Mobile Peer-To-Peer Video Streaming over Information-Centric Networks

Andrea Detti, Bruno Ricci, Nicola Blefari-Melazzi
CNIT – University of Rome "Tor Vergata" – Department of Electronic Engineering
Via del Politecnico 1 – 00133 Rome (Italy)
email: {andrea.detti, bruno.ricci, blefari}@uniroma2.it

Abstract - Information Centric Networking (ICN) is a network paradigm alternative to the classic host-centric communication model: it provides users with content exposed as names, instead of providing communication channels between hosts. In this paper, we present a peer-to-peer application for live streaming of video content encoded at multiple bit rates. The application enables a small set of neighbouring cellular/Wi-Fi devices to increase the quality of video playback by using the Wi-Fi network to share the portion of the live stream downloaded by each peer via the cellular network. The application exploits the main functionalities of ICN: routing by name, in network caching and multicast delivery. Our work includes the implementation of a Java prototype of the application on a test-bed composed by Linux machines running the CCNx tool and streaming MPEG-DASH videos. We measured the performance of our solution and verified on the field that ICN simplifies the development of applications, as it provides built-in functions, which would be much more difficult to implement by relying on classical TCP/IP tools only.

1 Introduction

The classical Internet model relies on the IP *host-centric* paradigm, in which the network layer is used to transfer bits among hosts. This model is a very good fit for a set of current Internet applications, such as conversational (VoIP) or remote control (SSH) services, in which two specific hosts need to exchange data. However, most Internet applications nowadays use the network as a repository of contents, identified by names. And when these contents are requested by a very large number of users, their delivery on top of a host-centric IP network has required to introduce many incompatible, proprietary, content-oriented functionality, like name-based routing, caching,

multicasting, data replication. For instance, Content Delivery Networks heavily exploit such functionality.

The research community is proposing a new network paradigm, called Information Centric Networking (ICN), with the aim of harmonizing, simplifying and making more efficient the handling of content within the network [1]. ICN proposes an evolution of Internet core functionality to inherently support content-oriented services in any kind of network: wide and local area networks, mobile ad-hoc or mesh networks. ICN rethinks network services and distributes information (or contents) identified by names rather than setting up bit pipes between hosts identified by addresses. When a user expresses an interest for a content to the ICN Application Programming Interface (API), the underlying ICN functionality takes care of routing-by-name the content request towards the "closest" copy of the content with such a name (e.g. original or replica server or an in-network cache), and of delivering the content back to the requesting user. Different ICN architectures have been proposed so far [2]; however, most past works (and this paper) take as reference the Content Centric Network (CCN) architecture [3], which is also supported by a real implementation for Linux, MAC OS and Android devices, named CCNx [4]. A conceptually similar proposal to CCN is NDN, which is actively working to achieve analogous aims and has its roots in CCN [5].

Video streaming is one of the applications that motivated ICN and is expected to be one of the major sources of traffic for both fixed and mobile networks [6]. The video streaming community is rapidly adopting pull-based, adaptive schemes (e.g. MPEG-DASH [7]), which perfectly fit the ICN service model. Indeed, the HTTP GET primitive used to pull video segments can be easily replaced by a similar ICN GET primitive. Pull-based streaming schemes are used both for client-server and peer-to-peer streaming (PPS) applications [8].

In this paper, we present an ICN peer-to-peer application for live streaming of videos encoded at multiple bit rates (adaptive live video streaming). Peers are assumed to be a *small* set of

neighbouring mobile cellular devices that cooperatively download a live video stream from the cellular interface and share downloaded video segments through a proximity channel (e.g. Wi-Fi Direct). The cooperation logic is designed to improve the playback quality perceived by a peer, with respect to the quality that the same peer could achieve by downloading the stream only by itself. The application exploits the CCN architecture [3]; as for video it uses the MPEG-DASH (Dynamic Adaptive Streaming over HTTP) streaming standard [7]. Our source code is freely available [24].

Although our solution presents some improvements with respect to existing applications, our primary goal is not to propose a better performing application, but to show how to exploit an ICN API, namely the CCN API, to simplify the application development. Indeed, if we had used the plain TCP/IP API, we would have had the burden of implementing and orchestrating on top of it routing-by-name, caching and multicast functionalities, which instead are built-in in CCN. In addition, it is worth noting that while there are many papers dealing with core ICN challenges, e.g. caching, routing scalability, transport mechanisms, security, etc.[9], only few of them are concerned with practical experiences on application design [11][12][13][16].

Our solution and code [24] have been tested both in an emulated environment and in a real cellular environment with mobile phones served by HSDPA networks of different operators.

2 Related works

2.1 CCN overview

CCN addresses contents by using unique hierarchical names that follow a URI syntax, e.g. ccnx:/foo.eu/video1. Long contents are split into chunks, uniquely addressed by names that contain the content name and the chunk number, e.g. ccnx:/foo.eu/video1/#x for the *x*th chunk of content ccnx:/foo.eu/video1 (in what follows we will omit the scheme identifier ccnx:/). To fetch a chunk, a receiver sends out an Interest message which includes the chunk name; then the network sends back

the data within a Data message. Interest and Data messages are sent and received through any network interface available on a node; these interfaces, in the CCN framework, are called *faces*.

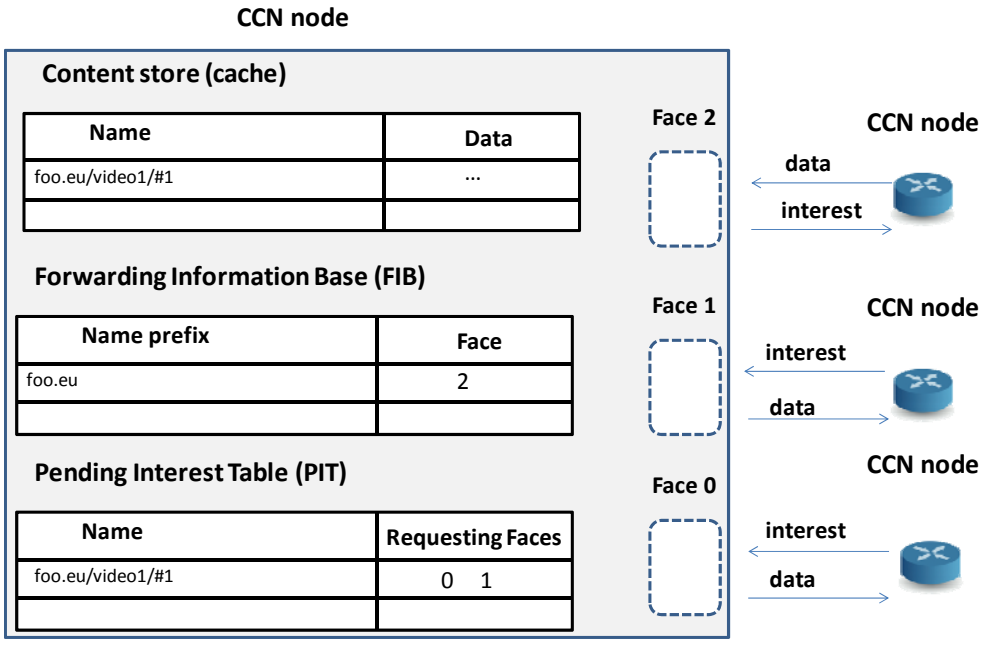

Fig. 1 – CCN node

Fig. 1 reports the main elements of a CCN node, namely the Forwarding Information Base (FIB), the Pending Interest Table (PIT) and the Content Store (aka the content cache). A CCN node uses a name-based FIB to route-by-name Interest messages using a prefix match logic. A FIB entry contains a name prefix (e.g. foo.com) and the identifier of the upstream faces on which the Interest message can be forwarded towards available sources (e.g. face 2 in case of Fig. 1). In case an Interest message matches more than one FIB entry, a forwarding strategy selects one or a set of them on which to relay the Interest. Similarly to IP forwarding, we assume that the CCN forwarding strategy selects the face that provides the *longest prefix match*.

While the FIB is used to forward Interest messages, the PIT is used to forward back Data messages. During the Interest forwarding process, a CCN node leaves reverse path information <chunk name, downstream face list> in the PIT, where the downstream face list contains the list of faces from which the node received the same Interest. For instance, in case of Fig. 1 the node has received two

Interests for the content ccnx:/foo.eu/video1/#x from faces 0 and 1. Only the first received Interest is forwarded; the following Interests are not forwarded but their downstream face identifiers are added to the downstream list of the PIT. When an Interest reaches a node having the requested chunk, the node sends back the chunk within a Data message. A node can have the chunk either because it is temporarily available in its Content Store or because there is a local repository application connected through a local face that permanently stores the chunks of a given content. The Data message is routed on the downstream path by consuming the information previously left in the PITs. The Data message is relayed hop-by-hop on all the downstream faces, so inherently providing in-network multicast distribution. Traversed CCN nodes temporarily cache forwarded Data messages in their Content Store so inherently providing in-network caching functionality.

To download a content, a receiver fetches all the related chunks by sending out a sequence of Interest messages. For flow control purposes, the receiver uses a receiver-driven approach [10] which consists in limiting the number of in-flight Interests through a congestion window (cwnd). The congestion window size may be constant or e.g. regulated by an Additive Increase Multiplicative Decrease (AIMD) congestion control mechanism.

CCNx [4] is a Linux-based implementation of CCN, whose faces are UDP or TCP tunnels. The software is mainly composed of: *ccnd* that implements the node functionality of Fig. 1 by using C code; a set of applications/libraries of which the most used are *ccnr,* which is a permanent repository of contents, *ccngetfile* and *ccnputfile* used to pull a content from or to push a content in a repository respectively, and *ccndc*, which controls the CCN FIB.

2.2 MPEG DASH

MPEG-DASH (Dynamic Adaptive Streaming over HTTP) [7] is the first standard for adaptive video streaming. In MPEG-DASH a video is divided in segments. Each segment is available at different bit rates, uniquely identified by a URL, and usually contained in a file with a M4S extension. A related file, named Media Presentation Descriptor (MPD), contains meta-information:

coding scheme, duration of segments, their playtime, resolutions and URLs. Video segments and related manifest files can be offered by an HTTP server.

A client wishing to see a video, first downloads the MPD file and then pulls segments with a given bit rate using HTTP GETs. Since each segment has an independent URL, plain HTTP proxies can be used to provide in-network caching functionality. The rate selection strategy and the coding scheme are not defined by the standard and can be chosen as a function of the specific application environment. The download strategy is also left to the application developer. For instance, VLC uses a playout buffer storing a number of DASH segments and downloads a new segment whenever a segment is drained from the buffer by the decoding process, so keeping reserve segments in the buffer.

