Providing crowd-sourced and real-time media services through a NDN-based platform

Giuseppe Piro, Vincenzo Ciancaglini, Riccardo Loti, Luigi Alfredo Grieco, Luigi Liquori

Providing crowd-sourced and real-time media services through a NDN-based platform

G. Piro[1], V. Ciancaglini[2], R. Loti[2], L.A. Grieco[1], and L. Liquori[2]

[1] DEI - Politecnico di Bari (Italy), {g.piro,a.grieco}@poliba.it
[2] INRIA - Sophia Antipolis (France), {vincenzo.ciancaglini, riccardo.loti, luigi.liquori}@inria.fr

Abstract. The diffusion of social networks and broadband technologies is letting emerge large online communities of people that stay always in touch with each other and exchange messages, thoughts, photos, videos, files, and any other type of contents. At the same time, due to the introduction of *crowd-sourcing strategies*, according to which services and contents can be obtained by soliciting contributions from a group of users, the amount of data generated and exchanged within a social community may experience a radical increment never seen before. In this context, it becomes essential to guarantee resource scalability and load balancing to support real time media delivery. To this end, the present book chapter aims at investigating the design of a network architecture, based on the emerging Named Data Networking (NDN) paradigm, providing crowd-sourced real-time media contents. Such an architecture is composed by four different entities: a very large group of heterogeneous devices that produce media contents to be shared, an equally large group of users interested in them, a distributed Event Management System that creates events and handles the social community, and a NDN communication infrastructure able to efficiently manage users requests and distribute multimedia contents. To demonstrate the effectiveness of the proposed approach, we have evaluate its performance through a simulation campaign using real-world topologies.

1 Introduction

Thanks to online social networks (like Facebook, Google+, LinkedIn, MySpace, and Twitter), which are experiencing an explosive growth since past few years, people can always stay in touch with each other and exchange messages, thoughts, photos, videos, files, and any other type of contents. This growth is sustained by the widespread adoption of new generation devices (such as notebooks, smart phones, and tablets) [1], together with emerging broadband wired and wireless technologies and, without any doubts it will become more and more evident in upcoming years. As a final result, people continuously generate and request a massive amount of data, provided by other users worldwide.

Horizons of social networks can be further extended, thus enhancing users interaction and data discovery, by introducing *crowd-sourcing approaches*, which refer to the practice of obtaining services and contents by soliciting contributions from a group of people that, in the most of cases, form an online community [2].

When used together, online social networks, mobile devices, and crowd-sourcing platforms have all the potentials to create connected communities scattered all over the world, thus paving the way to several novel applications. Their joint exploitation, for example, can be used to capture media contents, i.e., audio and video, from a very large number of users during an event (think for example to a football match, a concert, a public event, a dangerous situation, and so on) and delivering them in real-time to any other user in the world. This way, anyone can see (and listen) what others see (and listen) with their eyes (and ears) in a specific place of the world, while being physically far. To the best of our knowledge, network architectures enabling such kind of *social video real-time services* have not yet fully standardized. Hence, novel ideas and technologies have to be promoted by the research community in order to make this a very attractive application achievable as soon as possible.

As an answer to such issues, novel network architectures are being proposed, falling in the Future Internet umbrella [3]. They are mainly grounded on the Information-Centric Network (ICN) approach, which provide a way to drive the current *host-centric* Internet paradigm towards a novel *content-centric* behavior [3].

The ICN approach is currently investigated and developed in several projects, such as Data-Oriented Network Architecture (DONA), Publish Subscribe Internet Technology (PURSUIT), Scalable and Adaptive Internet Solutions (SAIL), CONVERGENCE, Named Data Networking (NDN), COntent Mediator architecture for content-aware nETworks (COMET), and MobilityFirst [5]. Despite some distinctive differences (e.g., content naming scheme, security-related aspects, routing strategies, cache management), they share a common receiver-driven data exchange model based on content names [6] [3].

In the authors' opinion, the ICN represents a very powerful technological basis to distribute *crowd-sourced* contents. Unfortunately, despite the number of works available in literature that investigate and propose innovative techniques to deliver contents in ICN, solutions properly designed for the considered scenario have not been proposed yet.

To bridge this gap, we conceived herein a network platform based on the NDN rationale, which is able to efficiently discover and distribute *crowd-sourced* multimedia contents within a distributed and on-line social network.

The devised platform is composed by (i) a community of users that, being into the same place to take part to an event record and broadcast it from their multiple points of view, (ii) a number of users interested to see what the aforementioned social community is capturing, (iii) a distributed *Event Management System*, which creates events and handles the social community, and (iv) a NDN communication infrastructure able to efficiently manage users requests and distribute multimedia contents. Moreover, it addresses four different tasks: *event announcement*, *event discovering*, *media discovering*, and *media delivering*. In addition, to offer an optimal management of multiple and heterogeneous events, we have also designed a hierarchical *name-space* and an efficient scheme, based

on the *sliding-window* technique, to download real-time media contents through NDN primitives.

We demonstrated the effectiveness of the our proposal with computer simulations conducted through the *ccnSim* simulator [7]. In particular, focusing the attention on a complex network architecture, which is composed by 68 routers connected among them according to the Deutsche Telekom topology, we evaluated the performance of multimedia services by varying the number of users that participate at a given event and generate different media contents, the amount network bandwidth dedicated to this service, and other parameters characterizing the *sliding-window* process.

The rest of the chapter is organized as in the following: a background on social networks, video distribution in Peer-to-Peer (P2P) systems, *crowd-sourcing* paradigm, as well as on the emerging NDN network architecture, will be provided in Sec. 2. An accurate study of the state of the art related to the management of streaming services in social, *crowd-sourced* and NDN-based platforms will be presented in Sec. 3, whereas the analysis of issues and challenges arising from the considered scenario will be discussed in Sec. 4. The conceived NDN-based solution will be deeply described in Sec. 5. Moreover, a performance evaluation of the proposed approach will be proposed in Sec. 6. Finally, Sec. 7 will draw the conclusions.

2 Preliminary concepts, background, and definitions

2.1 Online Social Networks

The past ten years have witnessed the arise of Online Social Networks as the predominant form of communication over the Internet. Online Social Networks are a software platform, in the form of a mobile application or an internet website, that allows for the building of social relations mongst people, and the exchange of information between them. In this sense they are different from Online Communities, although allowing sometimes for a similar form of communication, by being centered around the individual rather that a community.

Users of an online social network usually maintains a list of first-degree contacts (i.e. friends and families, or direct collegues), with whom to perform direct interactions, and a way of posting contents with different degrees of visibility (private content, visible to neighbors up to a certain degree in the social graph or public). The type and kind of content being published heavily depends form the scope of the Social Network, depending on which we can have:

- General purpose networks, such as MySpace [8] (at least in its first iteration) or Facebook [9];
- Geographically centered social networks, such as Sina Weibo [10] for the Chinese market, Orkut [11] and Hi5 [12] in South America, NK [13] in Poland, VKontakte [14] in Russia.
- Micro-blogging platforms, supporting only content in the form of text posts of limited length, such as Twitter [15], Thumblr [16], Identi.ca [17]

- Media-driven online social networks, i.e. networks dedicated to sharing a particular kind media only. this is the case of Vine [18] or Flickr [19];
- Interest-driven Online Social Networks, where the social graph between a user and its contacts is created according to specific criteria (common profession, similar interests). This is the case of LinkedIn [20]

Every Online Social Network is, up to a certain degree, a driving media for the so-called *crowdsourced content*, this means that they are platform where the users are creators and fruitors of content at the same time. In this sense one can expect a size and variety of produced content much higher than a normal website and, from a technical point of view, a bandwidth much less asymmetrical than canonical service, due to the need to support a consistent data upload together with the download.

2.2 Collaboration in real-time video distribution: P2P-TV

In a gossip-based peer-to-peer (P2P) system, all nodes (peers) interested in the same content build up a high-level overlay network [21]. Each peer establishes links with a limited set of peers called neighbors and exchanges pieces of data with them, receiving content from multiple sources, while serving multiple neighbors. Every peer offers its own upload bandwidth for content distribution, thus eliminating the need for high-capacity servers. This advantage has been fruitfully exploited in the past to design powerful file transfer applications [22]. Recently, the attention of the scientific community and industry has turned towards P2P-TV, due to the prevalence of this field in every day life, as well as in the market [23].
In P2P-TV systems, a multimedia data source generates a series of *chunks*, with each one of them containing a part of the audio/video bitstream, and makes them available to all peers connected to the overlay [24]. The main difference with respect to file sharing applications is related to the strict delay requirements of P2P-TV applications, which impose a deadline on each chunk at generation time: when a chunk is received after its *playout delay* (the deadline) has elapsed, it is considered lost [25]. Increasing the playout delay allows for more and more chunks to be received within this deadline: however, this advantage comes at the expense of the Quality of Experience (QoE) of the users, which is very sensitive to the timeliness of TV services [26]. Other performance limitations of classic P2P-TV are related to the so called bandwidth and content bottlenecks [27]. It is, in fact, possible that a chunk whose deadline is going to expire cannot be downloaded by a given node because none of its neighbors have that chunk (content bottleneck) or the upload bandwidth of the neighborhood is insufficient (bandwidth bottleneck). In both of these cases the chunk in question would be lost, and, consequently, the QoE would be impaired. Increasing the degree of cooperation of the overlay network by allowing each peer to establish direct links with many other peers is not a viable solution because of the overhead required to handle the high number of connections in each neighborhood.

2.3 The *crowd-sourcing* paradigm

The crowdsourcing concept itself takes form from a Wired article [28] in which the growing number of initiatives of offering services, from photos to division of labour, is distributed among the vast user pool of dedicated sites or communities, paralleling the jobs outsourcing of a business to an outsourcing to a vast crowd, a portmanteau created in that article that well identified the idea.
Vital in the crowdsourcing is the dense interconnection network between users, permitting new contributors into the service from a wide variety of expertise and capabilities. As such the pairing with social networks is self-emerging and obvious, as even the less structured, in terms of formal user connections and way of communicating, like Twitter for example, offers a deep, vast and variegated network of connections [29] and in many cases coordinating over a social network effectively permitted quicker and large scale operations, like disaster first assistance [30], validating the use those resources as a distribution hub for large scale jobs.
Crowdsourcing, due to the ability of easily complete hundreds of tiny per se jobs, usually difficult for total automation for a machine, is widely used and many dedicated platforms, like for example Amazon's Mechanical Turk (http://www.mturk.com) which aims to raise the workers quality by incentivating them with a small payment for each job completed. The tasks well suited for such approach ranges from language translation [31] and subtitle creation for movies or TV shows [32], to labeling by adding categories and metadata tags, videos [33], multimedia streams [34], images or paper document scans [35].
Still there are, regardless of the specific application, common problems, such as the quality of the work being directly proportional to the expertise of the workers, which are usually not professionals, and the usability of the software tools offered [36], the large amount of essentially spam results, polluting an otherwise higher level dataset or the quantity of workers not being sufficient to label the majority of data.
The approaches in dealing with low data quality are multiple, from offering better tools and referring to more selected worker force [36], to integrating missing data with known ones if the field is formalized enough [37] and comparing and integrating algorithmically [38] [39] or manually [31] [40] differing results on the same data.
Also exists similar approach using a probabilistical approach in detecting the unwanted negative "contributions" referred to as spam [41], this too contributes in raising the overall efficiency of the data elaboration.
And finally we can see adaptation of automatic cataloging to use previous human provided comparisons [42] and guided automatic frame extraction from videos [43] to generate the missing work needed if the community behind the crowdsourcing effort is not big enough or the data set to be analyzed too big.

