# Peer-To-Peer Live Adaptive Video Streaming for Information Centric Cellular Networks

Andrea Detti, Bruno Ricci, Nicola Blefari-Melazzi
CNIT – Department of Electronic Engineering – University of Rome "Tor Vergata" – Rome, Italy
e-mail: {andrea.detti, bruno.ricci, blefari}@uniroma2.it

***Abstract*— Information Centric Networking (ICN) is a new paradigm in which the network layer provides users with content exposed as names, instead of providing communication channels between hosts. We present a P2P application for the live streaming of video contents encoded at multiple bit-rates. The application enables a limited set of neighboring cellular devices to increase the quality of video playback, by cooperatively using their cellular (e.g. 3G) and proximity (e.g. Wi-Fi Direct) wireless connections. The application exploits key functionalities of ICN: routing by name, in network caching and multicast delivery. We developed a prototype of the application and assessed its performance in a test-bed based on Linux devices with 3G connections, the CCNx tool, the VLC player and the MPEG DASH streaming format. Our main contribution is to show how ICN can provide support to emerging video streaming applications, and ease their creation.**

***Index Terms*—Information Centric Networking; live adaptive video streaming; cellular networks; MPEG DASH.**

## I. INTRODUCTION

Information Centric Networking simplifies and optimizes the development and deployment of data dissemination services, by offering "out of the box" functionalities like in-network caching, content (or name)-based routing, multicasting, etc. Among the proposed ICN architectures (e.g. [1][2][3]), Content Centric Network (CCN) [2] is probably the best-known, thanks also to its CCNx [4] implementation. There are many literature works dealing with core ICN challenges, e.g. caching, routing scalability, transport mechanisms, security, etc. [5][6]. However, only few papers report practical experiences on application design [7][8][11].

In this paper we devise an ICN P2P application for live streaming of videos encoded at multiple bitrates (aka adaptive streaming). We use the CCN architecture and the MPEG DASH (i.e. Dynamic Adaptive Streaming over HTTP) streaming format [9]. Peers are a small set of neighboring mobile cellular devices, cooperatively using their cellular connections to improve the playback quality, with respect to the one they could achieve by downloading the stream independently.

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

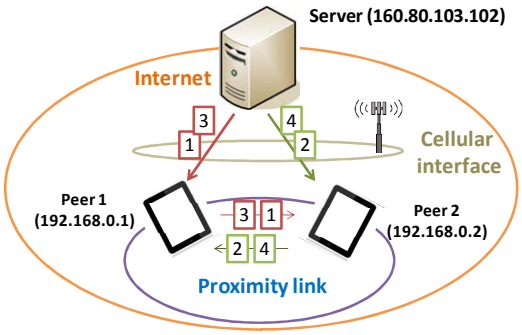


**Fig. 1 - Application scenario**

The application logic resembles the distributed BitTorrent approach (see Fig. 1): peers fetch segments through the cellular interface and share them via a proximity (one-hop) connection. We developed a CCNx based prototype of this application [13] and assessed its effectiveness in a test-bed formed by Linux laptops with real 3G connections.

Obviously the application concept is not novel and, albeit our solution has some differences with respect to the state-of-art (e.g. [10][11][12][14][15][17]), our primary goal is not to propose a better performing application, but to show "how to exploit" the application programming interface (API) of ICN to simplify the development process. Indeed, if we had used the plain TCP/IP API we would have had to implement and orchestrate routing-by-name, caching and multicast functionalities inside the application. In our case all this functionalities are performed by the underlying ICN, simplifying application development.

## II. CCN BACKGROUND

A CCN addresses contents (e.g. files) by using unique hierarchical names [2]. Long contents are split into *chunks*, uniquely addressed by names that contain the content name and a chunk identifier. To download a content, a client pulls the content chunks by issuing Interest messages. An interest message includes the chunk name and it is *routed-by-name* by CCN nodes, which (usually) use a longest prefix match strategy based on a name-based Forwarding Information Base (FIB). During the Interest forwarding process, CCN nodes leave reverse path information <chunk name – previous hops> in their Pending Information Table (PIT). When an Interest reaches a node having the requested

chunk, the node sends back the chunk within a Data message, which is routed on the reverse path by consuming the information previously left in the PITs. CCN nodes temporarily cache forwarded Data messages (aka *in-network* caching) and support *multicast* distribution. Indeed, in case of concurrent requests of the same chunk, a node forwards only one copy of an Interest message, stores in the PIT the set of previous-hop nodes requesting the chunk and, when receives a Data message, relays a copy of it towards each previous-hop node. A CCN implementation is CCNx [4], available for Linux, Mac and Android platforms. The links among CCNx nodes are UDP or TCP sockets, thus CCNx is actually an overlay deployment of CCN.