Streaming MPEG DASH videos over CCNx is rather straightforward. Each segment can be stored in a CCNx repository (i.e. ccnr) with the name of the related segment URL. The ccngetfile library can be used to fetch each segment. The video client can be a modified version of an off-the-shelf client (e.g. in [31] authors propose to use an open-source DASH-over-CCN VLC Plugin) or an unmodified version of the client bound with an HTTP-to-CCN proxy that converts HTTP GETs to ccngetfile instances; the latter approach is the one used in this paper.

2.3 Peer to Peer Video Streaming

Peer to Peer video Streaming (PPS) is a popular approach to distribute live media over Internet. The proposed architectures can be roughly classified in two classes [8]: Push/Tree based and Pull/Mesh based.

The Push/Tree based solution creates an overlay network among peers that has a tree shape; then the source pushes video data on such a tree. The Pull/Mesh based solution is inspired by the BitTorrent file sharing mechanism. A Tracker collects information about the state of the set of participating peers (aka swarm). A peer forms a mesh overlay network with a subset of peers, and exchanges video data with them. A peer announces which data items has available and requests

missing data items announced by (or discovered from) connected peers. In case of live streaming, the involved data set regards only a recent window of data items published by the source.

Pull/Mesh based PPS solutions are the best candidate for the ICN deployment, since most ICN approaches provide a pull-based API. In addition, Pull/Mesh based PPS are more robust than Push/Tree, and the Peer to Peer Streaming Protocol (PPSP) working group [26] is also proposing a Pull/Mesh based solution.

Besides the delivering strategy, the use of multi-rate encoding schemes has also a high impact on the quality of experience of PPS solutions. Multi-rate encoding schemes can be roughly classified as with or without dependent substreams. The use of a multi-rate encoding approach with dependent substreams, like Multiple Description Coding (MDC) or H.264 Scalable Video Coding (SVC), clearly improves the peer to peer (P2P) sharing. Data fetched by peers reproducing a video at low quality can be re-used also by peers that are reproducing the video at higher quality. Conversely in the case of a multi-rate encoding approach with independent substreams, e.g. a set of independent H.264 AVC streams coded at different rates, only peers that are reproducing the video at the same rate can share data, which obviously reduces the chances of sharing content. However, practically, MDC and H264 SVC encoding schemes are more complex than AVC; as of today, they do not avail themselves of efficient and cheap hardware decoding and there is little (experimental) software support for PC/Laptop. Instead, AVC decoding is available off-the-shelf for any device, including mobile phones and tablets.

2.4 Advances with respect to the state of art

In this paper we design a Pull/Mesh based PPS application for cellular devices that are physically close to each other. The application has four distinguishing features: it is based on *CCN*; it is devised for *live-streaming*; it handles videos using the MPEG-DASH streaming format, encoded at *multi-rates* with independent streams (H.264 AVC); the P2P collaboration strategy aims at *maximizing the video playback quality* by concurrently exploiting the cellular downlinks of peers.

This work is an evolution and a completion of three previous conference papers [15] [27] [28] of ours. Specifically, in [15] we proposed a CCN *on-demand single-rate* PPS application for mobile devices, whose goal is to *offload the cellular interface*; the considered streaming format was Apple Live Streaming. In [27] (and in its demo [28]) we proposed an early version of the application proposed in this paper that suffered of a long startup delay; for instance, in case of 3 peers and with 2 seconds long MPEG DASH video segments, the startup delay was in the order of 20 seconds. In this paper we modify the cooperation strategy introducing the concept of *parts* and significantly reduce the startup delay at the cost of a limited bandwidth overhead. Limiting startup delay is of great importance. In [29] the authors report some measurements carried out on a wide data set provided by the Akamai client-side media analytics plug in; these measurements show that "viewers start to abandon the video if the startup delay exceeds about 2 seconds. Beyond that point, a 1-second increase in delay results in roughly a 5.8% increase in abandonment rate". In this paper we not only repeat the set of experiment performed in [27] by using the new P2P strategy but also introduce a new formulation to compute the tradeoff between startup delay and bandwidth overhead, increasing also the measurement set.

As regards other papers on PPS, we note that in [13] the authors propose a CCN adaptive video streaming application called AMVS-NDN, which enables a mobile device *either* to use its own 3G/4G connection *or* to connect via Wi-Fi to another mobile device to exploit its possibly better 3G/4G link. The cooperation strategy behind this solution resembles a selection of the best cellular gateway, and thus the achievable video coding rate is bounded by the capacity of the downstream link of this *single* gateway. In [16] the authors set up a test-bed for video streaming over CCN, named NDN Video: the scenario is rather different from ours, as they consider a *fixed* network and a *client–server* interaction model, i.e. without P2P cooperation. The naming scheme, instead, is similar to ours. In [17] the authors propose a TCP/IP application dealing with *on-demand single-rate* video streaming. In [18] the authors propose a BitTorrent approach for live streaming in a *fixed network*. The video is *single-rate* and the cooperation is aimed at *offloading the server*. Finally, we

observe that our application could in part resemble the case of "multi-homed" video streaming [20][30], since we propose to concurrently use more (cellular) links to fetch data. However, in a multi-homed scenario, different links are hosted *by the same device* whereas in our case each (cellular) link is bound with a different device. Clearly many other papers on PPS exist (see e.g. [21] and its references) and the research topic is well-known. However, our focus is on ICN/CCN and P2P adaptive video streaming exploitation.

3 The P2P Video Streaming Application

3.1 Scenario

As shown in Fig. 2, we consider a small set of neighbouring mobile cellular devices (from now on called mobile video peers, or simply peers) interested in streaming the same live video. For instance, we can imagine a situation in which passengers of a train are interested in watching news with their mobile phones, or alternatively a pay-per-view scenario in which all the mobile devices available in an apartment are concurrently used to improve the video quality of a stream offered by a content provider like Netflix.

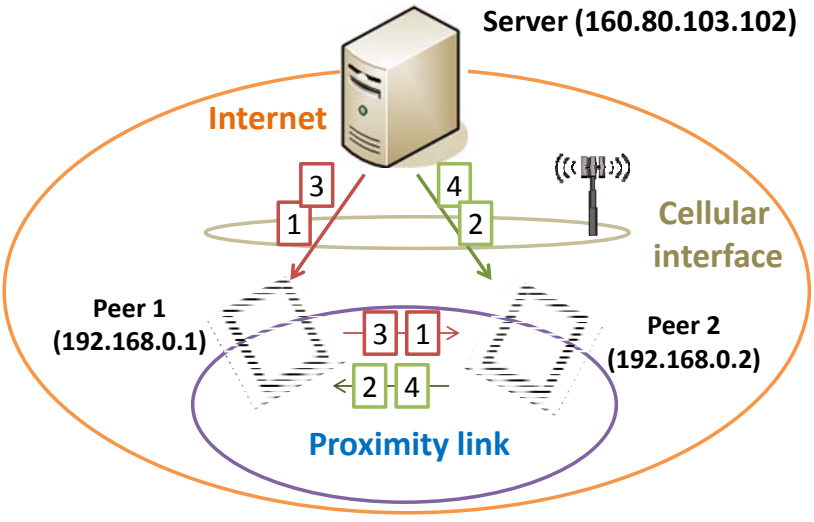

Fig. 2 - The application scenario

Each mobile device is connected to two different networks: a remote cellular network through the cellular interface (e.g., 3G), and a local *full mesh one hop* network, though a proximity wireless technology (e.g., Wi-Fi Direct).

Usually the transfer capacity of the *proximity* link is much greater than the single cellular link of each peer: thus, the remote link towards the cellular network operator is the bottleneck of the system. Besides, the group of mobile video peers is quite small (e.g. five peers), and thus the scalability of the application with respect to the number of peers of the group is not a central design issue.

3.2 Collaboration strategy at a glance

The application logic resembles that of a BitTorrent or Pull/Mesh approach in which the video server (i.e. a CCNx repository) is the seeder and the mobile video peers are the lechers. When two or more peers are interested in the same stream, each peer downloads from the server just a subset of video segments, sharing them with other peers through the proximity interface. For instance in Fig. 2, peer 1 pulls segments 1 and 3 from the server, and shares them with peer 2 through a proximity link. Peer 2 carries out the same operation for segments 2 and 4.

The tracker function is distributed. When another peer needs a video segment, it first checks the segment availability on the one-hop local mesh among peers; if the segment is found in a peer, it is downloaded from the proximity interface; otherwise it is downloaded from the server via the cellular interface. Being a live streaming, not all video segments are available from the beginning, and download operations are organized in periodical rounds, during which peers cooperate to download a window of latest-published segments (e.g. 3 segments). Moreover, peers communicate to each other their monitored cellular bandwidth and this information is used to select the bit rate of the video stream.

In what follows we show how we implemented this strategy with CCN means. Even though CCN functionality within any network node can further improve performance (e.g. due to in network caching), our PPS application strictly requires CCN functionality only on peers and server.

3.3 Video source, server and contents

The video source produces the MPD and M4S files, i.e. video segments coded at different bit rates. These contents are inserted in a video server, which is a plain CCN repository. The MPD file is available on the source since the beginning of the video stream distribution, while the segments are inserted in the repository as they are created during the live streaming.

Differently from [27] [28], we do not directly store MP4 files in the repository but, to reduce the initial playback delay, we fragment them in a number of *parts* and store the parts in the repository. It is not difficult to see that by choosing the fragmentation level (i.e. the number of parts per segment) it is possible to find a tradeoff between playback delay and application efficiency. The relationship among segments, parts and CCN chunks is sketched in Fig. 3

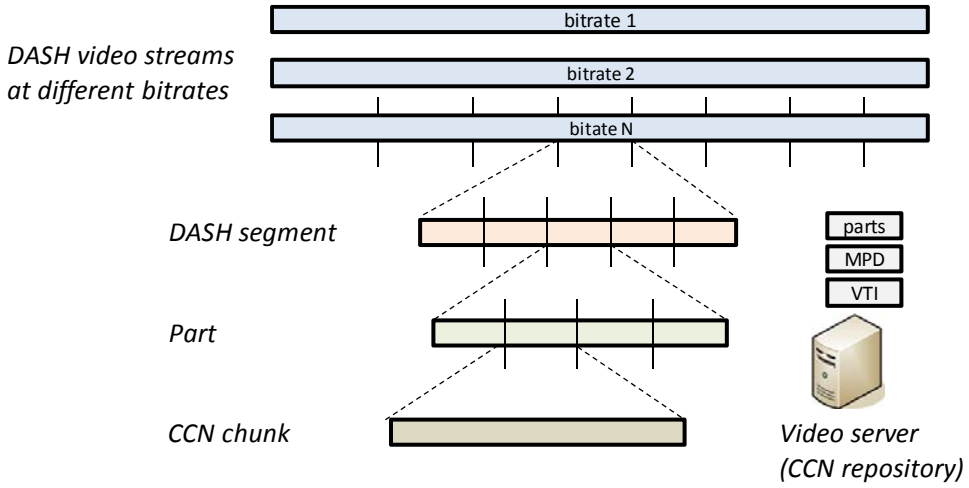

Fig. 3 – Segment, parts and chunks

Since a peer may join and leave the streaming anytime and the streaming is live, the peer needs to be mildly synchronized with the source, in order to pull only the last-produced video segments. We achieve this mild synchronization by publishing at the source side a so-called Video Timing

Information (VTI), which contains the sequence number of the last-produced video segment, its publishing time and the current time of the live stream.

