim*, an open source and scalable chunk level simulator of CCN, built on top of the Omnet++ framework [14].

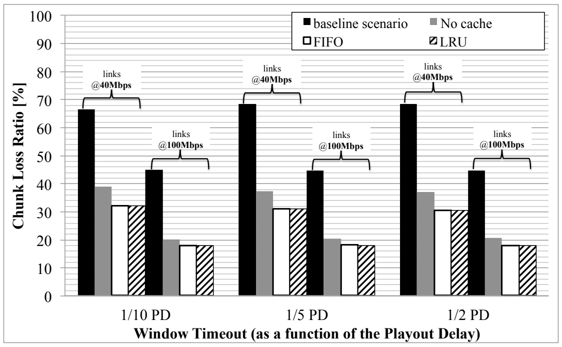
Network configuration and system parameters

Differently from [13], we considered a more complex network architecture composed by 68 routers connected among them according to the Deutsche Telekom topology [15]. A CCN node is directly installed to each router and no TCP or UDP encapsulation has been implemented. We assume the presence of only one small video-streaming provider that offers 5 parallel real-time transmissions to remote clients, each one connected to one router of the Deutsche Telekom network. In every simulation round, each video content is mapped to a video stream compressed using H.264 at an average coding rate randomly chosen in the range [250, 2000] kbps. On the other hand, every client chooses to watch one specific TV channel based on its popularity, which has been modeled through the Zipf distribution (in line with [13] we set α=1). In our tests, we adopted the optimal routing strategy based on the shortest path approach, already available within *ccnSim* [14]. Moreover, three caching strategies have been considered in our study: no-cache, LRU, and FIFO. When well-known LRU or FIFO policies are adopted, we set the size of the cache to 210 Mbits, i.e., a typical value for SRAM memories already available in the commerce [16]. The no-cache policy is intended to evaluate the performance of the CCN without using any caching mechanism. Furthermore, a *baseline* scenario, in which the no-cache policy is enabled and the PIT table is totally disabled (this means that each user establishes with the service provider a unicast communication and the server should generate a dedicated *Data* packet for each generated *Interest*), has been considered as reference configuration. Regarding the flow control mechanism, the window size W has been set to 10, ensuring that faces of the server are almost fully loaded in all considered scenarios. The transmission queue length associated to each face, Q, has been set in order to be larger than $Q = L_C \times \tau$, where L_C and τ represent the link capacity and the maximum propagation delay in the considered network topology, respectively.

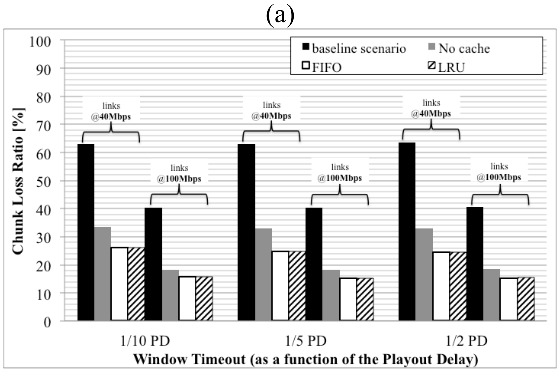
In order to evaluate performances of CCN-TV under various system configurations, we considered different settings of the bandwidth dedicated to real-time services (set in the range [40-100] Mbps), the *playout delay* (chosen in the range [10-20] s), and the *windowTimeout* (chosen in the range [1/10-1/2] of the *playout delay*). To conclude, each simulation lasts 300s and all results have been averaged over 15 simulations.

Simulation results

The chunk loss ratio, which represents the percentage of chunks that have not been received in time (i.e., before the expiration of the *playout delay*) by clients, is the first important parameter that we reported in Fig. 1 which describes how CCN-TV settings affect the quality of service offered to end users. We note that the amount of discarded chunks is very influenced by the *playout delay:* the highest *playout delay* allows the client to receive more *Data* packets before the expiration of the time deadline, thus reducing the amount of discarded chunks. In addition, the reduction of link capacities leads to a higher number of lost chunks, due to increased latencies induced by network congestion. By handling unicast communications, the *baseline* scenario generates the highest network congestion level, thus registering the worst performances. From this finding emerges the important role that both cache and PIT table have on network performances.