## III. The Video Streaming Application

### A. Scenario and assumptions

As shown in Fig. 1, we consider a group of neighboring users, that watch the same live video stream, offered by a server in the fixed Internet, with their mobile devices. Mobile devices are connected both to the Internet through a cellular interface, and to a *full mesh one-hop* network through a proximity wireless technology, e.g. Wi-Fi Direct. The transfer capacity on the proximity mesh is much greater than the cellular one, thus the cellular interface is the network bottleneck. Mobile devices run the P2P streaming application, while the server is a CCN Repository (i.e. a CCN node with a permanent storage), storing the video data. Notice that the group size is quite *small* (e.g. five peers), thus the scalability of the application with respects to the number of peers is not a central design issue.

### B. The streaming scheme

A CCN in meant for *pull* based services, where clients fetch contents from the network, without caring about where data comes from. Therefore, streaming schemes suitable for a CCN deployment are those in which video parts are pulled from clients, rather than being pushed by servers.

Accordingly, we use the MPEG DASH video standard, in which the video stream is temporally structured in *segments* (not to be confused with underling CCN *chunks*). A segment is an M4S file containing an interval of the video, is uniquely identified by a name (a URLs), and is available at different coding bit-rates to readily implement a receiver-driven rate adaptation. In addition to M4S files, an XML Media Presentation Descriptor file (MPD) describes the video segments through meta-information such as: the URLs, coding information, playback timing, etc. To play the video, a DASH client first fetches the MPD and then starts to pull and play M4S video segments from the network, following the MPD timing information and selecting the rate according to a local control algorithm.

### C. Video server and naming scheme

The video server is a CCN Repository, which publishes MPD and M4S files. The MPD file is available from the beginning of the video distribution, while M4S video segments are built and published during the live streaming.

To mildly synchronize peers and video source, we require the server to publish, for each new segment, a Video Timing Information (VTI), which contains the segment number of the last published video segment and the server clock.

The naming schemes of MPD, VTI and M4S data items are:

MPD ccnx:/server-prefix/filename.mpd

VTI ccnx:/server-prefix/filename.vti

M4S ccnx:/server-prefix/filename/SN=*x*/BW=*y*.m4s

where *x* is the video segment number and *y* is a coding rate. For instance 'ccnx:/foo.eu/video1/SN=10/BW=100.m4s' identifies an M4S segment where 'foo.eu' is the video server prefix, 'video1' is the file name, the segment number is 10 and the coding rate is 100 bps.

### D. Video peer

Mobile devices (aka peers), share their cellular access to cooperatively download the video stream with a coding rate higher than the one they could achieve by downloading the stream independently. In this section we show the working of a video peer and how these operations are implemented by exploiting CCN functionalities.

*Peer joining*

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. After the synchronization, the peer fetches the MPD file and begins to cooperate with other peers to pre-fetch and play segments.

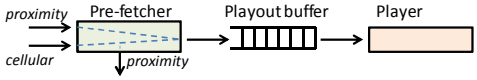


**Fig. 2 – Pre-fetcher, playout buffer and player**

*Collaborative pre-fetching*

Fig. 2 shows the main components of a video peer. A *pre-fetcher* concurrently downloads video segments from proximity and cellular interface, and upload segments received on the cellular interface on the proximity interface. All downloaded segments fill a playout buffer, drained by a DASH video player that starts the playback when the buffer contains $2P$ segments. Thus, the playout delay is $2\ PT_s$ where $T_s$ is the duration of a video segment (e.g. 2 sec).

The pre-fetcher downloads the stream in blocks of $P$ segments, named *windows* (a similar concept is used in [14]); when a new window is fully published by the source, a new *pre-fetch round* starts. At the beginning of a pre-fetch round, a peer randomly shuffles the sequence of segments to download, to avoid the occurrence of *duplicated cellular*

*fetches* (i.e. two or more peers downloading the same video segment from the cellular interface). To maximize both the use of radio resources and the achievable coding rate, the peer continuously uses both interfaces, by pulling one missing segment at a time using the cellular interface, and by trying to download all other remaining missing segments from the proximity interface. In Fig. 3 we report the time evolution of the video source, of the pre-fetchers ($P$=5) and of the players of two peers in the time period in which the source publishes the segments 15÷19. During this period, the video players play the segments 5÷9, while the pre-fetchers collaboratively download the block of segments 10÷14 from the cellular interface, sharing these segments through the proximity interface.

