Are Information-Centric Networks Video-Ready?

Christos Tsilopoulos, George Xylomenos and George C. Polyzos
Mobile Multimedia Laboratory, Department of Informatics
Athens University of Economics and Business
Patision 76, Athens 10434, Greece
Email: tsilochr, xgeorge, polyzos@aueb.gr

Abstract—**Video constitutes the majority of all Internet traffic and its share is expected to grow. Any future Internet architecture with a chance at success should provide some tangible benefits for video applications. Information-Centric Networking (ICN) architectures were designed with the specific goal of improving content distribution on the Internet; thus, this paper attempts to answer the obvious question: is ICN appropriate and ready for video traffic and, if not, what is missing or should be modified? To this end, we consider two different ICN architectures, *Content-Centric Networking* (CCN) and *Publish-Subscribe Internetworking* (PSI), and examine their applicability to Video on Demand and Live Video Streaming applications. Our goal is to clarify what ICN already does well for video, what it still needs to do, and, most importantly, what it could or should do differently.**

I. Introduction

The Internet architecture and the TCP/IP protocol suite are undoubtedly technological successes. What started out as an academic experiment, has become a global communication infrastructure with billions of connected devices delivering voluminous amounts of digital content. This success was accompanied by a shift in usage: the Internet was initially designed to interconnect hosts for sharing scarce computational resources, while nowadays it is mostly used for ubiquitous content retrieval, with video dominating Internet traffic [1]. Several add-ons and patches have been used to address the changing needs of Internet users, such as NATs to extend the scarce device address space and DNS redirections to serve large traffic volumes from the nearest servers. These solutions however either created their own problems [2] that were solved by yet more ad-hoc solutions [3], or were misused [4]. Overall, this has led to a situation where network operation and management have become too complicated, while evolution in the core of the protocol stack has slowed down.

This state of affairs motivated research in *clean-slate* architectural designs with *Information-Centric Networking* (ICN) holding a prominent position. ICN aims to facilitate content distribution by placing *self-identified information items* at the heart of the protocol stack and building routing and transport protocols around them. Several ICN designs have been proposed with substantial differences in the provided service models and core network functions, including information lookup, routing, forwarding and transport [5]. If these architectures are ever going to have an impact on the Internet however, it is imperative for them to work well with video. Internet video accounted for 57% of all Internet traffic in 2012 and is expected to reach 69% in 2017 [1], hence improvements in video delivery could be a major factor for the successful adoption of ICN.

However, implementing, say, video streaming over ICN is not trivial. Internet video streaming technologies have evolved based on the properties which no longer hold with ICN. For example, the unreliable best effort nature of IP forced video applications to include techniques for stream adaptation and loss tolerance, while delivery mechanisms were designed on top of transport protocols with well-defined interfaces and behavior, e.g. RTP/UDP and recently HTTP/TCP [6]. In contrast, many proposed ICN architectures are at an early stage of development, hence their service models and transport interfaces are far from final. Therefore the benefits of applying the same techniques for video streaming in an ICN context are questionable, for example, it may be impossible for video receivers to adapt to network path characteristics when each video chunk can be served by a different caching node. Indeed, support for efficient video delivery in ICN may require enhancements in the network design, e.g. developing specific transport protocols, extending the transport interfaces and reconsidering the end-to-end paradigm. It may even be the case that some of the basic design choices of ICN architectures are problematic for video delivery.

In this paper we consider the design of video streaming applications in two particular ICN proposals: (i) *Content-Centric Networking* (CCN) [7] and (ii) *Publish-Subscribe Internetworking* (PSI) [8]. We selected these particular ICN architectures because they exhibit fundamentally different architectural designs. CCN provides users with a *request-response* service model in which users *pull* individual data packets from the network. PSI on the other hand, offers a *publish-subscribe* service model in which the network *pushes* data to interested users. We also consider the implementation of two classes of streaming applications, with very different transport requirements: (i) *Video on Demand* (VoD) and (ii) *Live Video Streaming* (LVS). VoD is a file transfer application with the extra feature that the video file is viewed while being downloaded, hence reliability is a relatively high priority for VoD. LVS on the other hand is delay intolerant and, by necessity, more loss tolerant than VoD. The questions we want to answer are (i) what ICN already does well for video, (ii) what it still needs to do and, most importantly, (iii) what it could or should do differently.

The remainder of this paper is organized as follows. In Section II we present the current state of affairs for VoD and

LVS streaming on the Internet. In Section III we examine in detail how these applications can be supported in CCN, while in Section IV we do the same for PSI, paying particular attention to issues that still need to be addressed and aspects of the architecture that may need to be changed for video. We present our conclusions in Section VI.

II. Internet Video Transport

In this section we outline how viddeo applications operate on the current Internet, as background for the ensuing discussion of video support in the ICN architectures considered.