3.4 Naming scheme

We chose a hierarchical name for all the contents. The names used for MPD, VTI, M4S, PRI and PSI information are reported in Tab. 1. PRI and PSI are signalling information that will be described later on.

Content	Name
MPD	ccnx://server-prefix/filename.mpd
VTI	ccnx://server-prefix/filename.vti
M4S	ccnx://server-prefix/filename/SN=X/PN=Y/BW=Z.m4s
PRI	ccnx://prd/server-prefix/filename/SN=X/PN=Y/BW=Z.m4s
PSI	ccnx://prd/server-prefix/filename/PS/IP

Tab. 1 - Naming scheme

The server-prefix is a legal DNS name identifying the video server. The filename identifies the specific live stream. The parameter *X* is the video segment number, *Y* is the part number and *Z* is the bit rate of the segment. As an example, part 4 of segment 145 coded at 100 bps of the video stream video1 provided by the foo.eu server is identified by the name ccnx://foo.eu/video1/SN=145/PN=4/BW=100.m4s.

3.5 Video peer operation

3.5.1 Peer join

To join the video stream, the peer downloads the VTI file and gets synchronized with the video source, i.e. it is aware of the latest segment number published by the source, and of the source clock. Even if this synchronization is clearly not very precise, it is sufficient for our purposes.

3.5.2 Collaboration strategy

After joining, the peer fetches the MPD file and begins to cooperate with other peers to *pre-fetch* and play video segments. As shown in Fig. 4, a pre-fetcher module uses the CCN layer to concurrently download video parts from both the proximity and cellular interface. A peer first tries to download a missing part from others peers; if it is not available the peer will use the cellular interface. During the download of a part the peer can redistribute the downloaded chunks on the proximity interface to requesting peers in a multicast fashion. After the download, the part is also cached in the CCN content store, in order to be shared with other peers requesting it on the proximity interface. Thus, both multicasting and in-network caching capabilities of CCN are exploited.

The search of missing parts on other peers is carried out by a *proximity route discovery* procedure discussed in section 3.5.4 below, and the selection of the interface for the download of a part is enforced by a *dynamic management of the CCN FIB* discussed in the section 3.5.3 below.

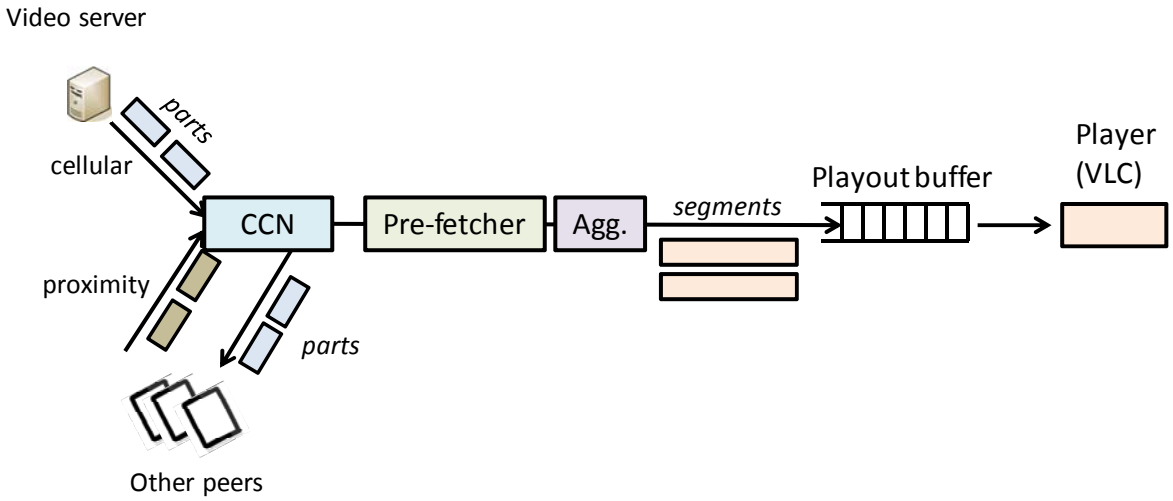

Fig. 4 – Pre-fetch and play

The pre-fetcher carries out download operations in batches of *A* parts, where a batch is called *pre-fetch window*. A pre-fetch window is composed by an integer number *P* of segments, i.e. *F=A/P* is the number of parts per segment. When the source has fully published a pre-fetch window, a *pre-*

fetch round starts and peers collaborate to download the pre-fetch window as fast as possible [18]. A pre-fetch round lasts for $P\ T_s$ seconds, i.e. the time needed by the source to produce the P video segments of the next window, being T_s is the duration of a video segment (e.g. 2 sec). At the end of a round, peers estimate their cumulative downlink cellular bandwidth and compute the highest possible video coding rate that they can request during the next round. This *rate selection* procedure is described in section 3.5.5 below.

As shown in Fig. 4, all downloaded parts are sent to an aggregation function that reassembles DASH segments and sends them to a playout buffer. The buffer is drained by a DASH video player (e.g. VLC) that starts the playback when the buffer contains $2P$ segments. Thus, the playout delay is $2\ P\ T_s$.

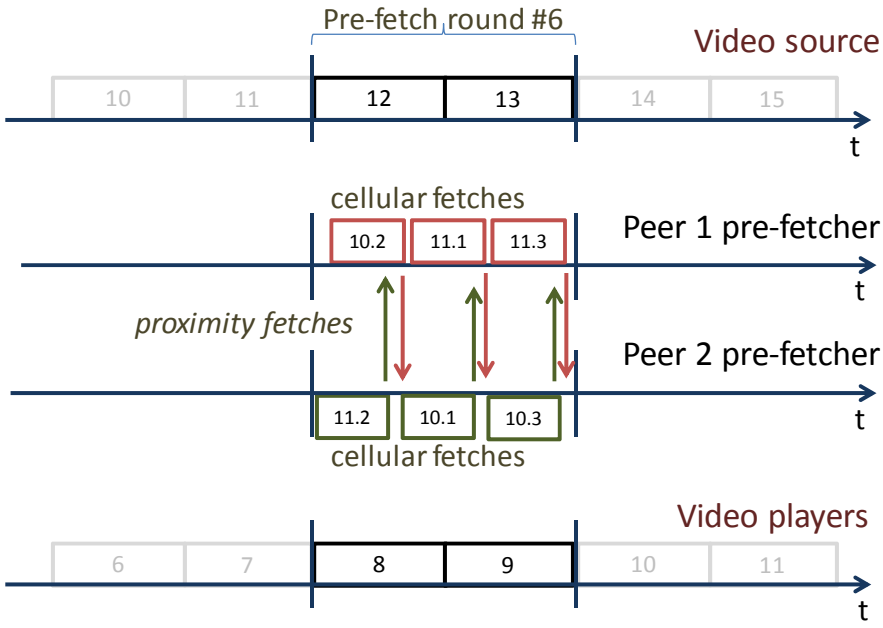

Fig. 5 - Time evolution during the 6th pre-fetch round

In Fig. 5 we report the time evolution during the 6th pre-fetch round of: the video source, the pre-fetchers (with A=6 and P=2) and the players of two collaborating peers. During this period, the source publishes segments 12 and 13. Players play segments 8 and 9. Pre-fetchers collaboratively download the parts of segments 10 and 11 from the cellular interface, sharing these parts on the

proximity interface. We assumed that each segment is divided in three parts, thus e.g. segment 10 is formed by 10.1, 10.2 and 10.3 parts. Moreover, to avoid the occurrence of duplicated cellular fetches (i.e. two or more peers downloading the same part from the cellular interface) peers randomly shuffle the sequence of parts to download during the round.

3.5.3 Management of CCN FIB

CCN selects the forwarding face of an Interest message using its FIB. Thus, by properly controlling the FIB entries with a routing strategy, it is possible to enforce the interface to be used on a per-content basis.

To download a part from the cellular interface, a peer inserts in its CCN FIB a *cellular-route*, pointing to the video server public IP addresses (e.g. discovered through DNS). As an example, Fig. 6 reports the CCN FIB of two video peers, 1 and 2. For peer 1, we can see that there is a cellular-route for part 3 of segment 11 pointing to the video server at the public address 160.80.103.102:9695, via the cellular interface rmnet0. For peer 2, a similar cellular-route is inserted for part 1 of segment 10.

If, instead, the desired part is found on the proximity network, a *proximity-route* is inserted in the CCN FIBs pointing to the neighbour peer. With reference to Fig. 6, we can see that there are two proximity-routes: for peer 1, the route is for part 1 of segment 10 and points to the IP address of peer 2 (192.168.0.2) via the proximity interface wlan0; for peer 2, the route is for part 3 of segment 11 pointing to the IP address of peer 1 (192.168.0.1) via the proximity interface wlan0.

Once the download has been completed, the cellular-route or the proximity-route is removed from the FIB, in order to limit the size of the information base.