2.4 Background on NDN

The ICN architecture conceived within the NDN project [44], known with the name Content-Centric Networking (CCN), is based on a "data-centric" ap-

proach: all contents are identified by a unique name, allowing users to retrieve information without having any awareness about the physical location of servers (e.g., IP address) [45, 46].

Each content is uniquely identified by the *Content Name*. In particular, NDN exploits a hierarchical naming scheme where names are composed by multiple human-readable components (e.g., strings), optionally encrypted, arranged in a hierarchy. This means that a *name tree* is introduced to identify all contents available into the network.

In NDN, the communication is driven by the *consumer* of data (i.e., the user that generates the request) and only two kinds of messages are exchanged: *Interest* and *Data*. The structure of both *Interest* and *Data* packets has been reported in Tabs. 1 and 2, respectively[3].

Table 1. Main fields of the *Interest* packet

Field	Description
Content Name	Specifies the requested item. It is formed by several components that identify a sub-tree in the name space.
Min Suffix Components	Enables the access to a specific collection of elements according to the prefix stored into the *Content Name* field.
Max Suffix Components	Enables the access to a specific collection of elements according to the prefix stored into the *Content Name* field.
Publisher Public Key Digest	Imposes that only a specific user can answer to the considered *Interest*.
Exclude	Defines a set of components forming the content name that should not appear in the response to the *Interest*.
Child Selector	In the presence of multiple answers, it expresses a preference for which of these should be returned.
Answer Origin Kind	Several bits that alter the usual response to *Interest* (i.e., answer can be "stale" or answer can be generated).
Scope	Limits where the *Interest* may be propagated.
Interest Life time	It indicates, approximatively, the time interval after which the *Interest* will be considered deprecated.
Nonce	A randomly-genenerated byte string used for detecting duplicates.

An user may ask for a content by issuing an *Interest*, which is routed towards the nodes in posses of the required information (e.g., the permanent repository, namely *publisher*, or any other node that contains a valid copy in its cache) [6], thus triggering them to reply with *Data* packets. Routing operations are performed by the strategy layer only for *Interest* packets. Whereas, *Data* messages,

[3] It has been taken from the official software implementation of the NDN communication protocol suite, i.e., CCNx, available at `www.ccnx.org`.

Table 2. Main fields of the *Data* packet

Field	Description
Content Name	Identifies the requested item.
Signature	Guarantees the publisher authentication.
Publisher Public Key Digest	Identifies the users that generated data.
Time Stamp	Expresses the generation time of the content.
Type	Explicits what the *Data* packet contains (i.e., data, encrypted data, public key, link, and NACK with no content).
Freshness Seconds	Defines the life time of the carried payload. It is used to schedule caching operations (eventually prohibited).
Final Block ID	Indicates the identifier of the final block in a sequence of fragments.
Key Locator	Specifies where to find the key to verify this content.
Content	It is the content with an arbitrary length.

just follow the reverse path towards requesting user, allowing every intermediate node to cache the forwarded content. We note that such a *routed-by-names* strategy enables in-network processing of data and requests and allows, at the same time, the optimization of service provisioning and the native support of one-to-many services.

All of these activities can be executed thanks to the exploitation of the following structures:

- the Content Store (CS): a cache memory that can implement different replacement policies as Least Recently Used (LRU), Least Frequently Used (LFU), and First Input First Output (FIFO);
- the Forwarding Information Base (FIB): is similar to an IP FIB except for the possibility to have a list of *faces* for each *Content Name* entry, thus allowing *Interest* packets to be forwarded towards many potential sources of the required *Data*;
- the Pending Interest Table (PIT): is a table used to keep track of the *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*.

When an *Interest* packet arrives to a NDN node, the CS is in charge of discovering whether a data item is already available or not. If so, the node may generate an answer (i.e., a *Data* packet) and send it immediately back to the requesting user. Otherwise, the PIT is consulted to retrieve if others *Interest* packets, requiring the same content, have been already forwarded towards potential sources of the required data. In this case, the *Interest*'s arrival face is added to the PIT entry. Else, the FIB is examined to search a matching entry, indicating the list of faces which the *Interest* has to be forwarded through. At the end, if there is not any FIB entry, the *Interest* is discarded. On the other hand,

when a *Data* packet is received, the PIT table comes into play. It keeps track of all previously forwarded Interest packets and allows the establishment of a backward path to the node that requested the data. To complete the description above, main functionalities of NDN nodes have been summarized in Fig. 1.

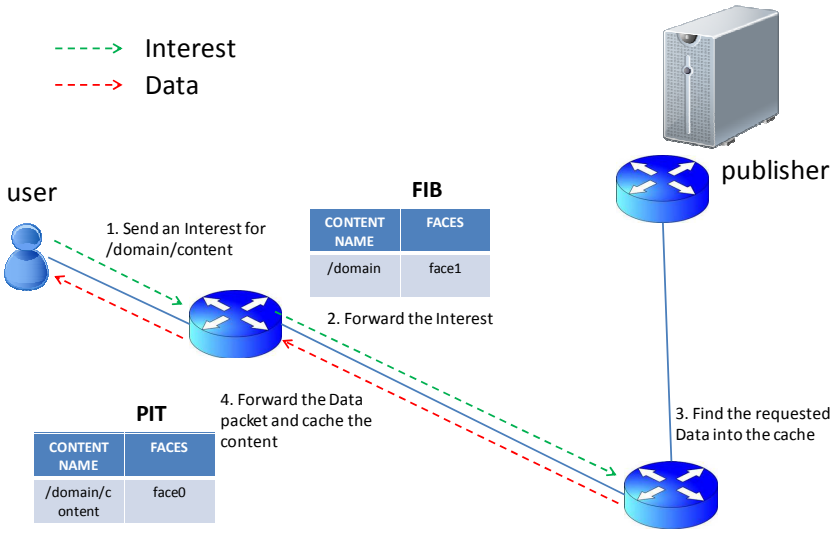

Fig. 1. An example showing NDN functionalities.

Other features are offered by the NDN architecture. For example, a NDN node may prohibit that its generated contents are cached, as well as a device sending the *Interest* packet could impose that the answer should be necessary generated by the publisher, thus bypassing any intermediate caches. Also, some security features, such as the encryption and the authentication of both *Content Name* (or a part of it) and the content stored in the *Data* packet are supported too [45] [47].

3 State of the art, literature review

3.1 The diffusion of real-time streaming services in social and crowdsourcing platforms

In what follows, we will describe some relevant works recently presented in literature, which focus on streaming services in social networks and *crowd-sourced* platforms. Moreover, a summary about the main aspects characterizing all the investigated works have been proposed in Tab. 3.

Incentive cooperation strategies for P2P live multimedia streaming social networks [48]. This paper focuses on live multimedia streaming services in P2P networks

Table 3. Summary of article presented in literature with topics they connect to

Work	Video streaming				Social networks			
	Tree-push node organization	Mesh-pull node organization	Real-time video & IPTV	Analyzing the bandwidth constraints	Crowdsourced tagging	Crowdsourced rating	User social connections exploitation	Using Game Theory derived incentives
[48]				x			x	x
[49]	x	x					x	
[51]				x			x	
[52]	x							x
[53]						x		
[55]						x		
[57]	x	x	x					
[60]		x		x				

.

under limited bandwidth conditions. In particular, it proposes a framework based on the game theory to model user behavior in a P2P network and designs incentive-based strategies to stimulate user cooperation in live streaming services.

Nettube: Exploring social networks for P2P short video sharing [49]. In this contribution, the adoption of clustering techniques is exploited to realize a P2P version of YouTube. According to the presented solution, users are grouped in clusters according to their interests. In each cluster, the designed solution try to store locally most requested videos, thus accelerating their downloading time. On the other hand, those videos that are unavailable in the cluster can be still recovered from the remote site. The conducted analysis demonstrates that this approach outperforms other ones, especially in the presence of short videos.

AMES-Cloud: A Framework for mobile social video streaming and sharing [51]. The authors realized a two part framework addressing the bandwidth limitations that can be found in mobile networks during the execution of video streaming services. The first part is a software agent that adapts, in real-time, encoding parameters based on currently available bandwidth. Its main goal is to dynamically avoid the loss of frames when the bandwidth is more limited and, at the same time, to properly increase the video bitrate when bandwidth availability goes up again due to better network conditions. The second part of the framework, instead, uses users' preferences matured in the social community to prefetch contents that have been recently viewed and rated.

Peer-Assisted social media streaming with social reciprocity [52]. This contribution tries to improve streaming services in P2P networks through the implementation of a credit system based on the user social relationship. To this end, it integrates game theory strategies and social group interactions.

Quantification of YouTube QoE via crowdsourcing [53]. This work focuses on the problem of evaluating QoE (Quality of Experience) of Youtube videosby crowdsourcing it. In that procedure, votes of users are adopted to generate the rating.

Recursive fact-finding: truth estimation in crowdsourcing [55]. The authors propose a recursive algorithm for user-submitted-facts verification in crowdsourcing, in the absence of a reputation system, based on other observations previously made by the same user. The approach is especially valid in real case scenarios as it works also with continuous, and thus ever-changing, streams of new data, compared to other algorithms only working on statical scenarios. While not directly related to video streaming, the authors points out that a useful use case could be validating crowdsourced user rating and tagging of video, a scenario where the speed and dynamical properties of the proposed algorithm, as well as the results quality, are very important.

A measurement study of a large-scale P2P IPTV system [57]. The performance evaluation of a large scale deployment of P2P IPTV, i.e., the PPLive system that is deployed and used mainly in China and Asia, has been presented in this work. The behavior of this network architecture has been also compared with respect to a typical TV set utilization patterns. The conducted analysis demonstrated that (i) users belonging to the P2P network have similar viewing behaviors as regular TV users, (ii) during a downloading session, a peer exchanges video data dynamically with a large number of peers, (iii) just a small set of peers act as video proxy and significantly contribute to the uploading process, and (iv) users of P2P IPTV system still suffer from long start-up delays and playback lags, ranging from several seconds to a couple of minutes.

PRIME: Peer-to-peer receiver-driven mesh-based streaming [60]. This contribution presents *PRIME*, i.e., a scalable mesh-based P2P streaming mechanism for live content. The discussed solution exploits a receiver-driven approach to fetch contents of a video stream and adopts an innovative mechanism to to minimize the bandwidth bottleneck among participating peers.

3.2 Real-time streaming services in NDN networks

During past few years, NDN 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 [61][62][63], congestion control issues [64], and routing strategies [65][66].

More recently, instead, thanks to the growing demand for multimedia services, some research activities are focusing the attention also to the management of multimedia applications in NDN [67]. The most interesting proposals

have been presented in [68]-[80], which deal to real-time voice transmissions, video on-demand services, and real-time streaming services.

As reported in Tab. 4, some works provide an analysis of performances of multimedia services offered through the NDN architecture (such as [73],[74] and [72]); whereas the other contributions suggest some extensions of the basic NDN idea, which refer to forwarding techniques, management/generation of contents requests, and CCNx messages, or define the structure of the *name-tree* which aims at implementing the conceived multimedia services.