A. Video on Demand

In *Video on Demand* (VoD), users download and view video that is prerecorded and stored in an Internet host, e.g. a dedicated video server. VoD supports VCR-style functionalities such as *pause*, *rewind* and *fast-forward*. From the perspective of network transport, VoD is similar to other file transfers, with the addition that the file is used (viewed) before completing the transfer. In most VoD services, video is delivered reliably, hence VoD services are usually implemented on top of TCP. In recent years, VoD over HTTP is increasingly getting popular, mostly because of the compatibility of HTTP with Internet middleboxes (firewalls and NATs). The transport details, however, remain the same; control and data signaling are wrapped in HTTP messages, with the actual transfer performed by TCP.

VoD applications do not have very strict delay requirements. Although smaller delays are better, large delays or delay fluctuations are not catastrophic. VoD players may pause playback and resume and as soon as their buffers fill with data. Of course, long delays and buffering times have a negative effect on user *Quality of Experience* (QoE). A common enhancement, *adaptive streaming*, offers the same video in multiple qualities (and bit rates). The transmitted quality is selected dynamically, either by the sender or the receiver, depending on the estimated end-to-end bandwidth throughput.

VoD viewers are not synchronized, therefore video requests are served individually and video packets are transferred via unicast. VoD data may be delivered via multicast, but that requires explicit synchronization of the viewers [9] and dropping some of the VCR-style functionality. In this case, multicast must be accompanied with a multicast transport protocol handling error, flow and congestion control such as pgmcc [10]. Although multicast transport protocols have been developed for IP networks, they have not found their way into the Internet ecosystem and have remained confined to research environments. For this reason, in the remainder of this paper we will only consider unicast VoD applications.

B. Live Video Streaming

In *Live Video Streaming* (LVS), video frames are captured, compressed, packetized and immediately transmitted to viewers. From the network transport point of view, LVS differs from VoD in two major aspects. First, LVS applications have critical low-delay requirements. End-to-end retransmission-based error control is not suitable in situations with large *Round Trip Times* (RTTs) whereas *Forward Error Correction* (FEC) schemes consume precious bandwidth. Hence, LVS applications normally use their own mechanisms for tolerating packet loss, such as error masking and adaptive streaming, over UDP.[1] The second difference is that LVS viewers are naturally synchronized in time, thus data can be delivered via multicast. In addition, since end-to-end error control is not employed in LVS, the overall transport scheme is less complex, compared to reliable multicast transport for VoD.

On the other hand, multicasting live video must cope with various forms of system heterogeneity, in terms of bandwidth (e.g. wired vs. wireless links) and hardware capabilities (HD screens vs. laptops vs. smart phones). Heterogeneity is addressed by schemes such as *Receiver-driven Layered Multicast* (RLM) [11]. In RLM, video is coded in multiple quality layers and each layer is mapped to a separate multicast group. Users subscribe to (and receive) as many multicast groups as their connectivity status allows. As with reliable mutlicast transport, RLM has not yet made it to the commercial Internet due to lack of IP multicast support. LVS is therefore most prevalent in controlled and homogeneous uniform environments, such as ISP networks offering IPTV services over cable or DSL.

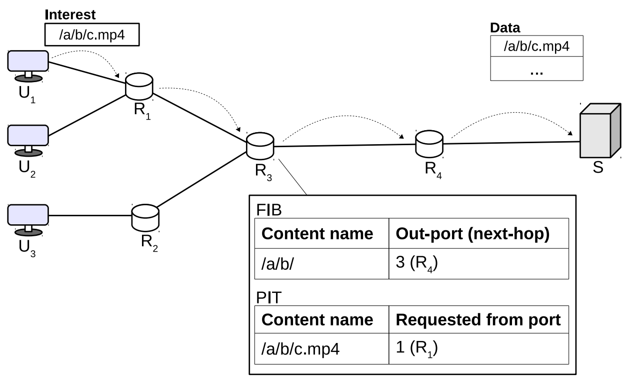
III. Video Delivery in Content-Centric Networking

A. Architecture overview

The *Content-Centric Networking* (CCN) architecture places *named content packets* at the thin waist of the protocol stack and provides users with a *request-response* service model in which users *pull* data packets from the network [7]. Users request *named* Data packets via *Interests*. Data packets contain a name that uniquely identifies the carried payload whereas Interests carry the name of the requested Data; no host addresses are used. For each Interest, a user receives *at most* one Data packet, thus there is a strict one-to-one relation between Interests and received Data packets. The structure of content names is similar to URIs: names are hierarchical with variable-length components, e.g. */a/b/c.mp4*.

Routers propagate Interests towards content sources and return Data packets along the reverse path with the help of three data structures: (i) the *Forwarding Information Base* (FIB), (ii) the *Pending Interests Table* (PIT) and (iii) the *Cache Store* (CS) (Figure 1). The FIB serves as a name-based routing table for forwarding Interests. At each arriving Interest, routers perform a *Longest Prefix Match* on their FIB and push the Interest towards a content source. FIBs are populated through name-based routing protocols either similar to the ones used in IP [7] or new ones [12]. Before forwarding an Interest, routers insert the Interest in the PIT, noting its incoming interface. If a PIT entry for the same name exists, meaning that the router has already forwarded an Interest for the same packet, the Interest is dropped, otherwise it is forwarded. This operation continues

[1]Recently, HTTP/TCP has been used for LVS, but this is mostly due to the compatilibity of HTTP with Internet middleboxes.