FIB of video peer 1

Name prefix	Output face
ccnx:/prd	internal
ccnx:/prd	224.0.0.1:9695 (wlan0)
ccnx:/foo.eu/video1/SN=11/PN=3/BW=100.m4s	160.80.103.202:9695 (rmnet0)
ccnx:/foo.eu/video1/SN=10/PN=1/BW=100.m4s	192.168.0.2:9695 (wlan0)
…	

FIB of video peer 2

Name prefix	output-face
ccnx:/prd	internal
ccnx:/prd	224.0.0.1:9605 (wlan0)
ccnx:/foo.eu/video1/SN=10/PN=1/BW=100.m4s	160.80.103.202:9695 (rmnet0)
ccnx:/foo.eu/video1/SN=11/PN=3/BW=100.m4s	192.168.0.1:9695 (wlan0)
…	

Fig. 6 - Example of CCN FIBs of two video peers

3.5.4 Proximity route discovery

The *proximity route discovery* allows a peer to discover the availability of parts on neighbour peers. It exploits the CCN Interest-Data interaction as follows. When a peer starts downloading a given part from the cellular interface, it also publishes a "signalling" content called *Proximity-Route-Info* (PRI), whose name is equal to the name of the downloading part with an added "prd/" control prefix (see Tab. 1). The data contained in the PRI is merely the IP address and CCNx port of the downloading peer. The PRI contents are stored in a fast repository inside the PPS application. As an example, the PRI published with the part ccnx:/foo.eu/video1/SN=1/PN=4/BW=100.m4s is ccnx:/prd/foo.eu/video1/SN=1/PN=4/BW=100.m4s.

The PRI acts as a routing announcement that must be solicited. Peers continuously query for PRIs of missing parts with periodic Interest messages. These Interests are routed by the FIB on a preconfigured multicast address. As shown in Fig. 6, the FIB has an entry for the prd prefix towards a multicast address bound with the proximity interface wlan0 and also has an entry used for incoming prd Interest, which points to a local face connected to the internal PRI repository.

Once a peer retrieves a PRI, it inserts the proximity-route in the FIB and then immediately starts to download the interested video part by requesting it to the CCN layer.

It is noteworthy that when a peer starts to fetch a part from another peer, the latter peer may still be downloading that part from its cellular interface. In this case, this peer will become the splitting point of a multicast tree, because as soon as it receives CCN chunks from the cellular interface, they will be relayed both to the local pre-fetcher and to all other peers that have established a proximity route for the part. In case of late discovery, a same (but delayed) distribution result is obtained thanks to the CCN content store of the video peer.

3.5.5 Video coding rate selection algorithm

At the beginning of a new pre-fetch round, each peer computes the bit rate of the video parts that are going to be downloaded in the round. The selected bit rate is the highest possible one that avoids emptying the playout buffer, i.e. video freezes. This evaluation is made considering: i) the available video coding rates BW_h, where h is a bit rate index; and ii) the *net* rate C_i that each peer may obtain on the cellular interface, i.e. the maximum download rate seen above the CCN layer.

We observe that the straightforward approach of selecting the first available bit rate below the cumulative net cellular bandwidth $C_{tot} = \Sigma\, C_i$ could not be effective. Indeed, a peer can download an integer number of parts, but not fractions of them, and all peers must download all parts of a pre-fetch window within the round period, in order to avoid starving the playout buffer. We refer to these constraints as *quantization constraints* and the bit rate index h^* of the video parts downloaded during the round #k+1 is derived by solving the following constrained maximization problem:

$$J_{i,h} = \text{floor}\left[\frac{A\, C_i(k)}{BW_h}\right] \qquad (1)$$

$$h^* = \max_h \left\{ \text{s.t.} \sum_{i=1}^{\min(A,M)} J_{i,h} \geq A \right\} \qquad (2)$$

The parameter $J_{i,h}$ in eq. (1) represents the integer number of video parts that a peer can download by using the cellular interface during a pre-fetch round, assuming that parts are coded with a constant bit rate $BW_{h.}$ The parameter $C_i(k)$ is the net cellular capacity estimated by peer i at the end of round k. The parameter A is the number of parts forming the pre-fetch window. The maximization of eq. (2) yields the index h^* of the highest video coding rate such that it is possible to download all parts of the pre-fetch window within the round duration. The parameter M is the number of peers. In what follows, we use the symbol BW to indicate the selected video coding rate BW_{h*}.

We note that the quantization constraints may prevent to exploit all the cumulative net cellular capacity C_{tot}. For instance, the sum in eq. (2) is limited to $min(A,M)$, since at most A peers can be exploited to download A parts from the cellular interface; thus the cellular capacity of the remaining peers is not used. In addition, even with $A>M$, the solution of eq. (2) may prevent some slow peers to download parts from their cellular interface; indeed, only peers that have $J_{i,h}>0$ will carry out remote downloads, while peers with $J_{i,h}=0$ will not, because these peers will be unable to download even a single part within the pre-fetch round duration. We analyse the inefficiency deriving from the quantization constraints in the next section.

To solve eqs. (1) and (2), a peer should know the set of available coding rates BW_h and all the net cellular capacities $C_i(k)$ for $1 \leq i \leq M$. If a video has L possible coding rates, a peer discovers these rates BW_h ($1 \leq h \leq L$) from the MPD file fetched during the join operation. The shared knowledge of $C_i(k)$ requires the ith peer to compute its own $C_i(k)$ and to distribute it to other peers. To compute $C_i(k)$, a peer monitors the download rate above CCN during round #k and either directly uses the observed value or computes $C_i(k)$ by injecting the observed value in a smoothing average algorithm (which we did in our implementation). The computed value is distributed by the peer as a named content, called *pre-fetch status information (PSI)*. The naming scheme used for the PSI is reported in Tab. 1, where the parameter IP is the IP address of the providing peer. Using the information

contained in the PRIs, at the end of a round, a peer has a list of the IP addresses of peers that participated to the cellular download and can pull the PSI of these peers through traditional CCN Interest-Data interaction. In this way, all peers will have the same set of $C_i(k)$ and compute the same value of h^*.

4 Dimensioning

The PPS application has two main configuration parameters, namely the number of parts A forming the pre-fetch window and the number of parts F per segment. These parameters should be carefully dimensioned, both to efficiently use the cellular radio resources and to limit the initial playout delay.

4.1 Efficiency and playout delay

We measure the efficiency E of the PPS application as the ratio between the video coding rate BW and the overall gross cellular capacity R_{tot} provided by downlinks of the peers.

$$E = BW/R_{tot} \quad = V\,T \qquad (3)$$

The efficiency E is lower than one for two reasons:

i) below the CCN API, the CCN/UDP/IP stack obviously introduces control overheads. Consequently, the overall net cellular capacity C_{tot} available at the CCN API is lower than R_{tot} of a factor V, named *control efficiency*; i.e. $C_{tot} = V\,R_{tot}$. In appendix I we trivially show that the control efficiency V has an inverse proportionality with the number of parts per segment F. In facts, the greater is F, the smaller is the byte length of a part and the higher is the control overhead. Moreover, for a fixed value of F, the control efficiency increases at the increasing of the overall gross cellular capacity R_{tot} since it is possible to select higher video coding rates; therefore the video parts have a greater byte length and a lower control overhead.

ii) upon the CCN API, the quantization constraints of the rate selection algorithm (see eqs. 1,2) may prevent the complete use of the overall net cellular capacity C_{tot}. Therefore, even in the ideal case of a video coding providing any possible video coding rate, the achievable video coding rate BW obtained through the cooperation could be lower than C_{tot} of a factor T, named *rate selection efficiency*; i.e. $BW = T\ C_{tot}$. In appendix II we analyse the behaviour of T versus A, in case of M of peers with net cellular rates C_i that follows a uniform or Gaussian distribution. We find out that T strongly depends on the ratio A/M and to achieve a higher efficiency a greater value of A/M is required. Moreover, T mildly depends on the heterogeneity of the downlink rates of the peers and increasing the heterogeneity slightly worsen the rate selection efficiency.

As discussed in section 3.5.2, the playout delay D is equal to the duration of $2P$ segments, where P is the integer number of *segments* of the pre-fetch window. Thus the playout delay can be written as:

$$D = 2\ P\ T_s = 2\ \frac{A}{F} T_s \quad (4)$$

where the ratio $P=A/F$ is the integer number of segment forming the pre-fetch window.

4.2 Dimensioning with delay constraint

We observe that playout delay and efficiency are contrasting performances. Indeed, by increasing F or by decreasing A, the delay decreases but the efficiency decreases as well. In what follow we discuss a dimensioning approach that gives priority to the delay. It consists in searching the couple (A,F) which assures a given playout delay D and, secondarily, a good efficiency E. Clearly, other dimensioning approaches are possible, e.g. by giving priority to the efficiency rather than to the playout delay.

Eq. 4 imposes that, to obtain a given delay D, the length of pre-fetch window A should be equal to:

$$A = \frac{D\,F}{2\,T_s} \quad (5)$$

Consequently, only the parameter F can be changed, to find a good value of the efficiency E. We analyse the impact of F on E under the constraint of eq. 5 by using a Matlab simulator. We assume to have M peers. The jth peer has a gross cellular capacity equal to R_j constant over the time and equal to $R+\delta_j R$, where R is a constant rate (e.g. 1 Mbit/s) independent of j and δ_j is a sample of a random variable. For each value of F, the simulator performs 2000 trials. At the end of the simulation, the final value of E is computed as the mean of the 2000 E values of the single trials. The size of the 95% confidence interval is below 1% of the mean value.

We carry out simulations in case of δ following uniform and Gaussian distributions with zero mean. In the uniform case we consider two possible ranges of the distribution, namely ± 0.2 and ± 0.8. In Gaussian case we consider two possible values of the standard deviation σ, namely $2\sigma = 0.2$ and $2\sigma = 0.8$. We only report the case of Uniform distribution in the interval ± 0.8, since the derived conclusions are valid also in the other cases.

Fig. 7, Fig. 8 and Fig. 9 report the efficiency $E = T\,V$ in case of $D = 4T_s$ for an average gross cellular capacity R per peer equal to 200 kbps, 500 kbps, 1Mbps respectively. We consider the case of 3, 5 and 8 peers. We also consider a configuration with a single peer, since also in this extreme case the PPS application should provide valuable performance. Fig. 10, Fig. 11 and Fig. 12 report the same results when $D = 2T_s$.

In case of a single peer, the rate selection efficiency T is obviously equal to one. Therefore, the increase of F only implies a penalty in the control efficiency V and thus a decrease of the efficiency $E = T\,V$. In case of more peers, the rate selection T efficiency is lower than one but improves increasing F (i.e. A from eq. 5). Fig. 7, Fig. 8 and Fig. 9 show that the gain in the rate selection efficiency T obtained increasing F overcomes the control efficiency loss, and the efficiency tends to increase at the increase of F.

We also note that for a given value of F, having peers with a higher gross cellular capacity R improves the efficiency E since higher video coding rate (BW) will be used, which improves the control efficiency.

Overall, we observe that a value of F between 15 and 20 provides efficiency closes to the maximum value in most of the considered scenarios. Further increasing F can excessively penalize the case of a single peer.