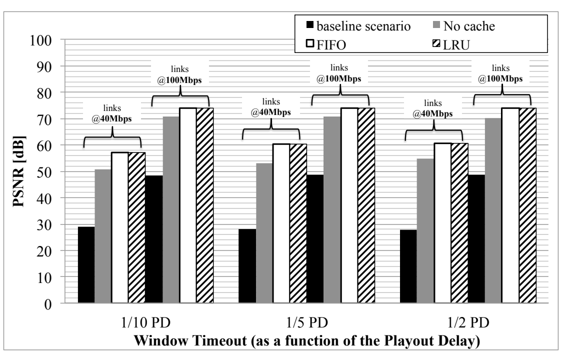
(a)



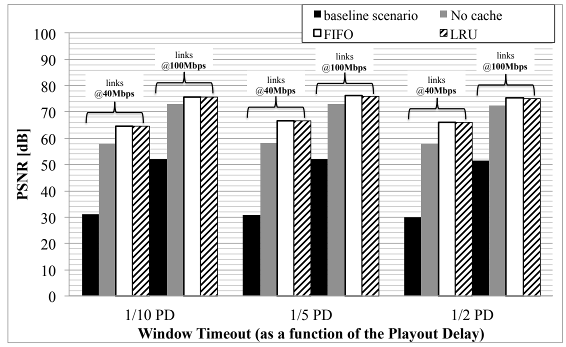
(b)

Figure 1. Chunk Loss Ratio when the *playout delay* has been set to (a) 10s and (b) 20s.

In order to estimate the Quality of Experience perceived by end users, we have also computed Peak Signal to Noise Ratio (PSNR) of received video flows (results are shown in Fig. 2). In line with previous results, the PNSR is higher in the same case in which the chunk loss ratio is lower. This means that quality of TV broadcast services ameliorates when we increase the *playout delay* and the link capacity. Also in this case, we remark that no-cache, LRU, and FIFO caching policies outperform always the *baseline* scenario.



(a)



(b)

Figure 2. PSNR of received video flows when the *playout delay* has been set to (a) 10s and (b) 20s.

To provide a further insight, we also reported in Fig. 3 the percentage of *Interest* packets sent by users and directly received by the service provider. In the *baseline* scenario, the total amount of generated *Interests* reach the remote server, thus excessively overloading its faces. By enabling the PIT table, even without implementing any caching mechanism, the system is able to halve the traffic load at the server side, thus improving significantly network performances. Finally, the traffic load handled by the server further reduces when a cache policy is activated. Anyway, it is evident that, in the presence of real-time flows, the cache does not represent an important CCN feature because it is not able to guarantee a notable improvement of system performances with respect to the case it is not used. On the other hand, we noticed that the PIT plays a more relevant role. In fact, in presence of live video streaming services, clients that are connected to a channel request same chunks simultaneously. In this case, a CCN router has to handle multiple Interest messages that, even though sent by different users, are related to the same content. According to the CCN paradigm, such a node will store all of these requests into the PIT, waiting for the corresponding *Data* packet. As soon as the packet is received, the router will forward it to all users that have requested the chunk in the past. According to these considerations, the use of the cache will not produce a relevant gain of network performances. Indeed, the PIT helps reducing the burden at the server side by avoiding that many *Interest* packets for the same chunk are routed to the server.

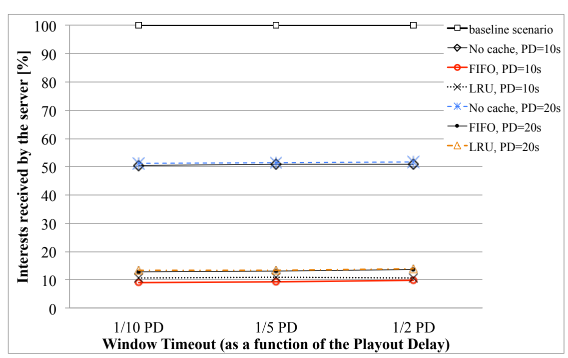


Figure 3. Percentage of *Interest* packets received by the remote server.