FIB

Content name	Out-port (next-hop)
/a/b/	3 (R_4)

PIT

Content name	Requested from port
/a/b/c.mp4	1 (R_1)

Fig. 1. A sample CCN interaction. U_1 issues an Interest for */a/b/c.mp4* which is located at S. Arrows show the propagation of the Interest. Data follows the reverse path. The FIB and PIT (not the CS) for R_3 are shown.

until the Interest reaches the content source which responds with the requested Data packet. Data packets are delivered by reversing the path taken by the Interest. At each hop, routers check their PIT for a matching entry, transmit the Data packet backwards and delete the PIT entry. If a router had received Interests for the same Data from multiple interfaces, the router duplicates the Data packet, thus realizing *multicast* delivery. Finally, the CS, as it name suggests, is a cache containing Data packets. Intermediate routers may directly serve an Interest if the requested Data is present in the CS. Routers fill their CSes with traversing Data packets, but their detailed operation (e.g. cache replacement policy and interaction with the routing system) is an open research issue [13].

Apart from multicast, CCN inherently supports *anycast*, when the same data is available in multiple locations, and Data *multihoming*, when a content source can be reached through multiple paths. These two features are implemented by having the FIB point to multiple next-hops for forwarding an Interest. The Interest route (and consequently the Data route) is decided by on-path routers through a CCN module called the *strategy layer*. Anycast and/or multipath is transparent to users.

The Interest-Data mechanism essentially constitutes a *request-response* communication model. Data transport is receiver-driven, with stateless senders (content sources), i.e. senders do not maintain any connection state, simply responding to incoming Interests. Transport related issues are handled by the receiver. For error control, the receiver retransmits Interests for missing Data packets. Flow and congestion control are also applied by the receiver by pipelining Interests. For example the amount of transmitted Interests can be controlled with a TCP-like sliding window mechanism [14].

B. Video on Demand

The transport interface in CCN is almost identical to HTTP's *request-response* model, therefore a VoD application in CCN works similarly to chunk-based video delivery over HTTP with two notable differences. First, in CCN requests are made on a packet granularity: an Interest results in the delivery of at most one Data packet, as opposed to HTTP in

```
<video>
    <quality level="low">
        <chunk id="1" from="0s" to="2s" no_packets="10" />
        <chunk id="2" from="2s" to="4s" no_packets="10" />
    </quality>
    <quality level="high">
        <chunk id="1" from="0s" to="2s" no_packets="20" />
        <chunk id="2" from="2s" to="4s" no_packets="20" />
    </quality>
</video>
```

Fig. 2. Example metadata for video */a/b/c.mp4*.

which a request for a video chunk causes the transmission of a number of TCP segments/IP packets. Second, in CCN the user is unaware of the serving host, since the network selects where and how to forward an Interest. This affects receiver-driven adaptivity mechanisms, since users cannot reliably estimate end-to-end throughput.

A VoD application in CCN would operate roughly as follows. The receiver issues an initial Interest for the video, say */a/b/c.mp4*, and receives a Data packet containing metadata. The metadata contains the information required to construct the names of individual Data packets, e.g. naming scheme, number of packets etc. Once this information is available, the receiver constructs Interests for each video packet and starts downloading the video. Adaptive streaming can be supported by adding corresponding metadata, as in *Dynamic Adaptive Streaming over HTTP* (DASH) [6]. Figure 2 shows an example of a metadata packet for video */a/b/c.mp4*. The metadata indicates that the video is offered in two quality layers (*low* and *high*), also showing the number of chunks of each quality layer, the duration of each chunk and the number of packets per chunk. Unlike with DASH, packet-level information is required because the size of Data packets is constrained by the *Maximum Transfer Unit* (MTU) and the receiver needs to request each Data packet separately.[2] In the example, the receiver may decide to start with the *low* quality and download the first chunk by issuing 10 Interests, namely */a/b/c.mp4/low/chunk-1/1* to */a/b/c.mp4/low/chunk-1/10*.

Since Interests are forwarded based on *Longest Prefix Matching* at each router, no additional routing information is needed in routers for VoD. On the other hand, CCN does not guarantee that Interests are always served by the same content source. Interests may be served by the CS of an intermediate router or they may be routed to different sources, if the content is available in multiple locations, therefore the content can be downloaded from the best available location. In addition, by storing packets at the CS of intermediate routers, lost packets can be quickly recovered from nearby routers, thus enhancing reliability without inflating delay.

However, while native anycast support is an advantage of CCN (and ICN in general), it complicates the design of VoD applications. Receiver-driven adaptive streaming is usually based on estimations of the end-to-end bandwidth throughput [16]. In the case of CCN, it is unclear whether

[2]The MTU is implementation and deployment specific. CCNx [15], the open-source CCN prototype implemented on top of UDP, uses an MTU of 8800 bytes.