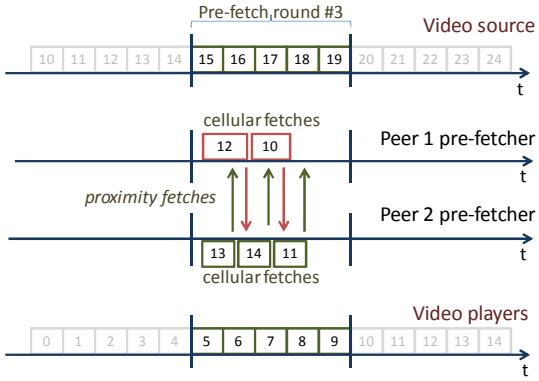


**Fig. 3 – Time evolution of the video source, and of the pre-fetchers (P=5) and of the players of two peers**

**FIB of peer1**

| Name prefix | output-face |
|---|---|
| ccnx:/prd | 224.0.0.1:9695 (wlan0) |
| ccnx:/foo.eu | 160.80.103.102:9695 (rmnet0) |
| ccnx:/foo.eu/video1/SN=11/BW=100.m4s | 192.168.0.2:9695 (wlan0) |
| ccnx:/foo.eu/video1/SN=13/BW=100.m4s | 192.168.0.2:9695 (wlan0) |
| ccnx:/foo.eu/video1/SN=14/BW=100.m4s | 192.168.0.2:9695 (wlan0) |
| … | |

**FIB of peer 2**

| Name prefix | output-face |
|---|---|
| ccnx:/prd | 224.0.0.1:9605 (wlan0) |
| ccnx:/foo.eu | 160.80.103.102:9695 (rmnet0) |
| ccnx:/foo.eu/video1/SN=10/BW=100.m4s | 192.168.0.1:9695 (wlan0) |
| ccnx:/foo.eu/video1/SN=12/BW=100.m4s | 192.168.0.1:9695 (wlan0) |
| … | |

**Fig. 4 – CCN FIBs of peers**

*Cellular fetches*

To support cellular fetches, the CCN FIB of each peer has a preloaded *remote-route* that addresses the server-prefix and points to the remote video server. For example, in Fig. 4, FIBs of peers 1 and 2 contain the remote-route 'ccnx:/foo.eu', pointing via the cellular interface (rmnet0) to the video server 160.80.103.102:9595. In Fig. 3, if peer 1 has to fetch segment #10, it requests the segment by name to the CCN API, which will forward the request and send back the data using the cellular link.

*Proximity fetches*

To support proximity fetches, a peer temporarily inserts in its CCN FIB a *proximity-route* pointing to the neighboring peer via the proximity interface. After that, the peer requests the segment by name to the CCN API, which will forward the request and send back the data using the proximity link.

For instance, Fig. 4 shows the FIB of peer 1 and 2 during the pre-fetch round depicted in Fig. 3. Peer 1 downloaded segments 10 and 12 from the cellular interface, so peer 2 inserted in its FIB the two proximity-routes pointing peer 1 (192.168.0.1:9695) through the proximity interface (wlan0).

We point out that any segment request matches with a remote-route. Anyhow, in presence of a proximity-route, the proximity is preferred, since it offers a longer prefix match.

*Proximity route discovery*

Using the proximity route discovery functionality, a peer promptly finds out the availability of proximity routes, i.e. of video segments in the neighborhood. The discovery protocol we propose exploits the CCN paradigm and works as follows. When a peer starts downloading the segment 'ccnx:/server-prefix/filename/SN=x/BW=y.m4s' from the cellular interface, it publishes a signaling message (which is like any other content from the CCN point of view) named *Proximity-Route-Info* (PRI). This message contains the control information needed to setup the related proximity route on other peers, i.e.: the IP address and CCNx port of the peer, the coding rate $y$ and the estimated net cellular rate $C_i$ of the peer (discussed later). The PRI name includes the name of the related video segment, without the coding rate component BW=$y$, and with the *control prefix* 'prd' (proximity-route-discovery). Specifically, the naming scheme of the PRI is:

PRI ccnx:/prd/server-prefix/filename/SN=x

During a pre-fetch round, peers discover the availability of missing segments on the proximity interface by continuously trying to retrieve the related PRIs. Requests of PRIs are routed-by-name on a preconfigured multicast address of the proximity interface. Indeed, the FIB of each peer is preloaded with the entry 'ccnx:/prd', pointing to such multicast address (Fig. 4). When a PRI is retrieved, peers insert the related proximity route in the FIB and then request the discovered video segment to the CCN API, so realizing a proximity fetch. At the end of the fetch operation, the proximity route is removed from the FIB to avoid the FIB growing. It is noteworthy that when peer 1 starts to fetch a video segment from peer 2, the pre-fetcher of peer 2 may still be downloading the segment from the cellular interface. In this case, peer 2 becomes the splitting point of a multicast tree. Peer 2 receives the chunks of the video from the cellular interface and relays them both to the local pre-fetcher and to the peers requesting them through the proximity interface, e.g. peer 1. In case of late discovery, the same result is obtained thanks to the CCN cache of peer 2.

*Selection of the coding rate*

At the end of a pre-fetch round, each peer computes the coding rate of the video segments that will be downloaded in the next pre-fetch round. We designed a heuristic rate-selection algorithm that operates on the basis of: i) the available video coding rates $BW_h$; and ii) the *net* rate $C_i$ that each peer may obtain through the cellular interface, i.e. the maximum download rate seen at application layer.

If a video has $L$ possible coding rates, a peer discover these rates $BW_h$ ($1 \le h \le L$) from the MPD file fetched during the join operation. Instead, values of $C_i(k)$ measured in the pre-fetch round $k$-1 are piggybacked by node $i$ in the PRI during the pre-fetch round $k$, to dynamically adapt the coding rate to the temporary conditions of the cellular links. $C_i(k)$ estimation may be done by monitoring the time needed to download video segments from the cellular interface or by downloading dummy files suitably arranged beforehand.

Bitrate selection works as follows: each peer sorts the $M$ peers in a decreasing ordered versus $C_i(k)$, and computes the coding rate $BW_h$ to be used in the round $k$ by solving the following constrained maximization problem:

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

$$\max_h \left\{ s.t. \sum_{i=1}^{\min(P,M)} J_{i,h} \ge P \right\} \quad (2)$$

The burden of a constrained maximization is necessary since cellular downloads occur on a per-segment basis, i.e. a peer cannot choose to download part of a video segment, but only a whole segment. We refer to this limitation as *quantization-constraint*. Accordingly, Eq. (1) represents the number of video segments that a peer may fully download using the cellular interface during a pre-fetch round, assuming that segments are coded with a constant bit rate $BW_h$. The maximization of (2) yields the highest possible rate $h$ such that it is possible to download all the segments of the round within the round duration.

The sum of (2) is limited to min($P$,$M$) since at most $P$ peers are required to fetch $P$ segments from cellular interface; thus, at most $P$ peers can actually carry out cellular fetches. Furthermore, even when $P \ge M$, the solution of (2) may prevent some peers to download segments from their cellular interface. Peers having $J_{i,h}>0$ do carry out cellular downloads; peers with $J_{i,h}=0$ do not carry out cellular downloads, since these peers are unable to download even a single segment within the duration of the pre-fetch round.

### E. Test-bed results

We implemented a Linux-based prototype of the streaming application using Java and the CCNx 0.7.0 tool. The video player is VLC 2.1.0, interacting with the streaming (proxy) application through a local HTTP connection.

We verified the effectiveness of the application with a test-bed formed by a video server on the public Internet, and five peers connected to each other by a Wi-Fi ad-hoc link and to the video server either through a real HSDPA cellular connection, or through an emulated connection. In the HSDPA case each laptop is tethered via USB with a mobile phone. In the emulated case, each laptop is directly connected to the server through a dedicated Ethernet link, whose rate is controlled by the Linux TC tool. We streamed the MPEG DASH video "Big Buck Bunny", whose resolution is 480p; the available video qualities are fourteen, with bit-rates ranging from 100 kbps to 4.5 Mbps; the video is formed by about 270 segments long $T_s = 2$ sec each.