Table 4. Summary of proposals presented in literature

Work	**Targeted multimedia service**			**Extensions of the basic NDN idea**			***Name-space* definition**
	VoIP	**Video on demand**	**Video real-time**	**CCNx messages**	**NDN router functionalities**	**New Interest management**	
[68]	x						x
[69]	x						x
[71]		x			x		
[72]		x					
[77]			x	x	x	x	
[78]			x	x	x	x	
[79]			x	x	x	x	
[70]		x					
[80]			x			x	
[73]		x					
[74]		x					
[75]		x					x
[76]		x	x				x

Anyway, since the present work targets real-time multimedia services, the discussion of the state of the art, provided in what follows, will focus the attention only to this kind of services.

Voice Over Content-Centric Networks (VoCCN) [68]. This service architecture supports a voice communication between two users. It supposes that

each user is identified by an unique *userId* and that it must announce its availability for the voice service by registering itself to a specific *namespace*. Moreover, a deterministic algorithm is used to generate *names* for identifying both users and contents during the execution of the service. At the beginning, the *service rendezvous* procedure is exploited to configured the VoIP session through the SIP protocol. The user that wish to initialize a call (i.e., the caller), maps a SIP INVITE message to an *Interest* packet that will be forwarded to the remote users (i.e., the callee). The *Content Name* of the first *Interest* packet is built by appending to the aforementioned *namespace* the content, eventually encrypted, of the SIP INVITE message. The callee will answer to this request by generating a *Data* packet containing the SIP response. From this moment on, the exchange of media contents is done using the RTP protocol. To each media chunk is assigned an unique sequence number and, in order to fetch quickly voice packets, the caller can generate and send multiple *Interest* packets at the same time. Every time a new *Data* packet is received, a new *Interest* is released, thereby restoring the total number of pending *Interests*.

Audio Conference Tool (ACT) [69]. It is a more complex architecture enabling audio conference services, which exploits the named data approach to discover ongoing conferences, as well as speakers in each conference, and to fetch voice data from individual speakers. A specific *namespace* and an algorithm to create *names* have been tailored to support all of these tasks. Before joining the conference, an user issues specific requests to collect information about the list of ongoing conferences and to known the list of speakers in a conference (i.e., the group of participants that produce voice data from which fetching *Data* packets). Similarly to VoCCN, each *Data* packet is identified by an unique segment number and an user could request multiple *Data* packets at the same time. Hence, whenever a new *Data* packet comes back, the user will issue a new *Interest*.

The MERTS platform [78]. It has been designed to handle at the same time real-time and non real-time flows in a NDN architecture. To this end, a new filed in the *Interest* message, e.g., the Type of Service (TOS), has been introduced to differentiate real-time and non-real-time traffics, thus allowing each NDN node to classify the type of service to which each packet belongs to. In addition, a flexible transport mode selection scheme has been devised to adapt the behavior of the NDN node according to the TOS associated to a specific packet. In order to serve real-time applications, the *one-request-n-packets* strategy is proposed, according to which the user issues a *Special Interest* (SI) asking for n consecutive *Data* packets. Such a request can be satisfied by more nodes inside the network that may store in their repository/cache one, more, or all requested chunks. When all the n chunks will be received by the user, a new SI is generated. To ensure that the end-to-end route will not be deleted after the reception of only a subset of chunks requested with the SI, the normal functionality of the PIT table is modified by imposing that the SI can be erased only after the expiration of its *life time*. Finally, with the aim of optimizing the memory utilization of the cache and

improving performances of non real-time services, the caching of real-time contents is completely disabled.

Adaptive retransmission scheme for real-video streaming [80]. This work proposes a novel and efficient retransmission scheme of *Interest* packets, which has been conceived to reduce video packet losses. In particular, the retransmission of requests not yet satisfied after a given timeout has been considered as an important technique able to offer a minimum level of reliability in NDN. From one side the lifetime estimation algorithm captures the RTT variation caused by the in-network caching and dynamically evaluates the value of the retransmission timeout. From another hand, the explicit congestion notification (ECN) field is added to the *Data* packet for signaling the ongoing network congestion to the end user. This information will be used by the retransmission control scheme to differentiates channel errors from network congestion episodes. Hence, based on the reason of packet losses, this algorithm adaptively adjusts the retransmission window size, i.e., the number of total *Interest* that can be retransmitted by the client.

Time-based interest protocol [77]. This proposal tries to avoid the waste of the uplink bandwidth due to the generation of multiple and contemporary *Interest*. To reach this goal, it introduces a new *Interest* packet, which is sent by the user in order to ask for a group of contents that are generated by the publisher during a specific time interval. During such time interval, all chunks generated by the remote server or transferred from other nodes can be delivered to the user. This novel scheme requires a modification of the normal behavior of a NDN router: the *Interest* packet should not be deleted from the PIT table until that its *life time* will expire.

The NDNVideo architecture [76]. It has been designed and implemented on top of CCNx, in order to offer both real-time and non real-time video streaming services. A first important issue addressed by the NDNVideo project is the design of the *namespace* that enables the publisher to uniquely identifying every chunk of the multimedia content and allows the consumer to easily seek a specific place in the stream. In particular, the *Content Name* is built in order to provide information about the video content, the encoding algorithm, and the sequence number associated to a given chunk. Moreover, to facilitate the seeking procedure, the user can specify in its first request a timecode that will be used by the server to select the most suitable *Data* packet within the video stream. Then, after the reception of the first *Data* packet, the user will ask for video data using consecutive segment numbers. To supports real-time streaming services, the client may issues multiple *Interest* packets at the same time. However, to avoid that it will fetch data too quickly and request segments that do not yet exist, the client estimates the generation rate of *Interest* packets by knowing the time at which previous data packet ware generated by the publisher (this information is stored in a specific field of the *Data* packet). If data are not received fast enough to playback the video at the correct rate, the client may skip to the most recent segment, thus continuing to see the video from there instead of pausing the playback. Finally, a low pass filter similar to the one defined in the TCP

protocol is adopted to adjusts the retransmission timeout based on previous RTT values.

4 Problems, issues and challenges identified

The present contribution focuses the attention on the design of a network architecture able to distribute, in real-time, users generated contents (e.g., multimedia flows) within a social community. In particular, we consider a scenario in which a group of users captures and transmits multimedia contents to a second group of consumers diffused worldwide.

We highlight that such a scenario embraces a number of significant use-cases, including those reported below:

- real-time broadcasting of social events when users capture media contents from different point of views and by using different technologies, encoding schemes, and resolutions;
- virtual tourism with which users visiting monuments (or other kind of tourist attraction) would share their experiences with other ones;
- real-time broadcasting of activities and environmental conditions during danger situations (as described in [81]);

The key aspect characterizing our idea is that multimedia contents are not provided by a a media server, but they are shared by a (potentially large) number of users. This scheme fully reflects the *crowd-sourcing* paradigm, which has been presented in the Sec. 2.

In our opinion, the design of an network architecture able to efficiently distribute *crowd-sourced* multimedia contents within a social community spread around the world is flanked by the born of a number of issues and challenges that need to be carefully investigated.

4.1 Issues arising from the considered scenario

First of all, when thinking of crowd-sourced media streaming, the issue of scalability is the first one to come to mind. In a canonical media portal, where there is only one content generator, the content provider needs to maintain an infrastructure capable of serving contents to a vast number of consumers. Generally, a Content Delivery Network (CDN) architecture can be exploited to maintain multiple and parallel multimedia sessions and to efficiently handle the distribution of media contents. In a CDN, in fact, contents are replicated among many servers, which are strategically located in different places, thus allowing users to seamlessly connect to the closest server [82][83]. However, in this network architecture, consumers themselves do not necessarily generate a huge upload traffic, being mostly responsible of posting text comments, ratings and other operations less bandwidth consuming. In the case of crowd-sourced media services, where a group of users is in charge of producing and sharing multimedia contents, high computational and bandwidth capabilities are not more required for a static

infrastructure of servers (like in the previous case), but they need to be distributed among users themselves. Unfortunately, in a scenario in which people participate to an event, it is impossible to guarantee these requirements. From one side, in fact, users generate contents through devices with scarce computational capabilities (i.e., smartphones and tablets). On the other hand, instead, they could be connected to the network by means of a link with limited capacity. Based on these considerations, it is necessary to conceive a service architecture that should be able to reduce, as more as possible, the traffic load managed by content producers, while ensuring a good level of the quality of service to other users of the social community that want to download media streams.

Another important issue refer to the availability of contents. In these years, in fact, we are assisting to a radically changing of the way digital resources are used: users are interested to share contents rather than to interact with remote devices [45]. This means that they need to fetch contents in a fast, reliable, and effective way, without knowing their location a priori. In other words, it is not more necessary to identify a media stream through the IP addresses of the device that provides it but using an unique *name*. Accordingly, the service platform should enable the classification, the retrieving and the delivering of media contents, just exploiting their names.

The seamless support of mobile users, which should not experience any service interruption when moving across different access networks, represents another relevant aspect to consider.

Finally, to conclude the discussion on issues arising from the considered service, other minor aspects that deserve our attention are: the possibility to trust contents independently from the location and the identity of who is providing them (security issue) and the capability to guarantee the resilience of ICT services to system failures (fault-tolerance issue).

4.2 Towards an ICN-based solution and related challenges

It is very important to remark that it is widely recognized that the current Internet architecture is not able to efficiently face all of the aforementioned issues [4][45][84].

Fortunately, the emerging ICN paradigm seems to have all the potentials to offer novel network architectures for the Future Internet, which will be able to overcome these problems [3]. In fact, among other novel and interesting features, the ICN approach provides (i) the addressing of contents through names, (ii) a faster content distribution due to the adoption of in-network caching strategies, (iii) the delivering of user demands through *routing-by-name* approaches, (iv) a simplified management of mobility, and (v) a native support for the security [84].

In line with this premise, we decided to found our proposal on the NDN network paradigm, which represents, as anticipated in Sec. 2.4, one of the most important ICN architecture presented in literature.

Obviously, the design of a NDN-based platform is not a simple task to accomplish, because of the presence of several challenges. In what follows we will

mention the most important ones for which our proposal intends to find a solution:

- *Namespace definition.* NDN architecture defines the resources exchanged in the systems via a URI notation. When designing a NDN-based application, the first step consists of defining what kind of data will be exchanged and how that data will be represented as a URI. In this regard, the design of the *name space* is a crucial activity that will impact on the performance of the entire system. In order to simplify and ameliorate routing operations and packet processing, it is necessary to conceive a hierarchical *name tree*, which will be able to efficiently map all the data that can be exchanged among nodes in the network .
- *Identification of events and media contents.* In the considered scenario it is important to devise a way to track and identify the list of active events and the set of available multimedia contents (i.e., those captured by users of the social community that are participating to a given event). According to the NDN paradigm, this challenge has to be addressed through a completely host-less protocol, which should allow any user to download a media stream for a given event by adopting NDN primitives.
- *Management of content requests.* According to the NDN paradigm, audio and video contents should be divided into a list of consecutive chunks that should be requested by the client during the service execution. Unlike Video-On-Demand, the distribution of a real-time stream has to deal with a specific class of problems to ensure the timely delivery of an ordered stream of chunks. Video and audio chunks, in details, have to be received in playing order and within a given time interval (the *playout delay*), before they are actually played, thus "expiring". A chunk not delivered before its expiration will result in degradation of the rendered video, impacting the end user Quality of Experience (QoE). To solve these challenges, client nodes implement a receiving buffer queue where the chunks are stored in order, that is emptied while the video is being played. Therefore, any chunk not received before its playing instant becomes useless. To reduce the chance of chunk loss, an efficient mechanism that control the retransmission of user's requests (e.g., requests for chunks close to expiration and not yet received).