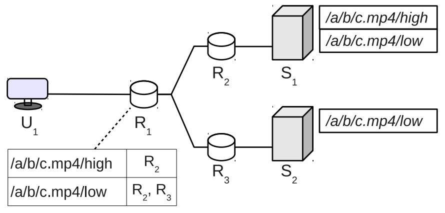


Fig. 3. Video */a/b/c.mp4/low* is located at S_1 and */a/b/c.mp4/high* is located at both S_1 and S_2.

such estimations can be reliably performed. We illustrate this with the example of Figure 3. Video */a/b/c.mp4* is available in two quality versions. S_1 stores both versions of the file, while a copy of the low quality version is also located at S_2 and appropriate routing information is installed in routers. U_1 starts viewing the video in high quality, thus video chunks are downloaded from S_1. If there is a capacity drop in the $S_1 \rightarrow U_1$ path (e.g. due to congestion), U_1 may switch to the low quality. At that point, it is up to R_1 to decide how to forward Interests, e.g. R_1 may choose to forward Interests towards S_2. However, the capacity of the $S_2 \rightarrow U_1$ path may be considerably smaller that the $S_1 \rightarrow U_1$ path, hence the QoE may drop further, despite the switch to the low quality version. Under these conditions, U_1 may improve its QoE by switching back to high quality, which will have to be fetched from S_1, contradicting the current assumptions of adaptive streaming. U_1 would maximize QoE by indicating that the low quality version should be fetched from S_1, but this is impossible in CCN due to the lack of endpoint addresses. Instead, a CCN VoD application should test both quality layers in order to determine which offers the best QoE, relying on the strategy layer to determine the best path for each content layer.

C. Live Video Streaming

Support for LVS in CCN has attracted interest as soon as the architecture appeared [17], [18]. Instead of having the source transmit LVS datagrams to named-hosts at will (as in UDP/IP), receivers in CCN need to request each named Data packet separately. The starting phase for LVS is similar to that in VoD, i.e. a receiver initially issues an Interest for the stream and receives a Data packet containing descriptive metadata. The metadata includes the naming scheme for deriving the names of individual packets and the name of the *latest* packet so that the receiver can construct the names for upcoming Data packets. Once this information is available, the receiver starts transmiting Interests for individual Data packets.

A difference between LVS and VoD is that streaming packets need to be delivered in real-time, i.e. packets need to be transmitted as soon as they are generated. In addition, a user cannot request one Data packet at a time, otherwise Data packets will be delivered with an inter-arrival delay of at least one *Round Trip Time* (RTT) which would cause jitter in the media playback. Real-time delivery is achieved by having receivers request a number of streaming packets upfront and issue subsequent Interests with a rate analogous to the stream's data rate. For example, consider a stream named with the prefix */a/b/c/live*, transmitting data with a *fixed* rate of 100 *packets per second* (pps) and that 2000 packets have already been transmitted. A viewer that wishes to *tune in* to the stream, estimates the RTT, say 100ms and decides to request Data packets for an interval of 2 RTTs, which is $2 \times (100\ pps \times 100\ ms) = 20$ packets. The user requests packets */a/b/c/live/2001* to */a/b/c/live/2020* and from there on she transmits subsequent Interests with a rate of 100 Interests per second.

The proactively issued Interests are forwarded to the content source and await there until the requested Data packet is generated and then immediately transmitted. If, in the mean time, other users issue Interests for the same packets, these Interests are suppressed by common on-path routers. When the respective Data packets arrive, these routers duplicate the Data towards the Interested receivers, thus live streaming packets are multicasted to viewers. In addition, the receiver-driven layered multicast model [11] can be implemented in a straightforward manner. Information about the available video layers can be included in the metadata during the initialization phase and viewers transmit the Interests for the packets of each desired layer. For example, in H264 Scalable Video Coding, scalable layers are characterized by three parameters: (i) the Dependency ID (DID) for spatial scalability (frame resolution), (ii) the Quality ID (QID) for quality (SNR) scalability and (iii) the Temporal ID (TID) for temporal scalability (frame rate) [19]. A stream source includes these parameters in the metadata Data packet and receivers may request upcoming Data packets of each scalable layer by transmitting Interests for $/stream\_name/DID_i/QID_i/TID_i/[packet\_num]$. Under this scheme, adapting to network conditions requires from the receiver to simply request the packets for the desired layers. No out-of-bound communication is needed for sending feedback to the media source in order to change its behavior.

A CCN feature that may benefit live streaming is the existence of Cache Stores (CS) in routers. Although live streaming applications are tolerant to packet losses, sometimes there may be adequate time to request the retransmission of a lost packet [20]. In the CCN context, lost Data packets may be recovered from an intermediate router's CS and not necessarily from the stream source, thus the recovery may take less that an end-to-end RTT. In this context, there is an additional incentive for attempting to recover packet losses. The operation may be enhanced by distinguishing video packets with respect to the cache replacement policy in CSes. For example, packets carrying data belonging to *B* frames may be not be cached at all, leaving room in Cache Stores for *I* and *P* frames. Moreover, P frames may be discarded sooner than I frames (indicated with a TTL value in the Data packet header) so that there is a higher cache hit probability when lost I-frames are requested.