*Tests with HSDPA connections*

Fig. 5 reports the coding rate of the video streaming ($BW_h$) in case of three collaborating peers connected to the server by HSDPA links with a pre-fetch window size $P$ of 10. The ticks of the y-axis of the plot indicate the first eleven available coding rates. The figure also shows the cumulative net bandwidth $C_{tot} = \sum_i C_i$ available on the cellular (HSDPA) interface, which obviously is an upper bound of any coding rate selection algorithm. The bit rate is measured on peer 1 (HTC Desire HD), which is present from the beginning of the test. We observe that upon the insertion of peer 2 (Samsung Galaxy SII) and peer 3 (Samsung Galaxy Nexus) at 140 sec and 220 sec respectively, the cumulative net cellular bandwidth and the coding rate of the streaming follow a similar behavior. It is worth noting that at the end of the test the cellular cell becomes more congested (for external reasons), and the coding rate is properly reduced.

We point out that the conservativeness of the rate selection algorithm in the selection of video rate $BW_h$ depends on the floor operation in Eq.(1). For instance, at sec 350 $C_{tot}$ is about 1950 kbps and the $BW_h$ is 1599 kbps instead of the possible 1898 kbps. In some tests we removed the floor operator, obtaining coding rates closer to the cumulative net cellular bandwidth, but suffering of some playback freezing, which we didn't experience when using the floor operator.

Fig. 6 shows the measurement of *raw* traffic (i.e. including protocol overhead) received on the HSDPA and Wi-Fi interfaces by peers 1 and 2, during the test of Fig. 5. Peer 3 obtained similar measurements. In the reported time period, all peers are present and the stream is coded at 1599 kbps. Peers use their cellular interface quite continuously and equally in terms of average bit-rate, obtaining an average raw HSDPA traffic in the order of 600÷650 kbps. These rather *low* HSDPA rates occur because we carried out the measurements in our university campus during the lunch break, when many students use their mobile phones. The sum of the raw bit rates downloaded from HSDPA is roughly 1880 kbps, as expected from collaboration: this value is also equal to the sum of the HSDPA and Wi-Fi rates obtained by a single peer. The average video coding rate measured in the specific time period of the plot is about 1520 kbps, rather than the nominal 1599 kbps; the resulting

overhead of CCN and lower layers (UDP/IP) on the HSDPA interface is about 19%, of which about 2.4% is due to UDP/IP and the rest is mainly due to the security-related information included in the CCN data units.

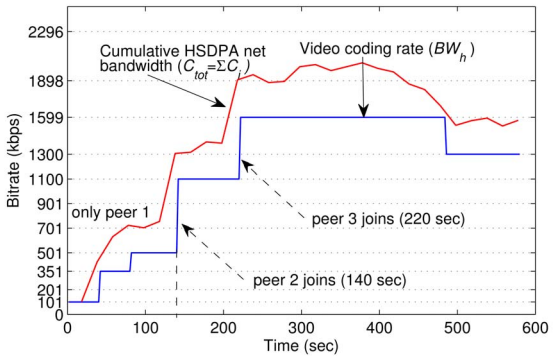


**Fig. 5 – Video coding rate and cumulative net cellular (HSDPA) rate seen by peer 1 in case of 3 peers and *P*=10**

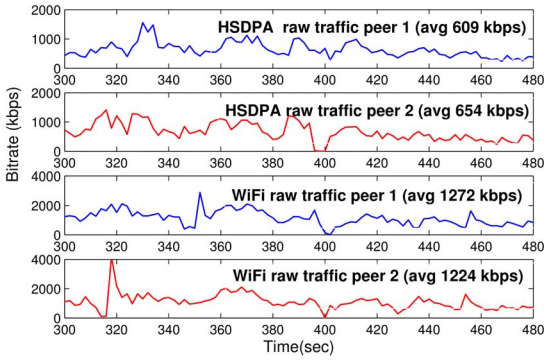


**Fig. 6 – Received raw traffic in case of 3 peers with P=10**

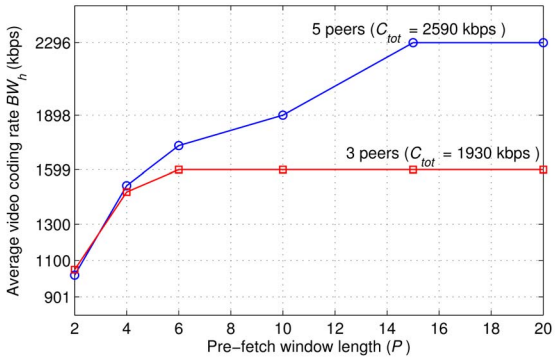


**Fig. 7 – Video coding rate versus pre-fetch window length (*P*) with emulated connection**

*Tests with emulated connections*

A fundamental parameter of the proposed scheme is the pre-fetch window length $P$: a small one would be preferable, since the playout delay is equal to $2PT_s$, but since too small windows imply inefficiencies, we need to find a trade-off.