We highlight that these challenges have been only partially investigated in literature and, at the same time, solutions dedicated to the envisaged scenario have never been discussed. This increases the novel nature of our contribution.

5 Proposed approach and solutions

The main goal of the service architecture described in this book chapter is to discover, organize, store, process, and deliver media contents, which are captured according to the *crowd-sourcing* paradigm, within a social community spread around the world. Such an architecture has been conceived as an extension of our previous contributions discussed in [79] and [85].

As shown in Fig. 2, the conceived platform is composed by a set of entities, which have been summarized below:

1. a community of users that, being into the same place to take part to an event (i.e., concert, football match, sightseeing, city accident, and so on) record and broadcast it from their multiple points of view;
2. a number of users interested to see what the aforementioned social community is capturing;
3. a distributed *Event Management System*, which creates events and handles the social community;
4. a NDN-based communication infrastructure able to efficiently manage user's requests and distribute multimedia contents.

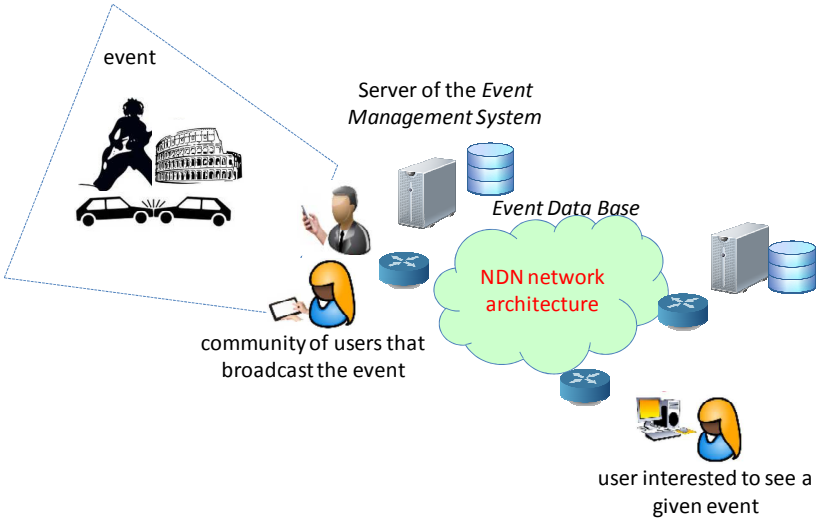

Fig. 2. Network architecture of the conceived platform.

The *Event Management System* covers a crucial role because it is in charge of registering of users that wants to provide multimedia contents for a given event and provisioning of information about available event to other users that request them. It is composed by a number of dedicated servers distributed within the NDN infrastructure. To efficiently track all the active events, each node of the *Event Management System* uses a *Event Data Base* to store details about events, such as the name, the location, and the duration. Moreover, the list of users that participate to an event, which represent those that capture and broadcast media contents, is saved too. For each of them, the data base will contain some parameters that are typically shared within a social community (like nick name, age, location, work, and so on), as well as the *media capabilities*

specifying the type of device, the encoding algorithms, the resolution, and the encoding bit rate adopted by the user to generate video and voice data.

Therefore, the user that would transmit its media stream, i.e., the *publisher*, has to register itself to the *Event Management System*. On the other hand, an user that wish to see and listen these contents, i.e., the *client*, should obtain all the information related to a specific event by interrogating the *Event Management System* through NDN primitives.

Hence, our platform addresses four different tasks: *event announcement*, *event discovering*, *media discovering*, and *media delivering*. Moreover, to efficiently support these operations, a *name space* has been properly designed to classify and map all the services activities in an ordered and hierarchical manner. According to NDN specifications, this would also provide enormous advantages for routing operations and packet processing.

5.1 Design of the *Name space*

The definition of the *name tree* is a key aspect of a NDN network because the *Content Name* field strongly influences the way *Interest* and *Data* packets are treated.

In line with theoretical suggestions presented in [68], the *name space* we conceived herein is based on both *contractable names* and *on-demand publishing* concepts. The *contractable names* approach identifies the ability of an user to construct the name of a desired content through specific and well-known algorithms (e.g., it knows the structure of the name tree and the value that each field of the *Content Name* may assume). The *on-demand publishing* criteria, instead, defines the possibility to request a content that has not yet been published in the past, but it has to be created in answer to the received request (this may occur very often during a real-time communication).

The *name tree* structure adopted in our platform is:

/domain/ndn_streaming/activity/details

Starting from the the root tree, which is identified with "/", we introduced the *domain* field in order to explicitly indicate the domain in which the service is offered. The second field adopts the keyword *ndn_streaming* to explicit the considered service (i.e., the streaming of a video content over the conceived NDN-based platform). The *activity* field specifies the task to which the *Interest* or *Data* packet belongs to. As anticipated before, it may assume four different values. i.e., *event_announcement*, *event_discovering*, *media_discovering*, and *media_delivering*. Finally, the latest field, i.e., *details*, is used for appending to the *Content Name* specific values that can be exchanged among nodes during the service execution.

5.2 Event announcement

An user that wants to provide multimedia contents for a specific event should register itself and announce its *media capabilities* to the *Event Management*

System. To this end, it firstly sends a *Interest* packet asking for the list of events active within a given location (i.e., the geographical area in which the user is located). The *Content Name* of this request is set to:

/domain/ndn_streaming/event_announcement/event-list/location/ .

This message will reach the closest node (i.e., a server of the *Event Management System* or a NDN router storing this data within its cache) able to provide these information, which will answer with a corresponding *Data* packet.

Then, the user will generate a new *Interest* packet through which it will conclude the registration. The*Content Name* of this new request will contain, in the *details* field, the position of the user and its *media capabilities*, as reported in the sequel:

/domain/ndn_streaming/event_announcement/

registration/event/position/media_capabilities/ .

This message will be processed by the *Event Management System* that will update the *Event Data Base* and will answer with a *Data* packet of confirmation.

5.3 Event and media discovering

In order to discover events in a given geographical area, an *Interest* packet is released with a *Content Name* set to:

/domain/ndn_streaming/event_discovering/location .

This message will be routed towards the first node in posses of this information, which will respond with a *Data* packet containing the requested information.

Once the user has identified the event of its interest, he will retrieve the list of available multimedia contents, together with *media capabilities* of which provide them, by sending an *Interest* packet with the *Content Name* equal to:

/domain/ndn_streaming/media_discovering/event .

To ensure that this request will be handled by only a node of the *Event Management System*, which is the only device storing updated information, the *PublisherPublicKeyDigest* and *AnswerOriginKind* fields of the *Interest* packet contain the hash function of the public key of the *Event Management System* and the numerical value 3, respectively. The corresponding *Data* packet, generated in answer to this request, will allow the user to select its preferred content among those available. From this moment on, the user can start fetching multimedia content from a specific source.

With the aim of designing a platform more flexible as possible, we assume also that the user can change the source of the media stream during the time. To better support this feature, it is necessary to send periodically the aforementioned *Interest* packet, thus being able to have always updated information about the available multimedia contents.

5.4 Media delivering

The media delivering process consists of a channel bootstrap phase, a flow control strategy, and an efficient mechanism for retransmitting *Interest* packets. It is performed after that the user has selected the event of its interest, i.e., the *event_name*, and the media source, identified through its social nick name (i.e., *source_nick_name*), from which fetching media data. To enable 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.

As detailed in [79] and [85], bootstrapping a channel involves the operations of finding the first valid chunkID of the video stream. Due to video codec requirements, the video stream can be visualized only once the first I-frame has been received. Therefore, a client has to first gather from the server the first chunk (and the corresponding chunkID) of the last generated I-frame. To do so, it sends an *Interest* packet in which a timecode (e.g., HH:MM:SS:FF) is appended to the the *Content Name* (this approach has been already introduced in [76]):

/domain/ndn_streaming/media_delivering/

event_name/source_nick_name/HH:MM:SS:FF .

The *Status* field set to *BOOTSTRAP* and the *Nonce* field set by the client. An *Interest* with Status = BOOTSTRAP would travel unblocked until it reaches the first good stream repository (i.e. a node who can provide a continuous real-time flow of chunks, not just cached ones).

From the moment a node receives the bootstrap data message, it can initiate the sliding window mechanism to request the subsequent chunks in an optimal way. Each chunks of the video stream is identified with an unique $chunkID$, as detailed in what follows:

/domain/ndn_streaming/media_delivering/event/source_nick_name/chunkID .

Each node has a windows of size W to store W pending chunks. We define *pending chunk* a chunk whose *Interest* has been sent by the node, and the window containing the pending chunks a *Pending Window*. Together with the chunkID, we store in the pending window other information, such as the timestamp of the first request and the timestamp of the last retransmission. Whenever a new data message is received, the algorithm described in Fig. 3 runs over the Pending Window, to perform the following operations:

1. Purge the Pending Window from all the chunks who are expired, i.e., who have already been played, to free new space in the sliding window.
2. Retransmit all chunks that have not been received for a given timeout (onward denoted as *windowTimeout*.
3. Transmit, for each slot that got freed by the received or expired chunks, the *Interest* for a new one.

```
1: procedure SendInterests(PW, W, WinT, Now, LC)
2:     # Remove all expired Interest
3:     for all CID in PW do
4:         if CID is expired then
5:             remove CID from PW
6:         end if
7:     end for
8:     # Resend stale Interests
9:     for all CID in PW do
10:        if lastTx(CID) < Now − WinT then
11:            resend(Int(CID))
12:            lastTx(CID) ← Now
13:        end if
14:    end for
15:    # Send Interests for new chunk
16:    NNC ← W − size(PW)
17:    for i ← 1, NNC do
18:        send(Int(LC))
19:        lastTx(LC) ← Now
20:        add LC to PW
21:        LC ← LC + 1
22:    end for
23: end procedure
```

Fig. 3. Sliding window algorithm

Furthermore, the same operations are performed if a node doesn't receive any data for at least *windowTimeout* seconds, in which case, all the *Interests* for non-expired chunks in the Pending Window are retransmitted, together with new chunks if new slots have been freed due to expired chunks.

Fig. 3 details the implemented algorithm; for the purpose of brevity and readability, the variable names have been contracted: PW is the Pending Window, W is the aforementioned system parameter, indicating how many *Interests* should a node have ongoing, $WinT$ is the window timeout, after which *Interests* in the Pending Window are resent, $Int$is a new *Interest* message, CID is a $chunkID$ in the pending window, $lastTx$ is the transmission time of the most recent $Interest$ for a given $chunkID$, LC is the $chunkID$ of the most recent requested chunk and NNC is the number of new chunks to request, after the pending window has been purged.

To provide a further insight, we reported in Fig. 4 an example of the conceived sliding window algorithm, in which we have set the value of W to be equal to 3.

As described in Sec. 2, NDN nodes along the routing path of an *Interest* will stop the propagation of said *Interest*, if they have previously routed another *Interest* for the same resource, and the correspondent data has note been sent back yet; instead, they will simply update their Pending Interest Table adding the face from where this newcomer *Interest* was originated, so to reroute the data back recursively along the path the *Interest* has gone through.