A potential drawback for supporting live streaming in CCN is *one Interest per Data packet* mode of operation. Requesting each individual streaming packet introduces significant load to the network (bandwidth spent for Interests, load in routers) and raises scalability concerns. Second, live streaming quality is also subject to the up-path conditions, i.e. the receiver-to-source path, apart form the down-path conditions. When the up-path is congested, Interests may be lost thus the respective Data packets are not transmitted at all. In this case the QoE degrades even though there may be sufficient bandwidth resources in the data-path. Such cases may occur when users get access to the network through asymmetric links, e.g. ADSL or satellite networks. These issues can be dealt by relaxing the strict one-to-one mapping between Interests and Data, for example introducing *Interest Aggregation* [21] or *Persistent Interests* [18]. With Interest Aggregation, a single Interest requests multiple Data packets at once whereas Persistent Interests mimic IP's channel mode and request all packets belonging to a stream. These modifications however, need to be thoroughly tested and cannot serve as *off the shelve* solutions. For example, in Interest Aggregation, loss of a single Interest will result in loss of multiple Data packets thus the user will experience bursty losses. Persistent Interests on the other hand have longer lifetimes than plain CCN Interests and the system may suffer from scalability issues with respect to memory requirements in routers, similar to the scalability constraints of IP multicast. Intermediate solutions, e.g. interleaved Batch Interests or Persistent Interests with shorter lifetimes (soft-state) may over-complicate router operation.

IV. Video delivery in Publish Subscribe Internetworking

A. Architecture overview

The *Publish Subscribe Internetworking* (PSI) architecture [8] shares many high-level goals with CCN, however, it follows an entirely different design approach. Compared to CCN, PSI differs in two major aspects. First, the notion of content objects in PSI is more abstract, i.e. a content object is not strictly mapped to a single data packet. Second, the core network operation (routing requests and forwarding data) is split in three distinct functional modules, emphasizing on the decoupling of routing control from data packet forwarding.

PSI models content objects as *publications*, content sources as *publishers* and content consumers as *subscribers*. The architecture provides users with a pub/sub API for announcing and requesting data. The architecture organizes the core network operation into three distinct subsystems: (i) the Rendezvous subsystem (RVS), (ii) the Topology Management and Path Formation subsystem (TMPFS) and (iii) the Forwarding subsystem (FS) [8]. The RVS serves as a resolution system; it maps available publications to publishers and resolves subscriptions. Nodes that implement RVS functionality are called Rendezvous Nodes (RNs) and they are organized as a *Distributed Hash Table* (DHT). The TMPFS monitors the network topology and connectivity state. The TMPFS functionality resides in Topology Manager nodes (TMs), which may be co-located with RNs or placed in separate physical hosts. The FS undertakes the actual packet forwarding. PSI employs LIPSIN [22] for packet forwarding, an efficient Bloom filter-based source-routing scheme. The forwarding functionality is implemented by all nodes. Nodes that participate solely in

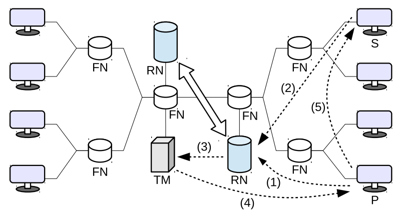


Fig. 4. A sample PSI interaction. P announces a content object to the network (step 1). S subscribes to it. The RVS handles the subscription (step 2), the TMPFS selects the delivery path (step 3) and notifies P (step 4). P transmits the requested object over the specified path (step 5).

packet forwarding are called Forwarding nodes (FNs). All nodes report their forwarding related information (link status) to the TMPFS so that the latter can make routing decisions. Figure 4 shows an example of a PSI network with two RNs, a single TM and several FNs.

Content delivery in PSI is step-wise process in which the three subsystems interact as shown in Figure 4. Initially, a publisher announces the availability of a publication to the RVS (step 1). To receive a publication, a user issues a subscription which is handled by the RVS as well (step 2). The RVS locates and selects the *best* publisher and requests the TMPFS to compute the publisher-to-subscriber path (step 3). The TMPFS selects a *suitable* path and encodes it into a LIPSIN source-route. The TMPFS hands the source-route identifier to the publisher and instructs the publisher to forward the requested item (step 4). Finally, the publisher transmits the requested content object over the specified path (step 5).

PSI does not enforce a particular naming scheme. The architecture is compatible with both flat and hierarchical identifiers. The granularity of publications is not mandated by the architecture either. Publications may represent files (of arbitrary size), chunks of files or individual network packets. Publications may also represent live streaming flows of undefined size. It is left to the application to decide how to represent a content object taking into consideration the transport context and that each available publication must be announced separately to the RVS. For example, it is not advisable to represent a live stream as a series publications, one for each data packet, as in CCN. Doing so in PSI would require that each stream packet is announced to the RVS and users issue subscriptions that are handled by the RVS-TMPFS subsystems. This triangular *resolution→path formation→notification* process may lead to excessive network load and large delays. In addition, since streaming packets have relatively short lifetimes, they would need to be unpublished (withdrawn) shortly after they are published, thus further burdening the network operation. Live streams are better represented with a single publication: stream sources announce the stream once and receivers *tune in* with a single subscription. Once the subscription is resolved, the source transmits the streaming packets to the subscriber until the latter withdraws the subscription. Note that an explicit *unsubscription* message is also required.