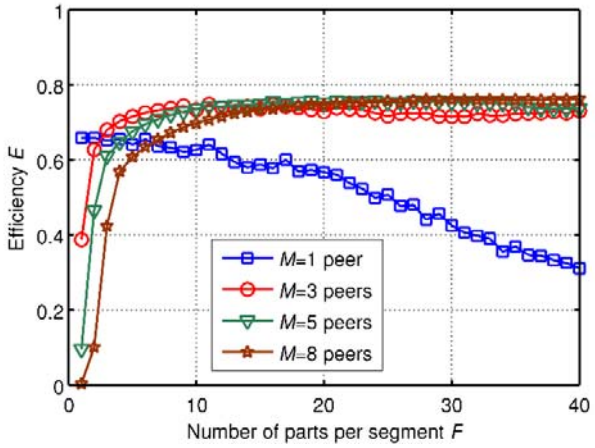

Fig. 7 – Efficiency E versus number of parts per segment F, δ uniform distribution ± 0.8, average gross cellular capacity per peer R = 200 kbps, playout delay D = 4 T_s

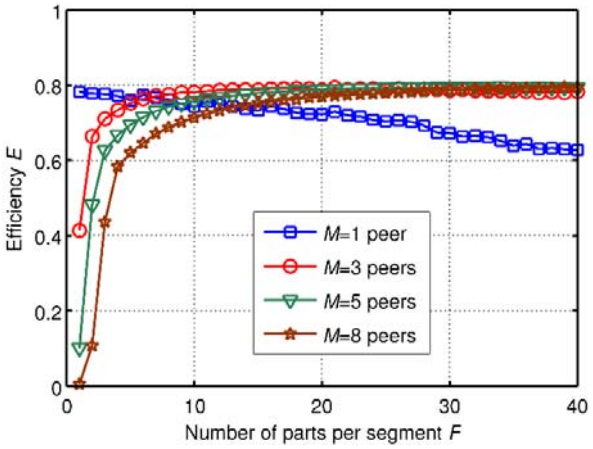

Fig. 8 - Efficiency E versus number of parts per segment F, δ uniform distribution ± 0.8, average gross cellular capacity per peer R = 500 kbps, playout delay D = 4 T_s

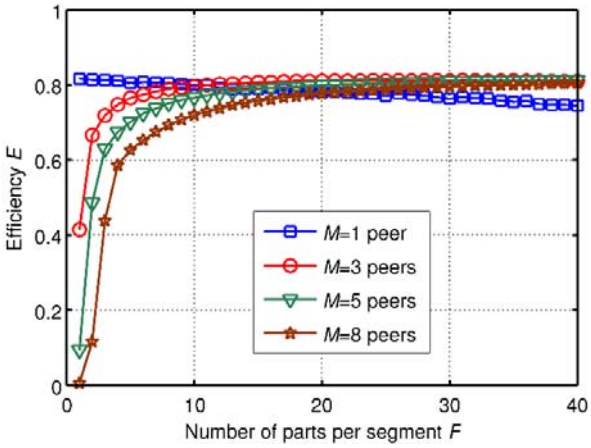

Fig. 9 – Efficiency E versus number of parts per segment F, δ uniform distribution ± 0.8, average gross cellular capacity per peer R = 1 Mbps, playout delay D = 4 T_s

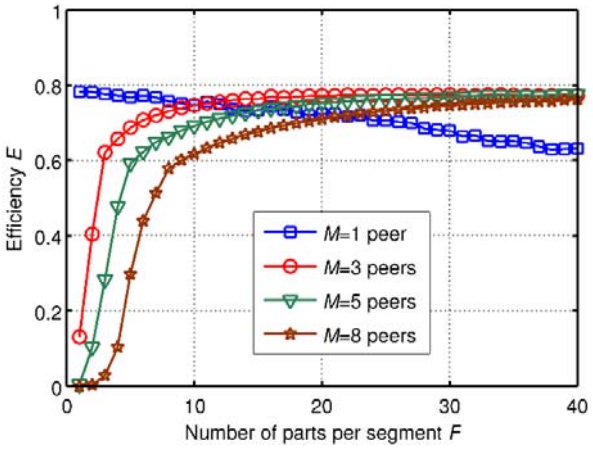

Fig. 10 – Efficiency E versus number of parts per segment F, δ uniform distribution ± 0.8, average gross cellular capacity per peer R = 200 kbps, playout delay D = 2 T_s

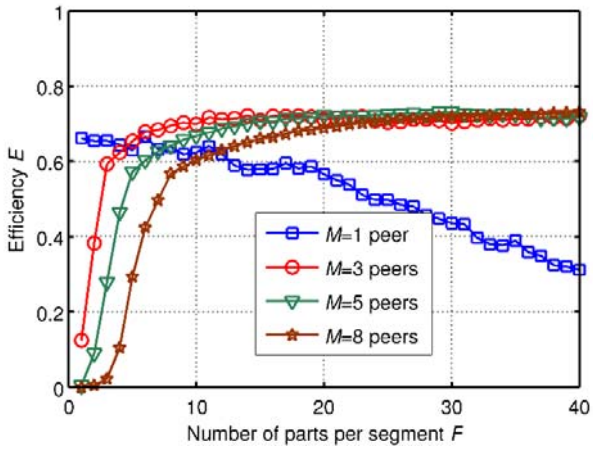

Fig. 11 – Efficiency *E* versus number of parts per segment *F*, δ uniform distribution ± 0.8, average gross cellular capacity per peer *R* = 500 kbps, playout delay *D* = 2 T_s

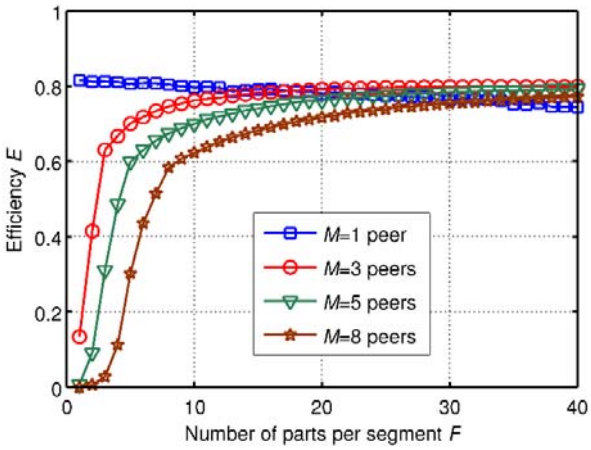

Fig. 12 – Efficiency *E* versus number of parts per segment *F*, δ uniform distribution ± 0.8, average gross cellular capacity per peer *R* = 1 Mbps, playout delay *D* = 2 T_s

5 Experimental assessment

5.1 The prototype

We implemented a Linux-based prototype of the PPS application using Java and plain CCNx 0.8.1. We used VLC 2.1.0 as MPEG-DASH video client; the interaction between client and application is made in a proxy-style, using a local HTTP connection. Fig. 13 shows the main software components. VLC is connected to a local HTTP proxy module that fetches video segments from the playout buffer of the PPS application. The buffer is filled by the pre-fetch and aggregation operations, which are also supported by PRI discovery and video coding rate selection. Pre-fetch, rate selection and discovery use CCNx to publish and download information; the discovery functionality also changes the configuration of the CCNx FIB. It is noteworthy that also the VLC client has a video coding rate selection algorithm, whose results may be different from the bit rate selected by the ICN PPS application. However, we found out that VLC is insensitive of the actual coding rate of segments returned to an HTTP GET, and only verifies that the HTTP answer contains the same URL of the requested segments.

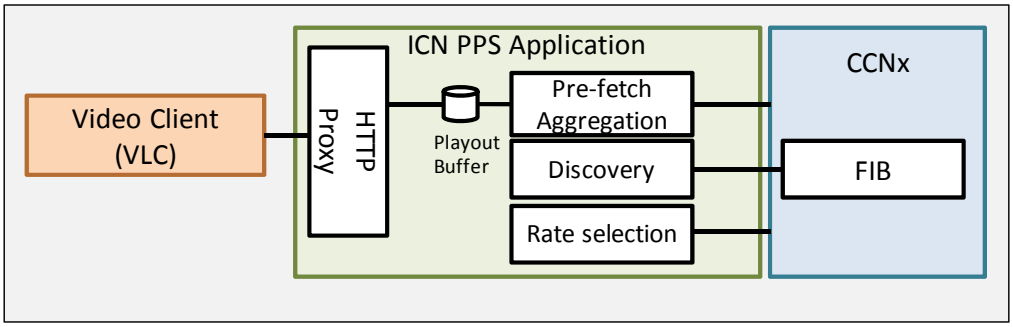

Fig. 13 – Software components

5.2 Test-bed setup

We verified the effectiveness of our application in the test-bed configurations reported in Fig. 14 and Fig. 15. Peers are Linux laptops connected to each other via a *Wi-Fi ad-hoc* full mesh at 54 Mbps, which represents the proximity interface. The video server is a fixed Linux PC on the public Internet. All the devices are located in our University laboratory. Peers are connected to the video server either through an *emulated* cellular connection (Fig. 14) realized with an Ethernet link with a rate controlled by the Linux TC tool, or through a *real HSDPA* cellular connection (Fig. 15), offered by a USB-tethered Android mobile phone.

We used the MPEG-DASH version of the movie "Big Buck Bunny" [32]. The resolution of the video is 480p, with 270 segments, each of them lasting $T_s = 2$ sec; the available video coding rates are fourteen, ranging from 100 kbps up to 4.5 Mbps. We uploaded the movie on a plain CCNx repository (i.e. the video server) splitting up each segment into 16 parts. We used a pre-fetch window length A of 48 parts, i.e. P=3 segments.