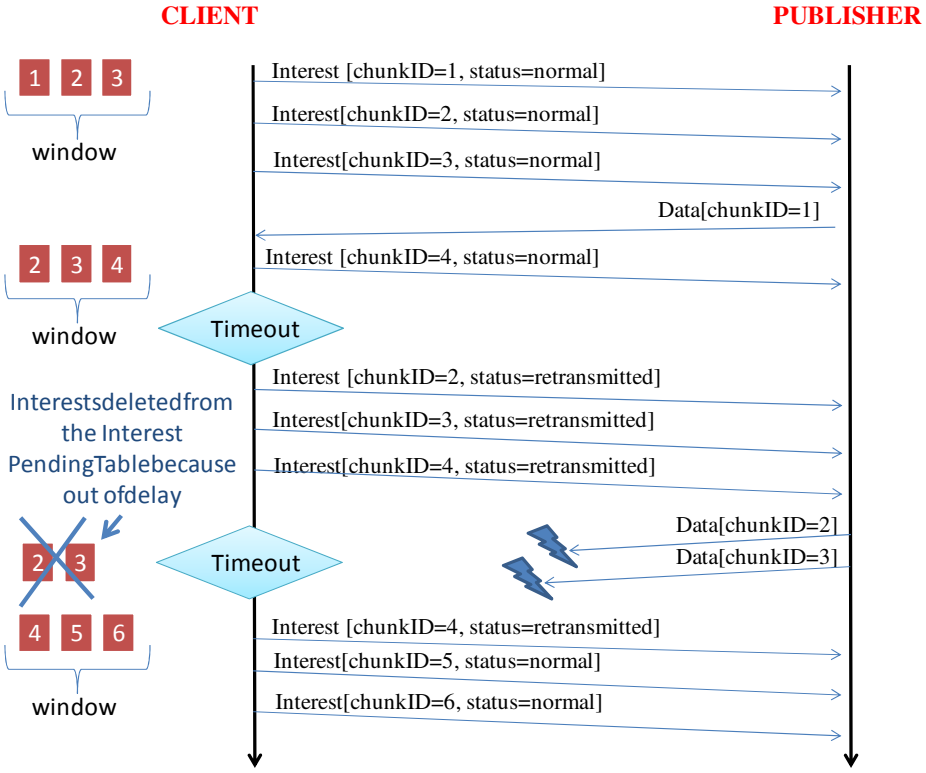

Fig. 4. Sliding window example

This mechanism ensures flow control and limits the propagation of duplicate *Interests*, in case several nodes in the same network are watching the same channel.

However, to make the *Interest* retransmission mechanism effective, a retransmitted *Interest* needs to propagate all the way up to the server, or to the first node with the desired chunk in cache. Therefore, retransmitted *Interests* carry the *Status* field set to *Retransmission* to mark if the *Interest* is a retransmission or not, and each node along the routing path propagates the *Interests* marked as retransmitted, thus skipping the usual NDN mechanism, unless the correspondent chunk is found in the cache.

6 Evaluation of the proposed approach

The *ccnSim*, i.e., an open source and scalable chunk-level simulator of NDN [7] built on top of the Omnet++ framework [86], has been exploited to evaluate performances of the conceived solution under different network configurations.

6.1 Simulation platform

By itself, *ccnSim* models a complete video distribution systems, with a high degree of fidelity concerning catalogs, requests and repositories distribution, and network topologies. Unfortunately, in its original version, *ccnSim* does not support real-time video transmissions. In our previous contribution, i.e., [79], we have already extended this simulator to implement real-time constraints required by our evaluations. In this work, we have further extended the implemented features, thus being able to model all the facets characterizing the distribution of *crowd-sourced* media contents within a social community through a NDN network infrastructure.

With respect to the original version of *ccnSim*, we introduced the following new features:

- a support for links with bounded capacity and packets with a well defined size, which was missing in *ccnSim*, to be able and estimate the NDN behavior under some bandwidth constraints;
- a transmission queue for each face of each node, in order to properly manage the packet transmission under constraints due to the data-rate of channels,
- the support for synthetic video traces, so to be able to transmit and receive chunk of real videos, and consequently being able to reconstruct the received video and evaluate its Peak Signal-to-Noise Ratio (PSNR);
- a cleanup mechanism for each node's PIT, to avoid having in long term stale entries due to expired chunks;
- an improved logging system, so to be able to record each node's received chunks and reconstruct the received video;
- more controls server-side, to send a data only for those chunks who have already been generated.
- the sliding window mechanism described above, and all the related data structures;
- the *Interest* forceful propagation in case of retransmission;

6.2 Network configuration and parameters

In our tests, we considered a complex network architecture composed by 68 routers connected among them according to the Deutsche Telekom topology. Every node of the network is considered to be a direct NDN node, i.e. no TCP or UDP encapsulation is implemented.

We assume the presence of only one event where participate a number of user that generate video flows with different encoding characteristics. In particular, without loss of generality, in every simulation round, each video content is mapped to a video stream compressed using H.264 [87] at a average coding rate randomly chosen in the range [250, 2000] kbps. We suppose also that these users are connected to the network through the same access point, which is identified by one router of the aforementioned topology, randomly chosen every run among available ones. On the other hand, clients of the social community, that are interested in downloading video contents generated by the previous group of users,

are connected to remaining nodes (1 client per node). Further, the selection of a media stream has been modeled considering that contents popularities follow a Zipf distribution, which is commonly adopted for user generate contents [88].

In our study, we adopted the optimal routing strategy, already available within the *ccnSim* framework. According to it, *Interest* packets are routed towards router to which are attached users that generate video data along the shortest path. On the other hand, three caching strategies have been considered in our study: *no-cache*, *LRU*, and *FIFO* [89]. When well known *LRU* or *FIFO* policies are adopted, we set the size of the cache to 10000 chunks. The *no-cache* policy is intended to evaluate the performance of the NDN 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 a reference configuration.

Once a client selects the video content of its interest, it performs the bootstrap process described in the previous section and then starts sending *Interest* packets following the designed sliding window mechanism. The window size W has been set to 10, ensuring that faces of the server are almost fully loaded in all considered scenarios. Also, the transmission queue length associated to each face, Q, has been set, in order to be larger than

$$Q = 2 * L_c * P_D, \tag{1}$$

where L_c and P_D represent link capacity and maximum propagation delay in the considered network topology.

The aim of our study is to evaluate the designed NDN-based service architecture by varying the amount of the network bandwidth dedicated to real-time streaming services (set in the range [50-100] Mbps), the *playout delay* (chosen in the range [10-20] s), the *windowTimeout* (chosen in the range [1/10-1/2] of the *playout delay*), the number of users participating to the event (set in the range [10-100]), and the popularity of available contents (in line with [89], α has been chosen in the range [1-1.5]). In addition, each simulation lasts 600s and all results have been averaged over 15 simulations.

To conclude, all simulation parameters have been summarized in Tab. 5.

6.3 Performance evaluation

The first important parameter that describes how the parameter settings affect the quality of service offered to end users is 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.

Figs. 5 and 6 show the chunk loss ratio measured in all considered network scenarios, when α, i.e., the parameter of the Zipf's law, has been set to 1 and 1.5, respectively. From reported results, several conclusions can be drawn. Fist of all, it emerges that a reduction of the link capacities leads to a higher number of

Table 5. Summary of simulation parameters

Parameter	**Value**
Topology	Deutsche Telekom with 68 routers
Link capacity	50 Mbps and 100 Mbps
Number of active events	1
Number of user generate contents	10, 20, 50, 100
Number of clients	67
Chunk size	10Kbytes
Video average bit rate	250kbps, 600kbps, 1000kbps and 2000kbps
W (window size)	10
Playout delay	10 s and 20 s
Window timeout	1/10, 1/5, and 1/2 of the playout delay
Protocol configuration	No cache, LRU, FIFO, *baseline scenario*
Cache size	10000 chunks
Simulation time	600 s
Number of seeds	15

lost chunks, due to increased latencies induced by network congestion. Moreover, we note that *playout delay* plays a fundamental role. When the *playout delay* increases, in fact, the client could receive a *Data* packet within a longer time interval, thus reducing the amount of chunks discarded because out of delay. On the other hand, a slight increment of the chunk loss ratio can be registered by increasing the *windowTimeout*. If the client retransmits an *Interest* packet after long time, there is the risk that the *Data* packet will be reached by the destination after the expiration of the *playout delay*. This result is more evident when the *playout delay* is set to 10s, which defines more strict constraints on the packet's delivery.

It is very important to remark that the number of users that generate multimedia contents influences significantly the performance of the proposed architecture. In particular, we found that the higher is the amount of available video contents, the higher is the registered chunk loss ratio. The reason is that when the number of available video increases, users may request different contents, thus growing the traffic load generated in the network. Obviously, the content popularity plays a crucial role in this case: worst performances are observed in scenario when a group of contents have the highest popularity (i.e., $\alpha = 1$).

By handling unicast communications, the *baseline scenario* generates the highest network congestion level, thus registering the worst performances. Thanks to the adoption of caching policies and the PIT table NDN provides a native support for multicast communication. In the considered scenario, where some users may request the same video content at the same time, this feature notably improves final performances. However, in our tests we realized that the presence of the cache can guarantee only a small reduction of the chunk loss ratio. In the presence of real-time flows, in fact, the cache does not represent an important NDN feature. On the other hand, we noticed that the PIT plays a more relevant role. In presence of live video streaming services, clients that are connected to a

channel request the same chunks simultaneously. In this case, a NDN router has to handle multiple *Interest* messages that, even though sent by different users, are related to the same content. According to the NDN 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.

To provide a further insight on this aspect, we also reported in Figs. 7 and 8 the percentage of *Interest* packets sent by users and directly received by the publishers. 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. With this very important finding, we demonstrate how the adoption of a NDN-based network infrastructure represents a winner solution to efficiently distribute *crowd-sourced* contents within a social network.

To conclude our study, we have computed PSNR, which is nowadays one of the most diffused metrics for evaluating user satisfaction, together with interactivity level, in real time video applications [90]. Results shown in Figs. 9 and 10 are in line with those reported for chunk loss ratio (the PNSR is higher in the same case in which the chunk loss ratio is lower).

7 Conclusions and future works

In this book chapter we have presented a NDN-based platform providing *crowd-sourced* and real-time media contents within an online social community. The performances of the conceived solution have been evaluated through computer simulations, carried out with an our extended version of the *ccnSim* simulator. The conducted analysis have clearly highlighted the impact that the number of media sources, the content popularity, the amount of network bandwidth dedicated to real-time services, the *playout* delay, and the network configuration have on the quality of service offered to end users. Among all the obtained findings, we demonstrated 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. In the future, we will plan to test the devised service architecture in real test-beds, thus evaluating its pros and cons under more realistic network settings.