For multicast delivery, the RVS tracks users subscribed to a content object. Whenever a user subscribes (unsubscribes) to (from) an object, the RVS updates the list of subscribed users for the particular publication and requests the TMPFS to compute a multicast delivery tree. Due to the properties of LIPSIN forwarding, multicast requires installing little to none forwarding state at FNs [23]. Once multicast forwarding state is setup and the respective LIPSIN identifier is constructed, it is communicated to the content source. PSI decouples routing control from routers (FNs) and delegates path computation to the TMPFS. This design choice has the advantage of making optimal routing decisions feasible, for example multicasting data over minimum-cost Steiner trees [24]. These optimizations are too complex to implement in IP and CCN in which routing and forwarding are strongly coupled and routing decisions rely on distributed mechanisms. On the other hand, centralized designs often face scalability issues, e.g. computation delays in the TM nodes.

B. *Video on Demand*

In a VoD service over PSI, users subscribe to the desired video. Subscriptions are handled by the network which locates the video source, computes the source-to-receiver path and instructs the source to transfer the video. Data packets are transferred on an end-to-end basis through a TCP-like mechanism that handles error, flow and congestion control. The transport operation may be driven either by the source [25] or by the receiver [26]. As described, it is up to the application to decide on the granularity of publications, i.e. whether a single publication is announced for the entire file or the file is split in chunks and separate publications are announced.

Since the data transport can be controlled by either of the communicating end-points, PSI supports both sender and receiver driven adaptive streaming. A chunk-based VoD application operates as in HTTP/DASH. Each chunk is represented with a publication and receivers request the video chunk by chunk. As in CCN , the effectiveness of receiver-driven adaptivity is questionable since the network (more specifically the RVS) may resolve subscriptions to different publishers, thus receivers cannot make safe assumptions for the end-to-end bandwidth throughput. In addition, in PSI each subscription must be handled by the RVS-TMPFS, thus there may be additional delays for flow establishment.

The centralized route control applied in PSI offers new capabilities in video delivery. Instead of having the endpoints re-actively adapt to bandwidth fluctuations, video delivery may be proactively assisted by the network. For example, the user subscribes to the video and the network *selects* an appropriate source based on the end-to-end path capacity and the offered video quality. Figure 5 shows a example where *video.mp4* is available in three quality versions (*high*, *medium* and *low*) and stored in three different locations (S_1, S_2 and S_3 respectively). The user subscribes to *video.mp4* and lets the network examine which version of the file can be better delivered to the user. For example, the $S_1 \rightarrow U_1$ path may have the largest capacity among the three paths but it may not suffice to transfer the high quality video. The path $S_2 \rightarrow U_1$ may have less capacity than $S_1 \rightarrow U_1$ but it may be fast enough for the delivery of the

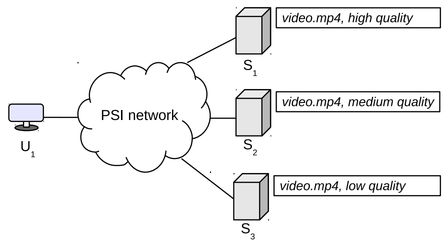


Fig. 5. U_1 subscribes to *video.mp4*. The network selects the best available version of the video considering location of the sources and the respective end-to-end paths.

medium quality, thus the network chooses to deliver the video from S_2. To implement such functionalities, the publish and subscribe semantics of PSI need to be extended and be more expressive so that announcing and requesting data contains additional information (e.g. transfer parameters) and not just the content's name. This is a situation where investigation for efficient video delivery calls for changes in the network architecture, instead of simply adapting video delivery on top of the currently offered service model.

C. *Live Video Streaming*

A live streaming application in PSI resembles live streaming in IP. The streaming flow is assigned with a name and users receive streaming packets by subscribing to the flow *once*; the network establishes the respective forwarding paths (or trees) so that the streaming flow in delivered to subscribed user(s). The internal network operation of PSI however differs from IP. In IP, subscriptions (i.e. multicast JOIN messages) are handled by IP routers through distributed multicast routing protocols whereas in PSI subscriptions are handled by the dedicated network elements that are part of the RVS and TMPFS subsystems. To stop receiving streaming packets, a user must *unsubscribe* from the stream.

PSI facilitates live streaming in two ways. First, as discussed, PSI natively supports multicast, though in a different fashion compared to IP and CCN. Multicast tree construction is controlled by the RVS and TMPFS. The choice for decoupled centralized routing control enables PSI to make optimal routing decisions, such as multicast delivery over minimum-cost Steiner trees [24], [27]. Live streaming in PSI is compatible with the layered multicast model as well; all a user has to do is to subscribe to the layers that match her connectivity status.