To analyze the impact of $P$ alone on the performance, we resort to a controlled test-bed in which we *roughly* emulate the cellular interface with a dedicated Ethernet link, whose rate is controlled by the Linux Traffic Control tool.

We consider two scenarios: $M$=3 peers, with raw cellular rates of 1000, 1000 and 400 kbps, and $M$=5 peers, with raw cellular rates of 1000, 1000, 400, 400 and 400 kbps. In these scenarios, we measured a cumulative net cellular rate $C_{tot}$ of about 1930 kbps ($M$=3), and 2590 kbps ($M$=5), consistent with the 19% of protocol overhead previously estimated.

We would expect to obtain video coding rates of 1599 and 2296 kbps, i.e. the highest rates still lower than $C_{tot}$ for three and five peers. Instead, Fig. 7 shows that such coding rates are reached only when $P$ overcomes a given threshold value, i.e. $P$=6 with three peers and $P$=15 with five peers. This behavior is due to the interplay of three factors affecting the relationship between $P$ and $M$, and whose effects tend to vanish as $P$ increases:

- (for $P<M$): the quantization-constraint imposes that no more than $P$ peers can perform cellular fetches. Thus, for $P<M$ the cellular links of $M$-$P$ peers cannot be exploited;
- (for $P<2M$): duplicated cellular fetches may occur in case of a small pre-fetch window, e.g. $P<2M$. This leads to a waste of the cumulative net cellular bandwidth and to temporary reductions of the coding rate. In the tests, duplicated cellular fetches show up for $P<2M$, making the coding rate switch between two close available levels. Consequently, the *average* coding rate does not strictly match any available coding rate (y-axis ticks);
- (for $P<3M$): the conservativeness effect of the floor operator of Eq.(1) is more severe for smaller pre-fetch windows. During the tests this limitation vanished for $P\geq 2M$ ($P\geq 3M$) in case of three (five) peers.

Following these *specific* results, and considering that the effects of the floor operation in Eq. (1) vanish for great values of $P$, a dimensioning approach that only consider the effects of Eq. (1) can be derived: we set an *exploitation target* in terms of a percentage $T$ of the cumulative net cellular bandwidth $C_{tot}$, and search the minimum value of $P$ supporting a stream coded at $T{\cdot}C_{tot}$. To this end, we assume to have an ideally coded video providing any possible rate and solve the following constrained minimization problem

$$\min_P \left\{ s.t. \sum_{i=1}^{P} J_{i,h} \geq P \right\} \qquad (3)$$

where $J_{i,h}$ is evaluated with Eq. (1), by assuming $BW_h = T \sum_i C_i$. We solve the minimization problem using MATLAB and assuming $M$={3,5} peers with a cellular rate $C_i$ that is a uniform random variable in the range R ± Δ·R. This specific distribution makes independent from the mean rate $R$ the random variable $C_i / (T \sum_i C_i)$ of Eq.(1), and thus the result of the minimization (3) as well.

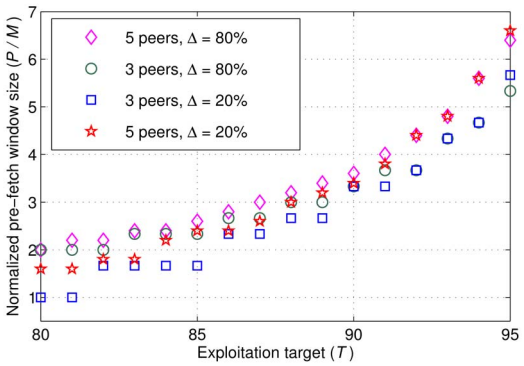


**Fig. 8 – Simulated minimum pre-fetch window length (P) versus exploitation target (T)**

Fig. 8 reports the values of *P*, normalized to the number of peers *M* and as a function of the exploitation target (*T*). For a given value of *T*, we observe that *P*/*M* mildly increases when we increase: i) the variability of cellular rates among peers, e.g. from Δ=20% to Δ=80%, or ii) the number of peers, e.g. from 3 to 5. However, for a given *T* the related differences are quite small. Conversely, valuable differences show up by changing the exploitation target *T*, so the more we want to exploit the cellular resources, the greater has to be the pre-fetch window *P* (and the playout delay). These results can be used to dimension *P*: for instance, in the case of Fig. 8, if we aim at exploiting the 85% of the cumulative cellular rate, we must choose a pre-fetch window size between 2 and 3 times the expected maximum number of peers. If the corresponding playout delay is not acceptable, we must reduce the duration of the video segments.