References

1. Cisco: Cisco visual networking index: Forecast and methodology, 20122017. White Paper (May. 2013)

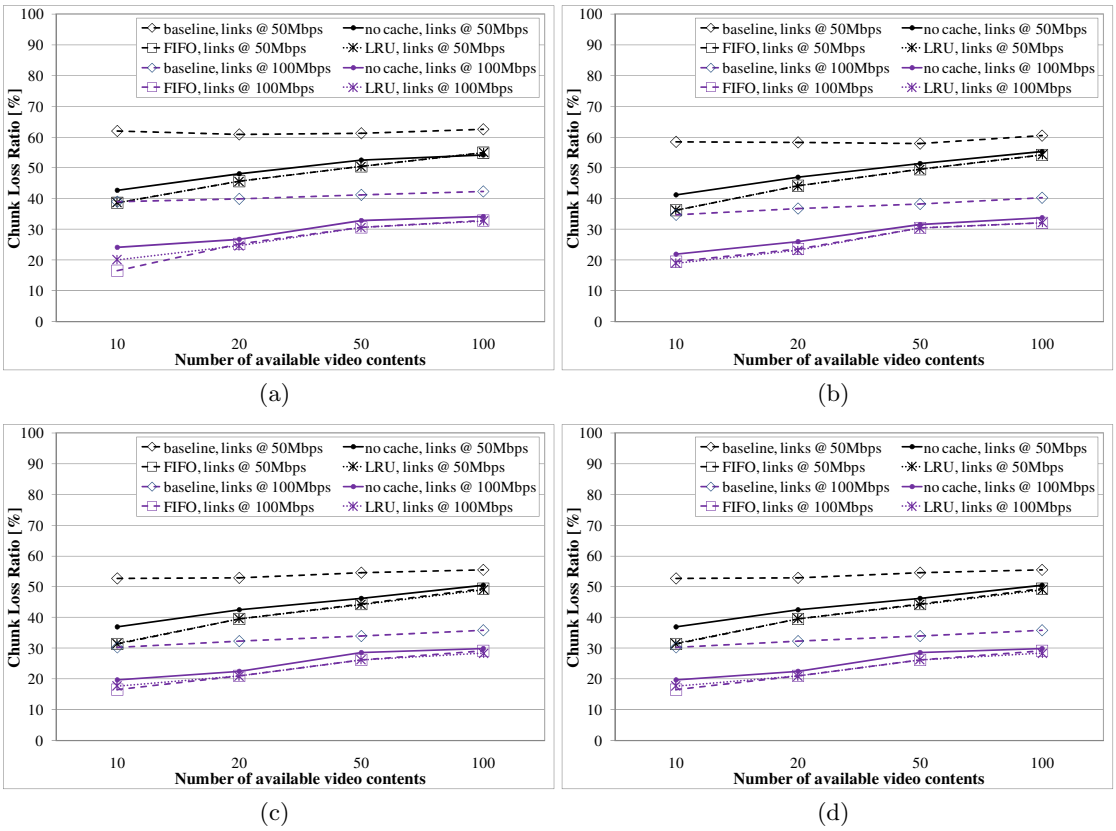

Fig. 5. Chunk loss ratio (scenario with $\alpha = 1$) when the PD and the $winT$ have been set to (a) 10s, and 1/10 PD, (b) 10s, and 1/2 PD, (c) 20s, and 1/10 PD, (d) 20s, and 1/2 PD, respectively.

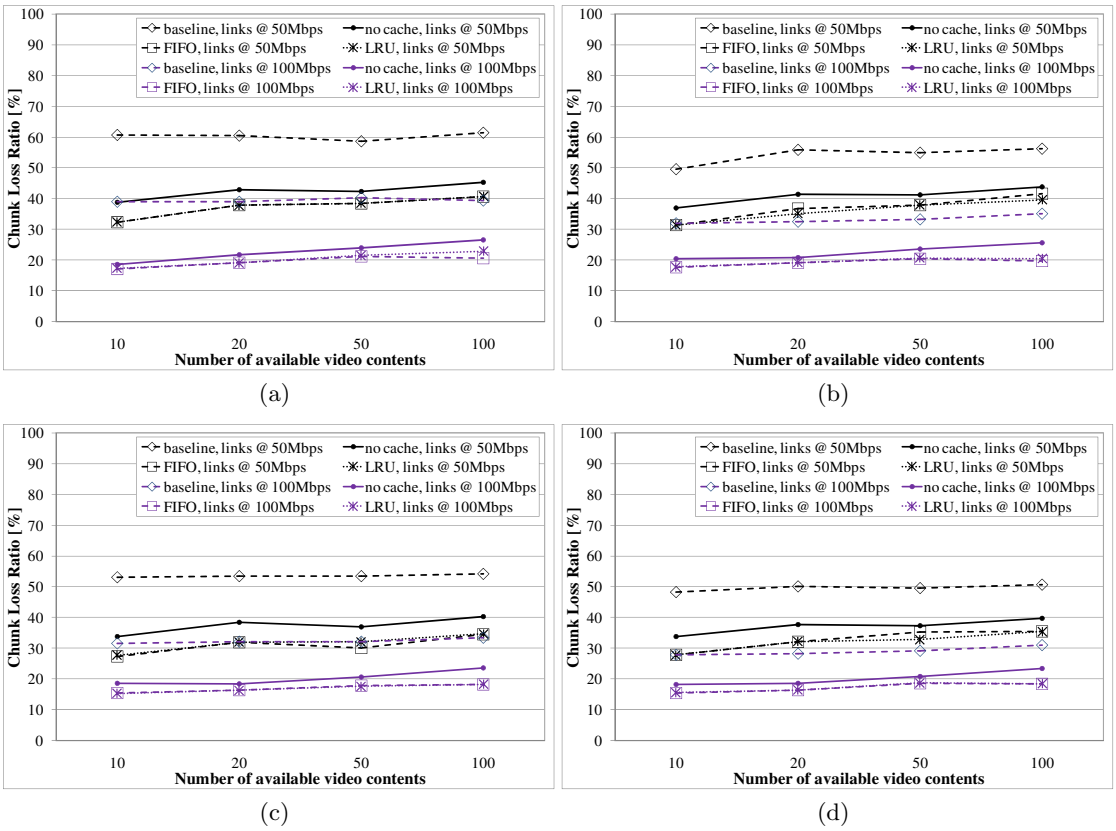

Fig. 6. Chunk loss ratio (scenario with $\alpha = 1.5$) when the PD and the $winT$ have been set to (a) 10s, and 1/10 PD, (b) 10s, and 1/2 PD, (c) 20s, and 1/10 PD, (d) 20s, and 1/2 PD, respectively.

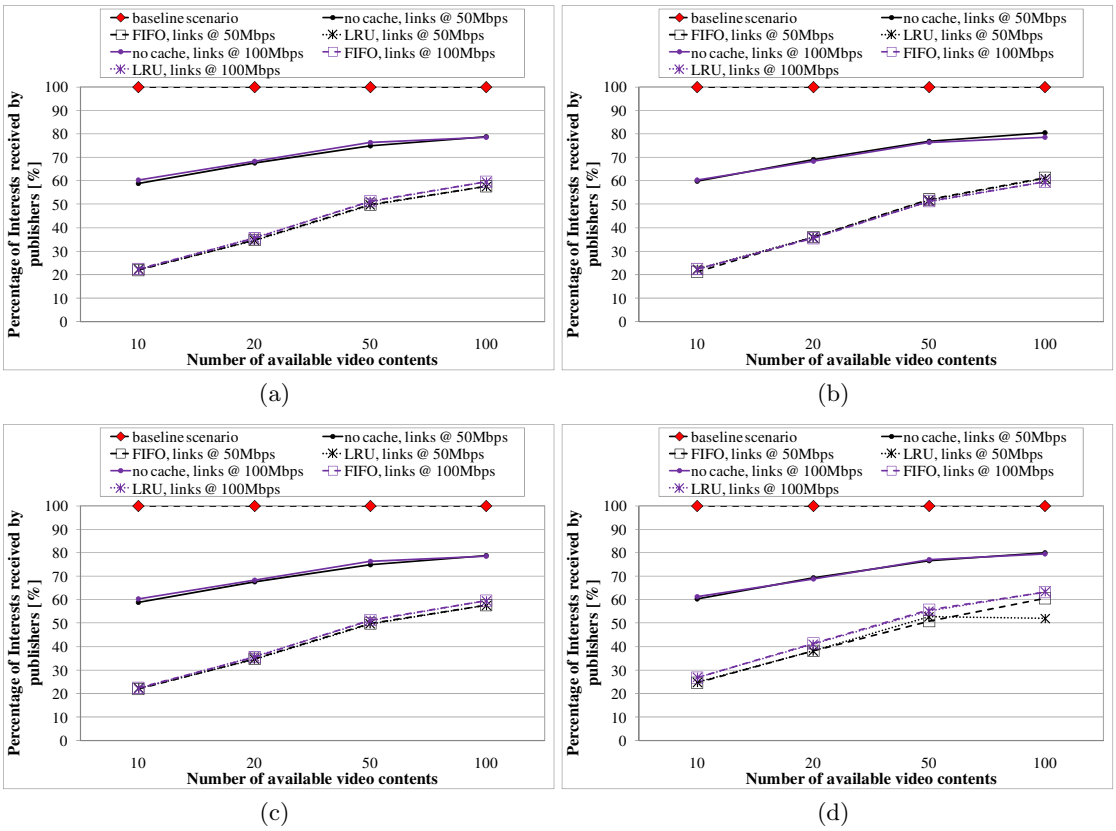

Fig. 7. Percentage of interests received by publishers (scenario with $\alpha = 1$) when the PD and the $winT$ have been set to (a) 10s, and 1/10 PD, (b) 10s, and 1/2 PD, (c) 20s, and 1/10 PD, (d) 20s, and 1/2 PD, respectively.

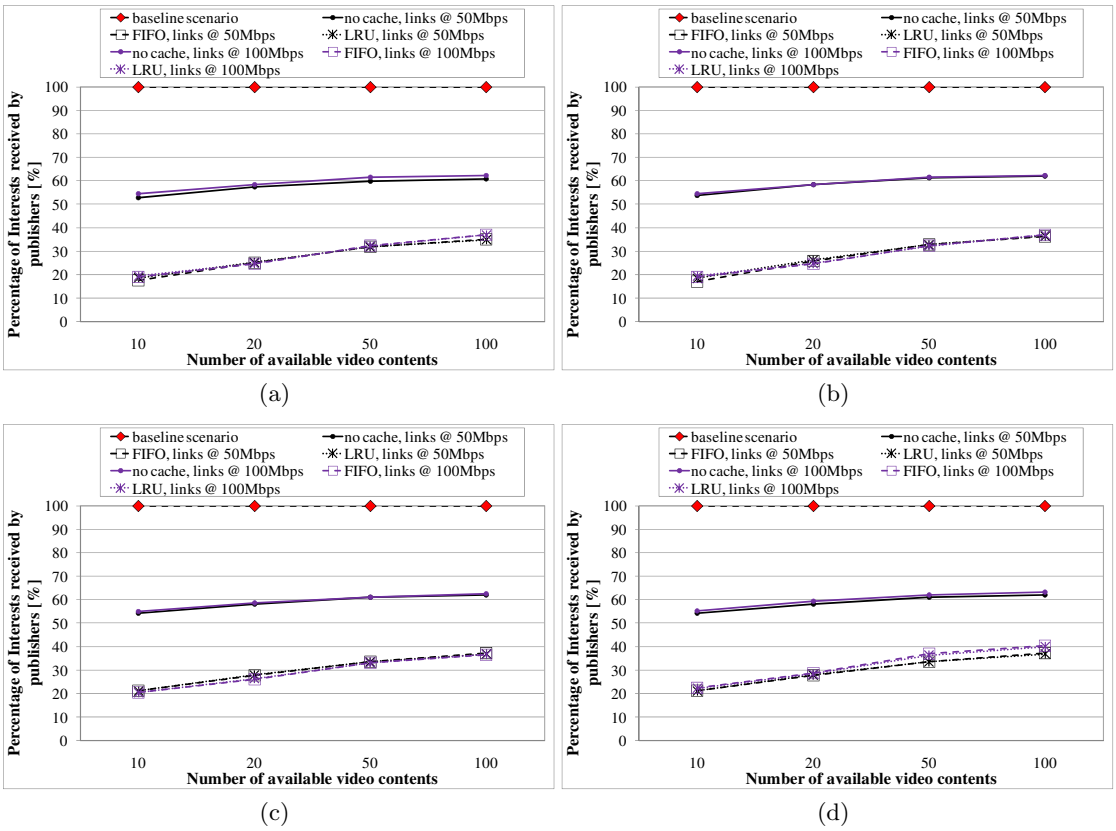

Fig. 8. Percentage of interests received by publishers (scenario with $\alpha = 1.5$) when the PD and the $winT$ have been set to (a) 10s, and 1/10 PD, (b) 10s, and 1/2 PD, (c) 20s, and 1/10 PD, (d) 20s, and 1/2 PD, respectively.