On the other hand, subscriptions (and unsubscriptions) must be handled by the RVS and TMPFS and are not directly forwarded to content sources as in CCN and IP Source Specific Multicast [28]. Hence, there may be large delays when joining and/or leaving a stream. This may have a negative effect in live streaming applications with high group dynamicity, e.g. channel surfing in IPTV, or frequent changes of receiving layers in receiver-driven adaptive streaming.

	CCN			PSI		
	Improved	**Unclear**	**Problematic**	**Improved**	**Unclear**	**Problematic**
VoD	Native anycast support. Enhanced retransmission-based error control with in-network packet-level caching.	Unreliable end-to-end throughput estimation for receiver-driven adaptivity.	Network overhead for explicitly requesting individual Data.	Native anycast support. Optimal path selection through centralized route control.	Unreliable end-to-end throughput estimation for receiver-driven adaptivity. Optimal path selection requires extensions to pub/sub primitives.	Delays for centralized resolution of subscriptions and unsubscriptions.
LVD	Enhanced retransmission-based error control with in-network packet-level caching. Packet distinction (e.g. I, P, B frames) in caching policies.	Service degradation in asymmetric links. Lost Interests upstream result in missing Data on the downstream.		Optimal multicast delivery through centralized route control.	Scalability of centralized multicast tree construction with dynamic user behavior.	

TABLE I
COMPARISON.

V. DISCUSSION SUMMARY

We summarize our discussion points in Table I which presents the architectural features of the examined ICNs with respect to video delivery. More specifically, Table I shows which features may *improve* video delivery compared to IP, what seems *unclear* and needs to be clarified and, last, what looks *problematic* and we believe must be sorted out in immediate research.

CCN inherently supports anycast which facilitates VoD since the network may locate the nearest cached copies of video chunks, thus reduce transfer time. In addition, packet-level caching in routers may assist retransmission-based error control and further speed up chunk transfers. On the other hand, it is unclear whether receiver-driven stream adaptation can be applied in CCN. Receivers do not know *where* the Data come from, therefore they may not safely estimate end-to-end throughput. It is questionable whether adaptation schemes, such as those examined in DASH, can be ported in CCN. The enhanced error-control, assisted by CSes in routers, facilitates LVS as well as it may allow receivers to recover some of the lost streaming packets within time. In addition, routers may treat video packets differently in their CSes according to the packets' importance in the live stream (*I* vs. *P* vs. *B* frames), in an attempt to further enhance error-control. What seems problematic in CCN, and applies to both VoD and LVS, is the need to indispensably request each individual Data packet and consequently transmit all the required Interests. Particularly for LVS, it is unclear how the system performs in topologies with asymetric links in which lost Interests in congested paths result in non-transmitted Data packets (and eventually in service degradation) regardless of the down-path condition. We believe that more flexible kinds of requesting Data, such as Batch or Persistent Interests, should be investigated.

PSI supports anycast, thus VoD transfers may be facilitated by letting the network locate the nearest VoD chunk replica. As in CCN, it is unclear whether anycasting is compatible with receiver-driven stream adaptation due to the unreliable end-to-end throughput estimation. In PSI, such worries may be bypassed by letting the network chose a suitable video quality and data path that fits the user's capabilities through PSI's decoupled routing modules (RVS and TMPFS). Embedding such capabilities to the network requires modifying the basic pub/sub primitives. However, it is unclear whether these extensions can be supported without sacrificing the architecture's generality or introducing unnecessary complexity. When it comes to LVS, PSI's centralized route control allows constructing optimal multicast trees, albeit the scalability of this feature, particularly when dynamic groups are concerned, needs to be investigated. What looks problematic in PSI, and this applies to both VoD and LVS, is the delay penalties one may have to pay for resolving requests through PSI's decoupled control plane modules. Subscriptions (and unsubscriptions) must be first resolved by the DHT-based RVS and then the TMPFS must compute the forwarding paths (and establish forwarding state in FNs if required). These two separate steps may impose large delays for flow establishment and ultimately affect the system's responsiveness. We believe that PSI should emphasize on resolving this issue.

VI. Conclusions and Future Work

Information-Centric Networking has been proposed on the ground that the Internet architecture needs a brave renovation in order to meet the high demands of modern applications. Given that video accounts for the majority of Internet traffic, ICN investigation should consider facilitating video delivery as a top priority.

ICN research is still at a seminal level and many of the architectural aspects that are critical to video streaming are still open. In this paper, we showcased that it may be too soon to start implementing streaming applications over the proposed ICN architectures, especially by trying to port existing design rationale in ICN-based video streaming applications. The APIs currently offered by ICN systems resemble the interfaces of existing Internet protocols, however the underlying network operation is still undefined making the behavior of a video streaming application may be unpredictable. In addition, instead of building video streaming applications on top of ICN, we believe that the research community may very well need to investigate changing the architecture itself and incorporate mechanisms that facilitate video delivery. We highlighted the example of embedding semantic information into the content objects (video quality, required throughput, etc) so that the network may actively assist video streaming by means of letting the network locate the suitable version of the requested video that can be better delivered to users and/or select appropriate paths. We have not witnessed such explorations in ICN research, yet we believe that this research direction may prove remarkably fruitful.