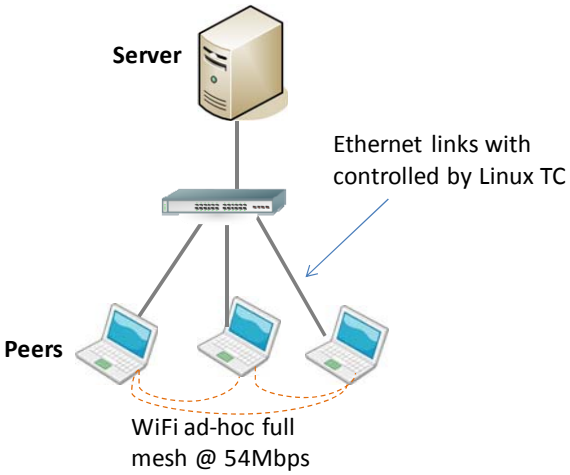

Fig. 14 – Test-bed setup with emulated cellular connections

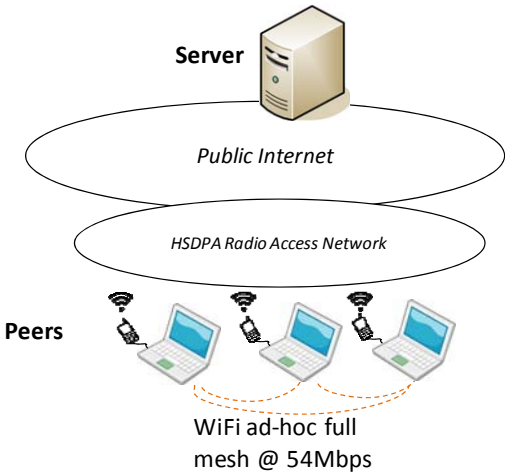

Fig. 15 – Test-bed setup with real HSDPA cellular connections

5.3 Tests with emulated cellular connections

Fig. 16 reports the cumulative net cellular capacity C_{tot} and the video coding rate BW versus time in case of 2, 4, 8 and 10 peers. The gross cellular capacities R_i of peers are homogeneous and configured with Linux TC tool at 500 kbps. The ticks in the Y axis of the plot indicate actual available video coding rates (i.e. 200, 350, 500, 700, ...).

This plot confirms the effectiveness of the PPS application in improving the video quality by exploiting the cellular capacity of peers. Indeed, the higher the number of peers, the higher the selected video coding rate BW. We note that the net cellular capacity C_i per peer is roughly equal to 380 kbps, i.e. 76% of the gross cellular capacity R_i =500 kbps enforced with the Linux TC[1]. We observe that the selected video coding rate (BW) may not be the one immediately below the cumulative net bandwidth C_{tot}. For instance, in case of 10 peers the video coding rate immediately below C_{tot} is 3400 kbps, but the rate selection algorithm chooses BW = 2800 kbps. This is due to the floor operation of eq.1, which lowers the rate selection efficiency T. In other tests, we have removed

[1] According Fig. 22 of Appendix I, we expect a control efficiency V of 82% rather than 76%. The additional 6% comes from an underestimation of the net cellular capacity due to the Java processing delay, which is not taken into account in eq. 8

the floor operator of eq. 1 and the rate selection algorithm has chosen the video coding rate closer to C_{tot}, but the video streaming has suffered of some playback freezes, which were not observed when using the floor operator.

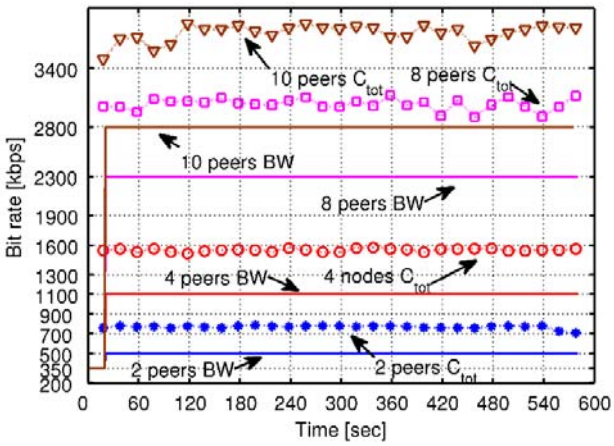

Fig. 16 – Cumulative net cellular capacity C_{tot} and video coding rate *BW* during video playback in case of 2,4,8,10 peers with gross cellular capacity per peer *R* = 500 kbps

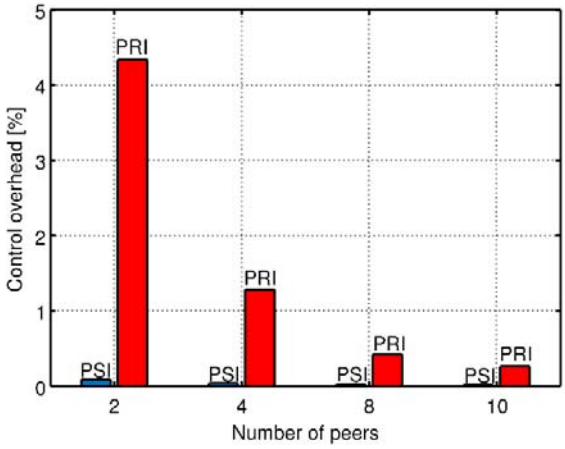

Fig. 17 – Percentage of PSI/PRI control overhead during video playback in case of 2,4,8,10 peers with gross cellular capacity per peer *R* = 500 kbps

Peer [kbps]			
N. Peers	*PSI*	*PRI*	*Video*
2	0.241	11.8	272
4	0.352	10.32	774
8	0.546	8.4	1987
10	0.661	8.112	3085

Tab. 2 – Average bitrate transmitted by one peer during video playback in case of 2,4,8,10 peers with gross cellular capacity per peer *R* = 500 kbps

Tab. 2 reports the average bit rate transmitted by a peer, which includes both CCN Interest and Data messages sent by the peer. We measured the amount of bit rate related to the video traffic and to the CCN-oriented signalling, which is due to the exchange of PSI and PRI signalling messages. The VTI (Video Timing Information) signalling message exchange occurs only at the peer joining, thus its traffic impact is negligible. The bit rate related to the PSI signalling linearly increases with the number of peers. Indeed, at the end of each pre-fetch round a peer sends a PSI message to all other

peers through a unicast CCN Interest-Data interaction. The bit rate related to the PRI signalling decreases with the number of peers. Indeed, at the start of a pre-fetch round, each peer sends a number of multicast PRI Interests equal to the number of video parts of the round. During the pre-fetch round, a peer replies with a multicast PRI Data message if the peer has fetched the related video part from the server. Increasing the number of peers implies that each peer fetches fewer video parts from the server and so the peer generates fewer PRI Data messages per round, while the number of PRI Interest messages per round remains constant.

Fig. 17 reports the PSI (PRI) control overhead, measured as the ratio between the PSI (PRI) and the video bit rate of a peer (see Tab. 2). We observe that both PRI and PSI control overheads are rather limited, up to 4.25% and they decrease when the number of peers increase.

Fig. 18 shows the dynamic behaviour of the PPS application when peers join and leave, in case of five peers with gross cellular capacities per peer R = 500 kbps. As expected, when a peer joins or leaves, the video coding rate BW promptly increases or decreases, respectively.

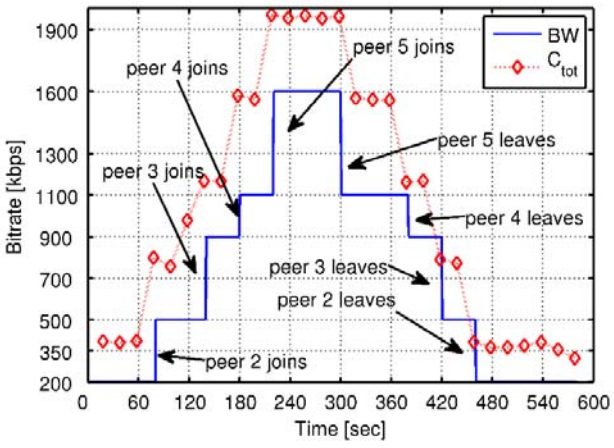

Fig. 18 – Cumulative net cellular capacity C_{tot} and video coding rate BW during video playback in case of 5 peers with gross cellular capacity per peer R = 500 kbps

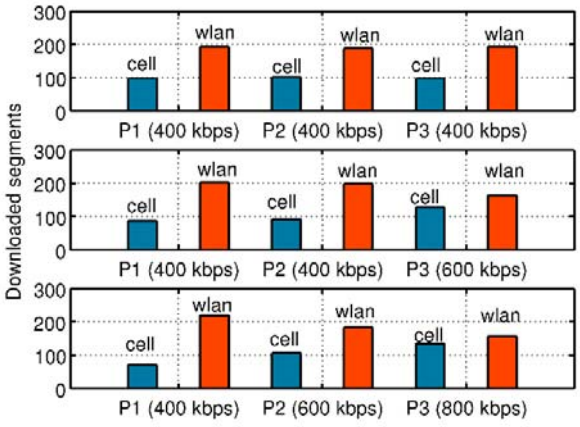

Fig. 19 – Number of segments downloaded from cellular and Wi-Fi interface in case of 3 peers (P1,P2,P3) with different gross cellular capacity per peer R, reported in parentheses

Fig. 19 reports the number of segments downloaded from the cellular interface and from the proximity link (Wi-Fi) in case of three peers (P1, P2 and P3) with three different configurations of their gross cellular capacity. The figure shows that the consumption of cellular capacity is greater for peers with higher capacity. Indeed, the application tends to exploit all the available cellular capacity to improve video quality. For instance, in the lower plot, the third peer, P3, has a gross cellular capacity equal to 800 kbps and its cellular capacity consumption is about twice the one of P1, which has a gross cellular capacity equal to 400 kbps.

5.4 Tests with real HSDPA connections

Fig. 20 reports the video coding rate (*BW*) in case of five collaborating peers connected to the video server with real HSDPA connections (Fig. 15), provided by the Telecom Italia Mobile operator. The figure reports also a sampling of the cumulative net cellular capacity C_{tot} measured by the peers. The streaming starts with just one video peer; the other four peers join one by one and then leave in the opposite order.