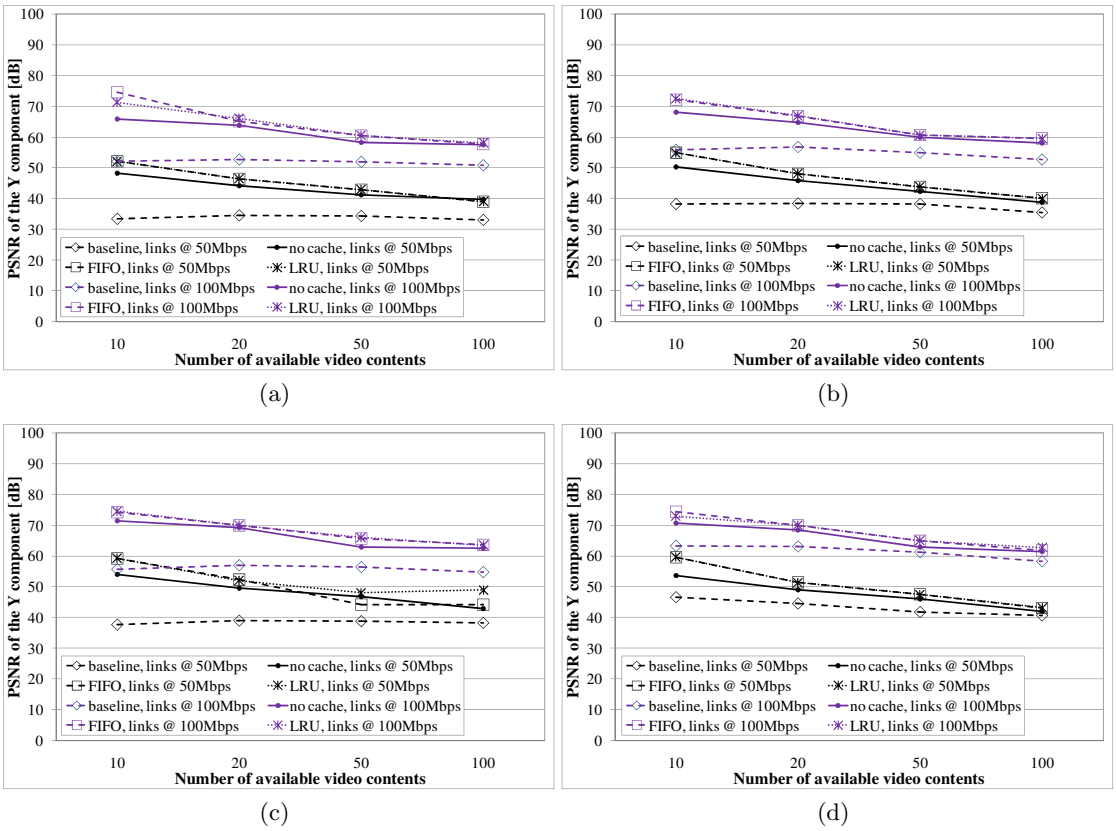

Fig. 9. PSNR of the Y component (scenario with $\alpha = 1$) when the PD and the $winT$ have been set to (a) 10s, and 1/10 PD, (b) 10s, and 1/2 PD, (c) 20s, and 1/10 PD, (d) 20s, and 1/2 PD, respectively.

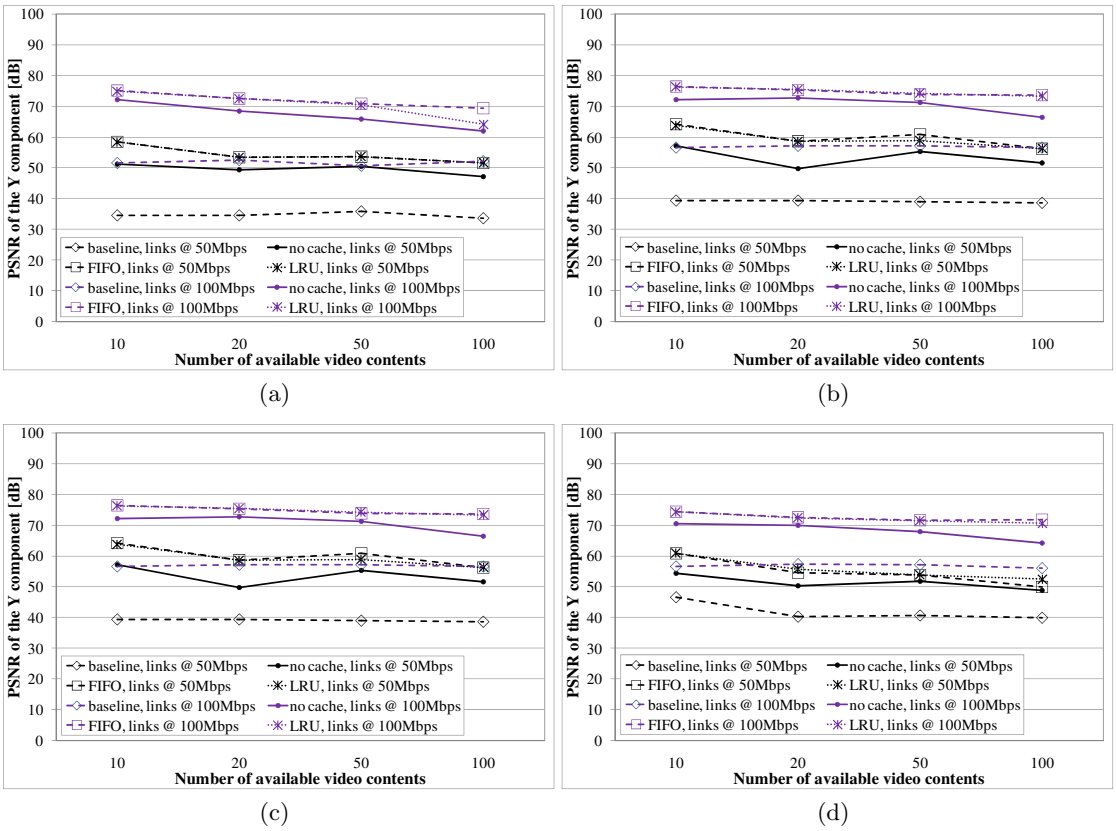

Fig. 10. PSNR of the Y component (scenario with $\alpha = 1.5$) when the PD and the $winT$ have been set to (a) 10s, and 1/10 PD, (b) 10s, and 1/2 PD, (c) 20s, and 1/10 PD, (d) 20s, and 1/2 PD, respectively.

2. Chatzimilioudis, G., Konstantinidis, A., Laoudias, C., Zeinalipour-Yazti, D.: Crowdsourcing with smartphones. IEEE Internet Computing **16**(5) (2012) 36–44
3. Matsubara et al., D.: Toward future networks: A viewpoint from ITU-T. IEEE Commun. Mag. **51**(3) (2013) 112–118
4. Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., Ohlman, B.: A survey of information-centric networking. Communications Magazine, IEEE **50**(7) (july 2012) 26 –36
5. Xylomenos et al., G.: A Survey of Information-Centric Networking Research. IEEE Surv. & Tuts. **PP**(99) (2013) 1–26
6. Bari et al., M.: A survey of naming and routing in information-centric networks. Communications Magazine, IEEE **50**(12) (Dec. 2012) 44–53
7. Rossini, G., Rossi, D.: Large scale simulation of ccn networks. In: In Algotel 2012. (2012)
8. MySpace: http://www.myspace.com
9. Facebook: http://www.facebook.com/
10. Weibo, S.: http://www.weibo.com/
11. Orkut: http://www.orkut.com/
12. Hi5: http://www.hi5.com/
13. NK: http://www.nk.pl/
14. VKontakte: http://www.vk.com/
15. Twitter: http://www.twitter.com/
16. Thumblr: http://www.thumblr.com/
17. Identi.ca: http://www.identi.ca/
18. Vine: http://www.vine.com/
19. Flickr: http://www.flickr.com
20. Linkedin: http://www.linkedin.com
21. Ceballos, M.R., Gorricho, J.L.: P2p file sharing analysis for a better performance. In: Proc. of the 28th International Conference on Software Engineering, ACM (2006)
22. Liu, B., Cui, Y., Lu, Y., Xue, Y.: Locality-awareness in bittorrent-like P2P applications. IEEE Transactions on Multimedia **3**(11) (2009)
23. Li, J.: Peer-to-Peer multimedia applications. In: Proc. of the 14th annual ACM international conference on Multimedia. (2006)
24. Liu, J., Rao, S.G., Li, B., Zhang, H.: Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast. In: Proc. of IEEE, Special Issue on Recent Advances in Distributed Multimedia Communications. (2008)
25. Xiao, X., Shi, Y., Gao, Y.: On Optimal Scheduling for Layered Video Streaming in Heterogeneous Peer-to-Peer Networks. In: Proc. of the 16th annual ACM international conference on Multimedia. (2008)
26. Leonardi, E., Mellia, M., Meo, M., A. P. Couto da Silva: Bandwidth-Aware Scheduling Strategy for P2P-TV Systems. In: Proc. of 8th International Conference on Peer-to-Peer Computing. (2008)
27. Ciullo, D., Garcia, M.A., Horvath, A., Leonardi, E., Mellia, M., Rossi, D., Telek, M., Veglia, P.: Network awareness of P2P live streaming applications: a measurement study. IEEE Trans. on Multimedia (12) (2010)
28. Howe, J.: The rise of crowdsourcing. Wired magazine **14**(6) (2006) 1–4
29. Huberman, B., Romero, D., Wu, F.: Social networks that matter: Twitter under the microscope. Available at SSRN 1313405 (2008)
30. Gao, H., Barbier, G., Goolsby, R.: Harnessing the crowdsourcing power of social media for disaster relief. Intelligent Systems, IEEE **26**(3) (2011) 10–14