## IV. Related Works and Conclusions

In this paper we presented a P2P CCN application for *live streaming* of videos encoded at *multiple bitrates*. The P2P cooperation allows peers to play a video at a coding rate close to the sum of their cellular capacities, to *improve video quality*. Some similar literature works are [10][11][12][14] and [17]. In [10] we propose a CCN P2P *on-demand single-rate* video streaming application for mobile cellular devices, whose goal is to *offload the cellular interface*. In [11] the authors propose a CCN adaptive video streaming application named AMVS-NDN, which enables a mobile device *either* to use its own 3G/4G connection *or* to connect via WiFi to another mobile device to exploit its possibly better 3G/4G link. AMVS-NDN cooperation logic resembles a selection of the best cellular gateway, thus the achievable video coding rate is bounded by the cellular rate of this *single* gateway. In [17] the authors set up a test-bed for video streaming over CCN, named NDNVideo: its 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 [12] the authors propose a *TCP/IP* application dealing with *on-demand single-rate* video streaming. In [14] 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 [16], 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*. Clearly many other papers on P2P video streaming exist (see e.g. [15] and its references) and the research topic is well-known. However, our focus is on ICN/CCN and adaptive video streaming exploitation. Future work includes the porting of the application on Android devices and a Java-based version of the application.

## References


[1] T. Koponen, M. Chawla, B.G. Chun, et al. "A data-oriented (and beyond) network architecture", ACM SIGCOMM 2007

[2] V. Jacobson, D. K. Smetters, J. D. Thornton et al., "Networking named content", ACM CoNEXT 2009

[3] A. Detti, M. Pomposini, N. Blefari-Melazzi, S. Salsano, "Supporting the Web with an Information Centric Network that Routes by Name", Elsevier Computer Networks, vol. 56, Issue 17, p. 3705–3722

[4] CCNx project web site: www.ccnx.org

[5] A. Ghodsi, S. Shenker,T. Koponen, A. Singla, B. Raghavan, J. Wilcox, "Information-centric networking: seeing the forest for the trees", ACM Workshop on Hot Topics in Networks 2011.

[6] S. Salsano, A. Detti, M. Cancellieri, M. Pomposini, N. Blefari-Melazzi, "Transport-layer issues in Information Centric Networks", ACM SIGCOMM Workshop on Information-Centric Networking (ICN 2012), August 17, 2012, Helsinki, Finland.

[7] Z. Zhu, S. Wang, X. Yang, V. Jacobson, L. Zhang; "ACT: Audio Conference Tool Over Named Data Networks", ACM SIGCOMM ICN Workshop 2011

[8] J. Burke, P. Gasti, N. Nathan, G. Tsudik, "Securing Instrumented Environments over Content-Centric Networking: the Case of Lighting Control and NDN", IEEE NOMEN 2013

[9] T. Stockhammer, "Dynamic adaptive streaming over HTTP: standards and design principles", ACM MMSys 2011

[10] A.Detti, M. Pomposini, N. Blefari-Melazzi, S. Salsano, A. Bragagnini, "Offloading cellular networks with Information-Centric Networking: the case of video streaming", IEEE WoWMoM 2012

[11] B. Han, N. Choi, T. Kwon, Y. Choi, "AMVS-NDN: Adaptive Mobile Video Streaming and Sharing in Wireless Named Data Networking" , IEEE NOMEN 2013

[12] L. Keller, A. Le, B. Cici, H. Seferoglu, C. Fragouli, A. Markopoulou, "Microcast: Cooperative Video Streaming on Smartphones", ACM MobiSys 2012

[13] http://netgroup.uniroma2.it/Andrea_Detti/ICNvideo-live-DASH

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

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

[16] Z. Xiaoqing, P. Agrawal, J.P. Singh, et. al "Distributed Rate Allocation Policies for Multihomed Video Streaming Over Heterogeneous Access Networks" IEEE Transaction on Multimedia, Vol. 11, No. 4, June 2009

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