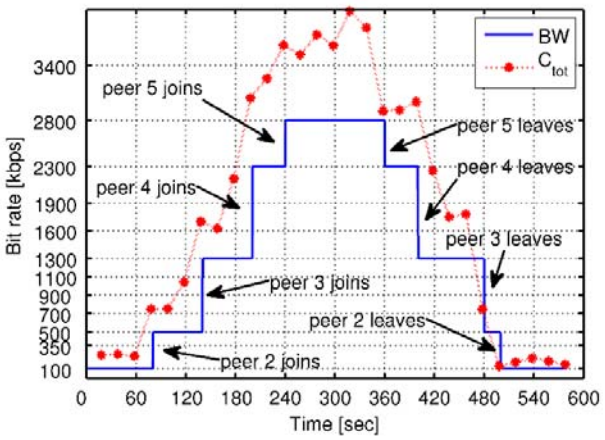

Fig. 20 - Cumulative net cellular capacity C_{tot} and video coding rate *BW* during video playback in case of 5 peers with real HSDPA connections

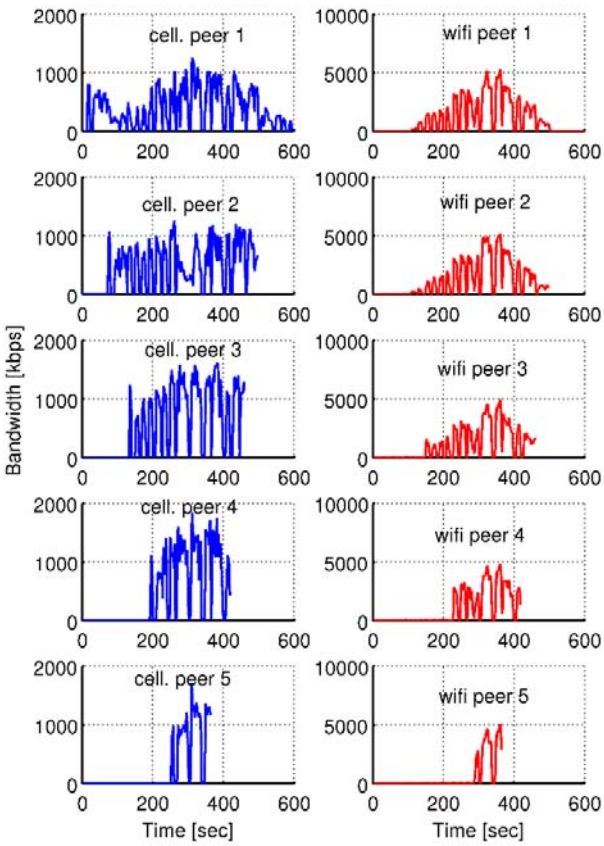

Fig. 21 - Cellular (HSDPA) and proximity (Wi-Fi) received gross traffic

Measurements are gathered from peer 1 (a Samsung Galaxy S II), which is the video peer present for the whole duration of the test. We can observe that, along with the insertions of video peer 2 (a Samsung Galaxy S3 Mini), of video peer 3 (a Google Nexus 5), of video peer four (an HTC One) and of video peer five (a Motorola Moto G) at seconds 70, 130, 190 and 240, respectively, both the cumulative net cellular capacity C_{tot} and the video coding rate *BW* follow the same behaviour. The same happens when video peers 5, 4, 3 and 2 leave the streaming, at seconds 360, 410, 480 and 500, respectively. Fig. 21 reports the measurement of gross traffic (including CCN/UDP/IP protocol overhead) received on both the HSDPA and Wi-Fi interfaces by all five video peers. All video peers exploit the cellular interfaces almost continuously. The "pulsing" behaviour is due to the round structure. The selected video coding rate is lower than the net cellular capacity (Fig. 20), thus the

download of the segments of the round finishes a little before the round end and the use of HSDPA bandwidth drops to zero until the start of the next round.

6 Conclusions

We presented an ICN-enabled peer-to-peer application for the adaptive live streaming of videos encoded at multiple bit rates to a small set of neighbouring mobile cellular devices. We used the CCN architecture, in combination with the MPEG-DASH (Dynamic Adaptive Streaming over HTTP) streaming standard. We showed how video peers can cooperatively download a video stream and share it on a proximity channel, thus improving the video stream quality. We implemented a prototype of the application using the CCNx tool and Java. The open-source code and more detailed explanation on how to reproduce out test can be found in [24].

Acknowledgements

The work for this paper was performed in the context of the FP7/NICT EU- JAPAN GreenICN project, http://www.greenicn.org.

References

[1] G. Carofiglio, G. Morabito, L. Muscariello, I. Solis, M. Varvello, From content delivery today to information centric networking, Computer Networks, Volume 57, Issue 16, 13 November 2013, Pages 3116-3127

[2] G. Xylomenos, C.N. Ververidis, V.A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K.V. Katsaros; G.C. Polyzos, "A Survey of Information-Centric Networking Research," Communications Surveys & Tutorials, IEEE , vol.16, no.2, pp.1024,1049, Second Quarter 2014

[3] Van Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, R.L. Braynard, "Networking Named Content," in Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, Rome, Italy, 2009.

[4] CCNx website http://www.ccnx.org

[5] L. Zhang, A. Afanasyev, J. Burke, Claffy, L. Wang, V. Jacobson, P. Crowley, C. Papadopoulos, B. Zhang: “Named Data Networking”, ACM SIGCOMM Computer

Communication Review (CCR), July 2014

[6] "Cisco Visual Networking Index: Forecast and Methodology, 2012–2017,".

[7] Thomas Stockhammer, "Dynamic Adaptive Streaming over HTTP: Standards and Design Principles," in Proceedings of the Second Annual ACM Conference on Multimedia Systems, San Jose, CA, USA, 2011.

[8] N. Magharei, R. Rejaie, Yang Guo, "Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches," in IEEE INFOCOM, 2007.

[9] A. Ghodsi et al., "Information-centric networking: seeing the forest for the trees," in The 10th ACM Workshop on Hot Topics in Networks (HotNets-X), Cambridge, 2011.

[10] S. Salsano, A. Detti, M. Cancellieri, M. Pomposini, and Nicola Blefari-Melazzi, "Transport-layer issues in Information Centric Networks," in ACM SIGCOMM Workshop on Information-Centric Networking (ICN 2012), Helsinki, 2012.

[11] Z. Zhu, S. Wang, X. Yang, V. Jacobson, and L. Zhang, "ACT: Audio Conference Tool Over Named Data Networks," in ACM SIGCOMM Workshop on Information-Centric Networking (ICN 2011), Toronto, 2011.

[12] Jeff Burke, P. Gasti, N. Nathan, and Gene Tsudik, "Securing Instrumented Environments over Content-Centric Networking: the Case of Lighting Control and NDN," in The 2nd IEEE International Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN 2013), Turin, 2013.

[13] B. Han, N. Choi, T. Kwon, and Y. Choi, "AMVS-NDN: Adaptive Mobile Video Streaming and Sharing in Wireless Named Data Networking," in The 2nd IEEE International Workshop on Emerging Design Choices in Name-Oriented Networking (NOMEN 2013), Turin, 2013.

[14] R. Rejaie, Yang Guo N. Magharei, "Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches," in IEEE INFOCOM 2007, pp.1424-1432, 6-12 May 2007.

[15] Andrea Detti, Matteo Pomposini, Nicola Blefari-Melazzi, Stefano Salsano, and Andrea Bragagnini, "Offloading cellular networks with Information-Centric Networking: the case of video streaming," in The Thirteenth International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2013), San Francisco, 2012.

[16] Derek Kulinski and Jeff Burke, "NDN Video: Live and Prerecorded Streaming over NDN. NDN Technical Report NDN-0007," 2012.

[17] L. Keller et al., "Microcast: Cooperative Video Streaming on Smartphones," in The 10th ACM International Conference on Mobile Systems, Applications, and Services (ACM MobiSys 2012), Low Wood Bay, 2012.

[18] J. J. D. Mol, A. Bakker, J. A. Pouwelse, D. H. J. Epema, and H. J. Sips, "The Design and Deployment of a BitTorrent Live Video Streaming Solution," in IEEE International Symposium on Multimedia, San Diego, 2009.

[19] Z. Xiaoqing, P. Agrawal, J. P. Singh, Tansu Alpcan, and B. Girod, "Distributed Rate Allocation Policies for Multihomed Video Streaming Over Heterogeneous Access Networks," IEEE Transaction on Multimedia, vol. 11, no. 4, 2009.

[20] Bo Li and Hao Yin, "Peer-to-peer live video streaming on the internet: issues, existing approaches, and challenges," IEEE Communication Magazine, 2007.

[21] Teemu Koponen et al., "A Data-oriented (and Beyond) Network Architecture," in Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 2007), New York, NY, USA, 2007.

[22] Andrea Detti, Matteo Pomposini, Nicola Blefari-Melazzi, and Stefano Salsano, "Supporting the Web with an Information Centric Network that Routes by Name," Elsevier Computer Network, vol. 56, no. 17, pp. 3705-3722, 2012.

[23] Video trace available at http://www-itec.uni-klu.ac.at/ftp/datasets/mmsys12/BigBuckBunny/bunny_2s_480p_only/

[24] ICN P2P application software available at http://netgroup.uniroma2.it/Andrea_Detti/ICNvideo-live-DASH

[25] David R. Cheriton and Mark Gritter, "TRIAD:a scalable deployable NAT-based internet architecture," 2000.

[26] Peer-to-Peer Streaming Protocol (PPSP) working group web page https://datatracker.ietf.org/wg/ppsp/charter

[27] A.Detti, B. Ricci, N. Blefari-Melazzi,"Peer-To-Peer Live Adaptive Video Streaming for Information Centric Cellular Networks", IEEE PIMRC 2013,London, UK, 8-11 September 2013 (pdf)

[28] A.Detti, B. Ricci, N. Blefari-Melazzi, "Supporting mobile applications with Information Centric Networking: the case of P2P live adaptive video streaming", ACM SIGCOMM 2013, ICN workshop,Hong Kong, China, 12 August 2013 (pdf)

[29] Krishnan, S. Shunmuga, and Ramesh K. Sitaraman, "Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs." In proceedings of the 2012 ACM conference on Internet measurement conference. ACM, 2012.

[30] S. Lederer, C. Muller, B. Rainer, C. Timmerer, H. Hellwagner, "Adaptive streaming over content centric networks in mobile networks using multiple links", in Communications Workshops (ICC), 2013 IEEE International Conference on (pp. 677-681).

[31] S. Lederer, C. Muller, B. Rainer, C. Timmerer, and H. Hellwagner, "An Experimental Analysis of Dynamic Adaptive Streaming over HTTP in Content Centric Networks", in Proceedings of the IEEE International Conference on Multimedia and Expo 2013, San Jose, USA, July, 2013

[32] Stefan Lederer, Christopher Muller and Christian Timmerer, "Dynamic Adaptive Streaming over HTTP Dataset", in Proceedings of the ACM Multimedia Systems Conference 2012, Chapel Hill, North Carolina, February 22-24, 2012.