31. Zaidan, O., Callison-Burch, C.: Crowdsourcing translation: Professional quality from non-professionals. In: ACL. (2011) 1220–1229
32. Bilić, V., Holderbaum, A., Kimmes, A., Kornelius, J., Stoll, C., Trier, W.V.: Amateur subtitling-selected problems and solutions
33. Eskevich, M., Jones, G.J., Aly, R., Ordelman, R.J., Chen, S., Nadeem, D., Guinaudeau, C., Gravier, G., Sébillot, P., De Nies, T., et al.: Multimedia information seeking through search and hyperlinking. In: Proceedings of the 3rd ACM conference on International conference on multimedia retrieval, ACM (2013) 287–294
34. Keimel, C., Pangerl, C., Diepold, K.: Comparison of lossless video codecs for crowd-based quality assessment on tablets
35. Bremer-Laamanen, M.: User benefits and crowdsourcing-articles in the spotlight. (2013)
36. Vondrick, C., Patterson, D., Ramanan, D.: Efficiently scaling up crowdsourced video annotation. International Journal of Computer Vision **101**(1) (2013) 184–204
37. Mo, K., Zhong, E., Yang, Q.: Cross-task crowdsourcing. (2013)
38. Liu, W., Hauptmann, A.G.: A crowdsourcing approach to tracker fusion
39. Liu, Q., Peng, J., Ihler, A.: Variational inference for crowdsourcing. In: Advances in Neural Information Processing Systems. (2012) 701–709
40. Aydin, B.I., Yilmaz, Y.S., Bulut, M.F., Demirbas, M.: Crowdreply: A crowdsourced multiple choice question answering system
41. Lakshminarayanan, B., Teh, Y.W.: Inferring ground truth from multi-annotator ordinal data: a probabilistic approach. arXiv preprint arXiv:1305.0015 (2013)
42. Yi, J., Jin, R., Jain, S., Yang, T., Jain, A.K.: Semi-crowdsourced clustering: Generalizing crowd labeling by robust distance metric learning. In: Advances in Neural Information Processing Systems. (2012) 1781–1789
43. Khosla, A., Hamid, R., Lin, C.J., Sundaresan, N.: Large-scale video summarization using web-image priors
44. : NDN project website (2011) Accessed: 2013-07-08.
45. Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M.F., Briggs, N.H., Braynard, R.L.: Networking named content. In: ACM CoNEXT '09. (2009)
46. Zhang, L., Estrin, D., Burke, J., Jacobson, V., Thornot, J., Smatters, D., Zhang, B., Tsudik, G., Krioukov, D., Massey, D., Papadopulos, C., Abdelzaher, T., Wang, L., Crowley, P., E., Y.: Named data networking (NDN) project, PARC Technical Report TR-2010-02 (Oct. 2010)
47. Smetters, D.K., Jacobson, V.: Securing network content, PARC Tech. Rep. TR-2009-1 (Oct. 2009)
48. Lin, W.S., Zhao, H.V., Liu, K.R.: Incentive cooperation strategies for peer-to-peer live multimedia streaming social networks. Multimedia, IEEE Transactions on **11**(3) (2009) 396–412
49. Cheng, X., Liu, J.: Nettube: Exploring social networks for peer-to-peer short video sharing. In: INFOCOM 2009, IEEE, IEEE (2009) 1152–1160
50. Mislove, A., Gummadi, K.P., Druschel, P.: Exploiting social networks for internet search. In: 5th Workshop on Hot Topics in Networks (HotNets06). Citeseer. (2006) 79
51. Wang, X., Chen, M., Kwon, T., Yang, L., Leung, V.: Ames-cloud: A framework of adaptive mobile video streaming and efficient social video sharing in the clouds. (2013)
52. Wang, Z., Wu, C., Sun, L., Yang, S.: Peer-assisted social media streaming with social reciprocity. (2013)

53. Hoßfeld, T., Seufert, M., Hirth, M., Zinner, T., Tran-Gia, P., Schatz, R.: Quantification of youtube qoe via crowdsourcing. In: Multimedia (ISM), 2011 IEEE International Symposium on, IEEE (2011) 494–499
54. Gardlo, B., Ries, M., Hossfeld, T.: Impact of screening technique on crowdsourcing qoe assessments. In: Radioelektronika (RADIOELEKTRONIKA), 2012 22nd International Conference, IEEE (2012) 1–4
55. Wang, D., Abdelzaher, T., Kaplan, L., Aggarwal, C.: Recursive fact-finding: A streaming approach to truth estimation in crowdsourcing applications. Urbana **51** (2013) 61801
56. Hei, X., Liu, Y., Ross, K.W.: Iptv over p2p streaming networks: the mesh-pull approach. Communications Magazine, IEEE **46**(2) (2008) 86–92
57. Hei, X., Liang, C., Liang, J., Liu, Y., Ross, K.W.: A measurement study of a large-scale p2p iptv system. Multimedia, IEEE Transactions on **9**(8) (2007) 1672–1687
58. Liu, J., Rao, S.G., Li, B., Zhang, H.: Opportunities and challenges of peer-to-peer internet video broadcast. Proceedings of the IEEE **96**(1) (2008) 11–24
59. Venkataraman, V., Yoshida, K., Francis, P.: Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. In: Network Protocols, 2006. ICNP'06. Proceedings of the 2006 14th IEEE International Conference on, IEEE (2006) 2–11
60. Magharei, N., Rejaie, R.: Prime: Peer-to-peer receiver-driven mesh-based streaming. IEEE/ACM Transactions on Networking (TON) **17**(4) (2009) 1052–1065
61. Carofiglio, G., Gallo, M., Muscariello, L., Perino, D.: Modeling data transfer in content-centric networking. In: Int. Teletraffic Congress, (ITC). (2011)
62. Tortelli, M., Cianci, I., Grieco, L.A., Boggia, G., Camarda, P.: A fairness analysis of content centric networks. In: Proc. of Int. Conf. on Network of the Future, NOF, Paris, France (Nov. 2011)
63. Rossi, D., Rossini, G.: On sizing CCN content stores by exploiting topological information. In: IEEE INFOCOM, NOMEN Worshop. (2012)
64. Grieco, L.A., Saucez, D., Barakat, C.: AIMD and CCN: past and novel acronyms working together in the Future Internet. In: Capacity Sharing Workshop 2012 (CSWS'12) co-located with ACM SIGCOMM CoNEXT 2012. (Dec. 2012)
65. Tortelli, M., Grieco, L.A., Boggia, G.: CCN forwarding engine based on bloom filters. In: Proc. of ACM Int. Conf. on Future Internet Technologies, CFI, Seoul, Korea (Sep. 2012)
66. You, W., Mathieu, B., Truong, P., Peltier, J., Simon, G.: Dipit: A distributed bloom-filter based pit table for ccn nodes. In: Computer Communications and Networks (ICCCN), 2012 21st International Conference on. (30 2012-aug. 2 2012) 1 –7
67. Grieco, L.A.: Emerging topics: special issue on multimedia services in information centric networks (guest editorial). IEEE COMSOC MMTC E-letter (2013) 4–5
68. Jacobson, V., Smetters, D.K., Briggs, N.H., Plass, M.F., Stewart, P., Thornton, J.D., Braynard, R.L.: Voccn: voice-over content-centric networks. In: ACM ReArch '09. (2009)
69. Zhu, Z., Wang, S., Yang, X., Jacobson, V., Zhang, L.: ACT: audio conference tool over named data networking. In: Proceedings of the ACM SIGCOMM workshop on Information-centric networking, New York, NY, USA, ACM (2011) 68–73
70. Detti, A., Pomposini, M., Blefari-Melazzi, N., Salsano, S., Bragagnini, A.: Offloading cellular networks with information-centric networking: The case of video streaming. In: Proc. of IEEE Int. Conf. on a World of Wireless, Mobile and Multimedia Networks, WoWMoM. (2012) 1–3

71. Li, Z., Simon, G.: Time-shifted TV in content centric networks:the case for cooperative in-network caching. In: Proc. of IEEE ICC. (Jun. 2011)
72. Xu, H., Chen, Z., Chen, R., Cao, J.: Live streaming with content centric networking. In: Proc. 3rd Int. Conf. on Networking and Distributed Computing, Hangzhou, China, 2012. (2012)
73. Liu, Y., Geurts, J., Point, J.C., Lederer, S., Rainer, B., Mueller, C., Timmerer, C., Hellwagner, H.: Dynamic Adaptive Streaming over CCN: A Caching and Overhead Analysis. In: Proc. of IEEE Int. Conf. on Communication, ICC, Budapest, Hungary, June (Jun. 2013)
74. Lederer, S., Mller, C., Rainer, B., Timmerer, C., , Hellwagner, H.: Adaptive streaming over content centric networks in mobile networks using multiple links. In: Proc. of IEEE Int. Conf. on Communication, ICC, Budapest, Hungary, June (Jun. 2013)
75. Han, B., Wang, X., andTaekyoung Kwon, N.C., Choi, Y.: AMVS-NDN: Adaptive Mobile Video Streaming and Sharing in Wireless Named Data Networking. In: In Proc. of IEEE Int. Workshop on Emerging Design Choices in Name-Oriented Networking, NOMEN, Torino, Italy (2013)
76. Kulinsky, D., Burke, J., Zhang, L.: Video streaming over named data networking. IEEE COMSOC MMTC E-letter (2013) 6–9
77. Park, J., Kim, J., Jang, M.W., Lee, B.J.: 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) 512–513
78. Li, H., Li, Y., Lin, T., Zhao, Z., Tang, H., Zhang, X.: MERTS: A more efficient real-time traffic support scheme for Content Centric Networking. In: Proc. in IEEE Int. Conf. on Computer Sciences and Convergence Information Technology, ICCIT. (2011) 528–533
79. Ciancaglini, V., Piro, G., Loti, R., Grieco, L.A., Liguori, L.: CCN-TV: a data-centric approach to real-time video services. In: in Proc. of IEEE International Conference on Advanced Information Networking and Applications, AINA, Barcelona, Spain (Mar. 2013)
80. Han, L., Kang, S., Kim, H., In, H.: Adaptive retransmission scheme for video streaming over content-centric wireless networks. IEEE Com. Let. **PP**(99) (2013) 1–4
81. Goodchild, M.F., Glennon, J.A.: Crowdsourcing geographic information for disaster response: a research frontier. International Journal of Digital Earth **3**(3) (2010) 231–241
82. Pallis, G., Vakali, A.: Insight and perspectives for content delivery networks. Commun. ACM (2006) 101–106
83. Vakali, A., Pallis, G.: Content delivery networks: status and trends. IEEE Internet Computing **7**(6) (2003) 68–74
84. Melazzi, N.B., Chiariglione, L.: The Potential of Information Centric Networking in Two Illustrative Use Scenarios: Mobile Video Delivery and Network Management in Disaster Situations. IEEE COMSOC MMTC E-letter (2013) 17–20
85. Piro, G., Ciancaglini, V.: Enabling real-time TV services in CCN networks. IEEE COMSOC MMTC E-letter (2013) 17–20
86. : Omnet++ home page
87. Wiegand, T., Sullivan, G., Bjontegaard, G., Luthra, A.: Overview of the H.264/AVC video coding standard. IEEE Trans. on Circuits and Systems for Video Technology **13**(7) (Jul. 2003) 560 –576
88. Chiocchetti, R., Rossi, D., Rossini, G., Carofiglio, G., Perino, D.: Exploit the known or explore the unknown?: hamlet-like doubts in icn. In: Proceedings of the

second edition of the ICN workshop on Information-centric networking. ICN '12, New York, NY, USA, ACM (2012) 7–12
89. Rossi, D., Rossini, G.: Caching performance of content centric networks under multi-path routing (and more). In: Technical report, Telecom ParisTech. (2011)
90. Piro, G., Grieco, L., Boggia, G., Fortuna, R., Camarda, P.: Two-level Downlink Scheduling for Real-Time Multimedia Services in LTE Networks. In: IEEE Trans. Multimedia. Volume 13. (Oct. 2011) 1052 – 1065