4. Conclusion

In this work, we investigate performances of the CCN-TV architecture, which has been properly designed for offering real-time TV broadcast services in CCN networks, under different system settings. Besides having demonstrated the effectiveness of the discussed architecture, presented results have highlighted that, differently from any caching policies, the PIT table has a fundamental role in reducing the burden at the server side in the presence of real-time streaming services.

References

[1] B. Ahlgren, P. A. Aranda, P. Chemouil, S. Oueslati, L. M. Correia, H. Karl, M. Sollner, and A. Welin, "Content, connectivity, and cloud: ingredients for the network of the future," *IEEE Commun. Mag.*, vol. 49, no. 7, Jul. 2011.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *ACM CoNEXT*, 2009.

[3] H G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, "Modeling data transfer in content-centric networking," in *Int. Teletraffic Congress, (ITC)*, 2011.

[4] M. Tortelli, I. Cianci, L. A. Grieco, G. Boggia, and P. Camarda, "A fairness analysis of content centric networks," in Proc. Of Int. Conf. on Network of the Future, (NOF) Paris, France, 2011.

[5] D. Rossi and G. Rossini, "On sizing CCN content stores by exploiting topological information," in *IEEE INFOCOM, NOMEN Worshop*, 2012.

[6] O L. A. Grieco, D. Saucez, and C. Barakat, "AIMD and CCN: past and novel acronyms working together in the Future Internet," in Proc. of *Capacity Sharing Workshop (CSWS) co-located with ACM SIGCOMM CoNEXT,* Dec. 2012.

[7] M. Tortelli, L. A. Grieco, and G. Boggia, "CCN forwarding engine based on bloom filters," in Proc. of ACM Int. Conf. on Future Internet Technologies (CFI), Seoul, Korea, Sep. 2012.

[8] W. You, B. Mathieu, P. Truong, J. Peltier, and G. Simon, "Dipit: A distributed bloom-filter based pit table for CCN nodes," in Proc. of Int. Conf. on Computer Communications and Networks (ICCCN), 2012.

[9] "Z. Li and G. Simon, "Time-shifted TV in content centric networks: the case for cooperative in-network caching," in Proc. of IEEE ICC, Jun. 2011.

[10] H. Xu, Z. Chen, R. Chen, and J. Cao, "Live streaming with content centric networking," in Proc. 3rd Int. Conf. on Networking and Distributed Computing, Hangzhou, China, 2012.

[11] J. Park, J. Kim, M. Jang, and B. Lee, "Time-based interest protocol for real-time content streaming in content-centric networking (CCN)", in Proc. of IEEE Int. Conf. on Consumer Electronics (ICCE), 2013.

[12] H. Li, Y. Li, T. Lin, Z. Zhao, H. Tang, and X. Zhang, "MERTS: A more efficient real-time traffic support scheme for Content Centric Networking", in Proc. of IEEE Int. Conf. on Computer Sciences and Convergence Information Technology (ICCIT), 2011.

[13] V. Ciancaglini, G. Piro, R. Loti, L. A. Grieco, and L. Liquori," CCN-TV: a data-centric approach to real-time video services", in Proc. of IEEE Int. Conf. on Advanced Information Networking and Applications (AINA), Barcelona, Spain, Mar., 2013.

[14] G. Rossini and D. Rossi, "Large scale simulation of CCN networks," in *In Algotel* 2012.

[15] http://www.telekom.com/ .

[16] D. Perino and M. Varvello. "A reality check for content centric networking", in Proc. of the ACM SIGCOMM workshop on Information-centric networking (ICN). New York, USA, 2011.

Giuseppe Piro is a postdoctoral researcher at Politecnico di Bari, Italy. His main research interests include quality of service in wireless networks, network simulation tools, 4G cellular systems, nanonetwork communications, and future Internet architectures. Piro has a PhD in electronics engineering from Politecnico di Bari. He founded the LTE-Sim project and is a developer of Network Simulator 3.

Vincenzo Ciancaglini is a PhD candidate at INRIA Sophia Antipolis Méditerranée, France. His research topics include the interoperability of content-cased systems, future internet architectures and media streaming systems. Ciancaglini has a MSc in Electrical Engineering, from Stockholm's Royal Institute of Technology, Sweden and a MSc in Telecommunications Engineering form Politecnico di Torino, Italy.