Acknowledgment

This research has been co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: THALES.

References

[1] "CISCO Visual Networking Index: Forecast and Methodology, 20122017," 2013. [Online]. Available: http://www.cisco.com

[2] K. Moore, "Things that NATs break," 2004. [Online]. Available: http://web.mit.edu/6.033/2002/wwwdocs/papers/what-nats-break.html

[3] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, "STUN-simple traversal of user datagram protocol (UDP) through network address translators (NATs)," RFC 3489, IETF, Mar, Tech. Rep., 2003.

[4] P. Vixie, "What DNS is not," *Communications of the ACM*, vol. 52, no. 12, pp. 53–47, 2009.

[5] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katsaros, and G. Polyzos, "A Survey of Information-Centric Networking Research," *IEEE Communications Surveys Tutorials*, (DOI 10.1109/SURV.2013.070813.00063; available online since 19 July 2013).

[6] T. Stockhammer, "Dynamic adaptive streaming over HTTP: standards and design principles," in *Proc. of ACM MMSys*, 2011, pp. 133–144.

[7] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. of ACM CoNEXT*, 2009, pp. 1–12.

[8] G. Xylomenos, X. Vasilakos, C. Tsilopoulos, V. Siris, and G. Polyzos, "Caching and mobility support in a publish-subscribe internet architecture," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 52–58, 2012.

[9] K. A. Hua, Y. Cai, and S. Sheu, "Patching: a multicast technique for true video-on-demand services," in *Proc. of ACM Multimedia*, 1998, pp. 191–200.

[10] L. Rizzo, "pgmcc: a TCP-friendly single-rate multicast congestion control scheme," in *Proc. of ACM SIGCOMM*, 2000, pp. 17–28.

[11] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *Proc. of ACM SIGCOMM*, 1996, pp. 117–130.

[12] R. Chiocchetti, D. Rossi, G. Rossini, G. Carofiglio, and D. Perino, "Exploit the known or explore the unknown?: Hamlet-like doubts in ICN," in *Proc. of ACM SIGCOMM ICN workshop*, 2012, pp. 7–12.

[13] W. Chai, D. He, I. Psaras, and G. Pavlou, "Cache Less for More in Information-Centric Networks," in *Proc. of IFIP NETWORKING*, 2012, pp. 27–40.

[14] G. Carofiglio, M. Gallo, and L. Muscariello, "ICP: Design and evaluation of an interest control protocol for content-centric networking," in *Proc. of IEEE INFOCOM NOMEN workshop*, 2012.

[15] CCNx open-source prototype. [Online]. Available: http://www.ccnx.org

[16] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," in *Proc. of ACM MMSys*, 2011, pp. 169–174.

[17] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, P. Stewart, J. D. Thornton, and R. L. Braynard, "VoCCN: voice-over content-centric networks," in *Proc. of ACM ReArch workshop*, 2009, pp. 1–6.

[18] C. Tsilopoulos and G. Xylomenos, "Supporting Diverse Traffic Types in Information Centric networks," in *Proc. of ACM SIGCOMM ICN workshop*, 2011, pp. 13–18.

[19] Y.-K. Wang, M. Hannuksela, S. Pateux, A. Eleftheriadis, and S. Wenger, "System and Transport Interface of SVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1149–1163, 2007.

[20] C. Papadopoulos, "Retransmission-based error control for continuous media applications," in *Proc. of NOSSDAV*, 1996.

[21] D. Byun, B.-J. Lee, and M.-W. Jang, "Adaptive Flow Control via Interest Aggregation in CCN," in *IEEE ICC*, 2013.

[22] P. Jokela, A. Zahemszky, C. Esteve-Rothenberg, S. Arianfar, and P. Nikander, "LIPSIN: line speed publish/subscribe inter-networking," in *Proc. of ACM SIGCOMM*, 2009.

[23] C. Tsilopoulos and G. Xylomenos, "Scaling Bloom filter-based multicast via filter switching," in *IEEE ISCC*, July 2013.

[24] D. Li, Y. Li, J. Wu, S. Su, and J. Yu, "ESM: Efficient and scalable data center multicast routing," *IEEE/ACM Transactions on Networking*, vol. 20, no. 3, pp. 944–955, 2012.

[25] C. Stais, A. Voulimeneas, and G. Xylomenos, "Towards an Error Control Scheme for a Publish/Subscribe Network," in *Proc. of IEEE ICC*, 2013.

[26] Y. Thomas, C. Tsilopoulos, G. Xylomenos, and G. C. Polyzos, "Multisource and Multipath File Transfers through Publish-Subscribe Internetworking," in *ACM SIGCOMM ICN workshop*, 2013.

[27] C. Tsilopoulos, I. Gasparis, G. Xylomenos, and G. Polyzos, "Efficient real-time information delivery in future internet publish-subscribe networks," in *Proc. of ICNC*, 2013, pp. 856–860.

[28] H. Holbrook and B. Cain, "Source-specific multicast for IP," RFC 4607, August, Tech. Rep., 2006.