Appendix I: analysis of the control efficiency

The transport of a video segment over CCN implies an IP/UDP/CCN control overhead, and **fragmenting a video segment in *F* parts tends to increase the amount of overhead**. To analyse this effect, we consider a video with constant coding rate BW_h. In this case, the bit length L_h of each part can be written as

$$L_h = \frac{BW_h\, T_s}{F} \quad (6)$$

where T_s is the time duration of a segment. The number of CCN Data messages required to transport a part can be written as:

$$N_h = ceil\left[\frac{L_h}{D_s}\right] \quad (7)$$

where D_s is the payload size of the CCN Data message, e.g. 4096 bytes. Each Data packet has: a CCN control information, whose size CCN_h is in the order of 630 bytes (measured from CCNx traces and mostly due to security data); an UDP header of 8 bytes; a number of 20 byte IP headers equal to the number of involved IP packets that, for simplicity, we approximate equal to 4, independently of the Data packet payload length. Thus, the number of control bytes per part is equal to 718 N_h bytes and the control efficiency V can be written as:

$$V = \frac{L_h}{L_h + 718\, N_h} = \quad (8)$$

Fig. 22 reports the control efficiency V versus the number of segment per parts F, for different values of the video rate BW. The upper bound of the control efficiency, namely 0.8509, is achieved when a part can be transported by Data messages having maximum payload length (e.g. 4096), i.e. when the ceil operation in eq. 7 is not influent. Conversely, the higher the percentage of Data messages not completely full, the higher the control overhead and the lower the control efficiency.

For a given value of F, in case of a stream with high bit rate (e.g. BW = 3Mbps), a part is so long that the percentage of Data messages not completely full is rather limited; thus, the control efficiency is quite close to its upper bound. In case of low bit rates (e.g. BW = 200kbps), a part is composed by few bytes, thus the percentage of Data messages not completely full can be greater and the control efficiency decreases.

For a given video coding rate BW, increasing the number of parts per segment F, may initially increase or decrease the control efficiency V, depending on the impact of F in the ceil operation of eq. 7. However, by increasing F above $BW\ T_s / D_s$, the length of a part becomes lower than the Data message maximum payload length D_s and thus the control efficiency starts to monotonically decrease. For instance, in case of BW=200 kbps this occurs for F>12.

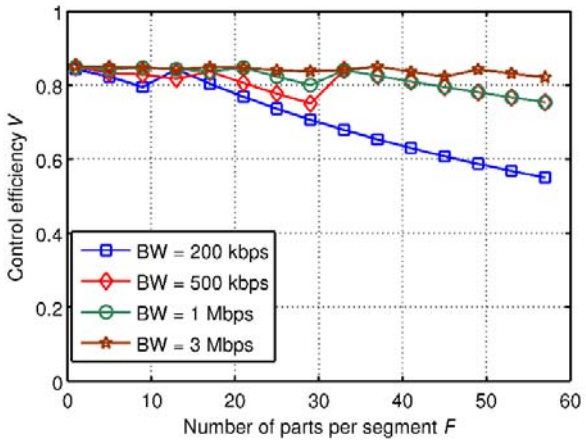

Fig. 22 – Control efficiency *V* versus number of segment per parts *F* for different video rate *BW*, T_s = 2s

Appendix II: analysis of the rate selection efficiency

In this section we evaluate the function $T(A)$ in case of M peers. The jth peer has a cellular net capacity C_j constant over time and equal to $C+\delta_j C$, where C is a constant rate (e.g. 2 Mbit/s) independent of j, while δ_j is a sample of a random variable. This configuration represents a scenario in which participating peers have values of downlink cellular capacity distributed around a central value C. For instance if δ follows a uniform distribution in the interval [−0.8, 0.8], the net cellular capacities of involved peers can vary up to the 80% with respect to a central value C.

To derive the function $T(A)$ we use a Matlab simulator that for each value of A: randomly generates the cellular net capacity C_j of peers; sets $BW_h = T\,C_{tot}$ in the eq. 1; and searches for the highest value of T satisfying the following condition:

$$\max_{T}\left\{\text{s.t. } \sum_{i=1}^{A} J_{i,h} \geq A\right\} \quad (9)$$

We observe that since $C_j = C+\delta_j C$, the random variable $J_{i,h}$ is equal to $A\,(1+\delta_i)/T\sum_j \delta_j$ and thus the maximization of eq. 9 is independent from the constant rate C.

For each value of A, the simulator performs 2000 trials. At the end of the simulation, the final value of T is computed as the mean of the 2000 T values of the single trials. The size of the 95% confidence interval resulted to be below the 1% of the mean value.

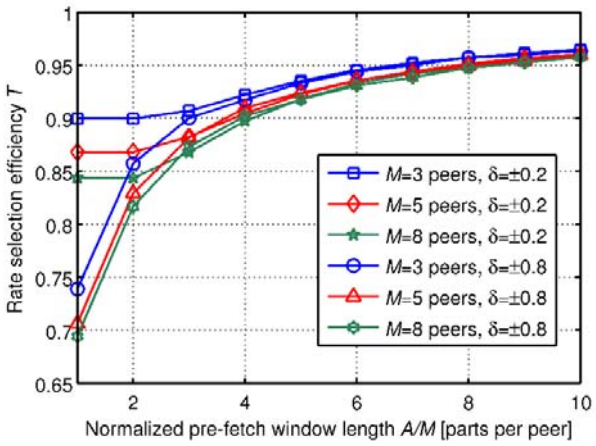

Fig. 23 – Rate selection efficiency *T* vs normalized pre-fetch window, δ uniform distribution ± 0.2 and ± 0.8

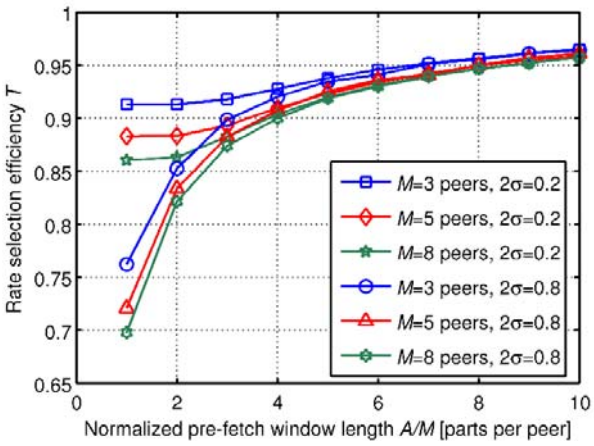

Fig. 24 – Rate selection efficiency *T* vs normalized pre-fetch window, δ Gaussian distribution with mean 0, 2σ=0.2 and 2σ=0.8

Fig. 23 reports the rate selection efficiency T versus the length of the pre-fetch window A normalized to the number of peers M, in case of δ with uniform distribution ranging in the interval [-0.2, 0.2] and in the interval [-0.8, 0.8]. We observe that to achieve a higher rate selection efficiency, a greater value of A/M is required. This means that the more we want to exploit the cellular resources, the longer the pre-fetch window has to be. We also point out that an increase of the number of peers M requires a linearly proportional increase of the pre-fetch window length, to achieve the same rate selection efficiency T. For instance, Fig. 23 shows that to achieve T close to 0.9 we need A/M equal to about 4, thus A=12 in case of 3 peers and A=24 in case of 8 peers. This implies that for a given value of A the rate selection efficiency decreases by increasing the number of peers M.

Fig. 23 also shows that for a fixed value of A, an increase in the variation range of random variable δ from 0.2 to 0.8 decreases the rate selection efficiency T; i.e. increasing the heterogeneity among peer cellular capacities worsen the rate selection efficiency T.

Fig. 24 show results in case of δ following a Gaussian distribution with zero mean and 2σ equal to 0.2 and 0.8, respectively. We do not observe significant differences with respect to the performance results obtained with a uniform distribution of δ.

Appendix III: definition of parameters

Parameter	Definition
A	Number of video parts of a pre-fetch window
BW_h	Available hth coding rate of a DASH video stream
BW	Video coding rate selected by the rate selection algorithm
C_i	Net cellular capacity of peer i, i.e. cellular bit rate available above the CCN API
C_{tot}	Cumulative net cellular capacity, i.e. sum of the net cellular capacities of the peers
CCN_h	Size of the CCN header of a Data message
D_s	Size of the payload of a Data message
E	Efficiency of the PPS application, i.e. ratio between the maximum achievable video coding rate BW and the overall gross cellular capacity R_{tot}
F	Number of parts per segment
L_h	Length of a part encoded at the BW_h coding rate
M	Number of mobile video peers
N_h	Number of CCN Data messages required to transport a video part encoded at the BW_h coding rate
P	Number of video segments of a the pre-fetch window
R_i	Gross cellular capacity of peer i, i.e. cellular bit rate available below the IP layer and used to transport the IP/UDP/CCN payloads and headers
R_{tot}	Cumulative gross cellular capacity, i.e. sum of the gross cellular capacities of the peers
T	Rate selection efficiency, i.e. ratio between the maximum achievable video coding rate BW and the overall net cellular capacity C_{tot}
T_s	Playback duration of a segment
V	Control efficiency, i.e. ratio between net and gross cumulative cellular capacities

Appendix IV: glossary

Term	Definition
CCN	Content Centric Network, an ICN architecture
CCNx	The software implementation of CCN
Cellular route	A CCN FIB entry pointing to the repository public IP address
Chunk	A part of a large content transported by a Data message
Content Store	A data cache of a CCN node
DASH	Dynamic Adaptive Streaming over HTTP
Data	CCN message used to transport a generic data item
Face	A logical/physical interface available to a CCN node for sending and receiving CCN messages
FIB	Forwarding Information Base, i.e. the CCN data structure used for routing-by-name Interest messages
ICN	Information Centric Networking
Interest	CCN message used to request a generic data item
MPD	Media Presentation Descriptor, i.e. the DASH manifest file that describes the video segments forming the video streams
Peer or video peer	A mobile CCN device interested in watching a given video and connected with other peer using a local one-hop full mesh wireless network
PIT	Pending Interest Table, i.e. the CCN data structure used to forward back the Data messages
Playout delay	Delay between the production of a video segment by the source and its playback by the peer
PPS	Peer-to-Peer video Streaming
Pre-fetch round	Download phase that lasts for a period equal to *P* video segments
Pre-fetch window	A sequence of *P* video segments

PRI	Proximity Route Information, i.e. a signaling content published by a peer to announce the availability of a video part
Proximity network	A local one-hop full mesh network to which all the peers are connected
Proximity route	A CCN FIB entry steering Interests to a peer in the proximity network
Proximity route discovery	Process ran by each peer to discover the availability of a part on neighboring peers
PSI	Pre-fetch Status Information, i.e. a signaling content published by a peer to announce its net cellular capacity
Segment	A portion of a DASH video, usually represented as a M4S file
Server	A fixed CCN device available on the public Internet which stores all the video parts in a CCN repository
Video source	A functionality producing the video parts stored in the server
Video part	A fragment of a video segment
VTI	Video Timing Information, i.e. a signaling information published by the video source to indicate the current production time of the stream