Supporting Diverse Traffic Types in Information Centric Networks

Christos Tsilopoulos, George Xylomenos
Mobile Multimedia Laboratory, Department of Informatics
Athens University of Economics and Business, Athens, Greece
Email: {tsilochr, xgeorge}@aueb.gr

***Abstract*—In this paper we focus on the issue of transferring diverse kinds of information through *information-centric networks* (ICNs). We argue that the *one request per packet* mode of operation suggested in the early development of ICN applications is not a good fit for some types of traffic, such as media streams and real-time notifications. To efficiently deliver all kinds of information, we argue that an ICN should not only identify information by its name, it should also be aware of the nature of its traffic. We classify information traffic types based on two characteristics: a) reliable vs. unreliable transfer and b) real-time vs. on-demand delivery. The combination of these two characteristics leads to three broad categories: a) *channels*, b) *on-demand documents* and c) *real-time documents*. To handle all traffic types, we propose two extensions to the CCN architecture: *Persistent Interests* and *Reliable Notifications*. We describe how these additions, together with a careful selection of information names, can efficiently support these three categories of information traffic types.**

***Index Terms*—Information-centric networks, network layer design, information traffic types.**

I. Introduction

Information-centric networks (ICNs) have gained the attention of the research community as a new paradigm in networking that can better address user needs in a networked world. Proponents of ICNs argue that placing information at the heart of an internetworking architecture allows the network to apply a set of mechanisms and algorithms to increase the users' perceived satisfaction in terms of application performance and secure access to information [1], [2], [3], [4], [5], [6]

From a technical viewpoint, the core abstraction in an ICN is *named data* instead of the *named hosts* of the Internet. Instead of identifying the source and destination host, the header of a network layer packet contains an identifier for the data carried. The primitive actions at the network layer follow a *receiver-driven* approach: to receive a packet of named data, a user must first request it by its name. This is in stark contrast to IP where a user may freely send packets to recipient hosts. An ICN's responsibility is to route requests for information to the *best* available location holding the desired data (*best* being subject to various metrics, such as hop count, latency or security) and then deliver the data back to the requestor.

Although the core abstraction at the internetwork layer - requesting and receiving named data - is radically different than that of the TCP/IP based Internet, ICNs do not necessarily violate the design choices that led to the success of the Internet. We believe that the design of an ICN should embrace the End-to-End argument [7] by keeping the network core as simple as possible and pushing functionalities at the edges. At the network layer, packets are statistically multiplexed and are susceptible to loss either due to failures in the link layer or due to congestion. Hence, the network layer of an ICN should provide users with an *unreliable, best effort, information delivery service*.

Early work on transport over ICNs [8], [9] showcased how media streams, specifically voice conversations, can be supported in an ICN. Both papers apply the same logic: a voice conversation between two points is decomposed into two unidirectional streams of named data. For each stream, the communication end points (sender and receiver) use the same algorithmic function to generate names for each data packet in the stream. The receiving side issues a series of requests, one per named packet.

Based on these applications, we argue that sending a request for each packet in a media stream may lead to inefficiencies as a) bandwidth is wasted for control messages, b) network elements in the core are overloaded by the large number of requests and c) if a request is lost and fails to reach the data source, the corresponding data packet will not be forwarded to the receiver.

Points (a) and (b) could be tackled via *batch requests*: a single request carrying the names of many data packets. In this manner the number of requests issued by the receiver can be drastically reduced, saving bandwidth and unloading intermediate network elements. However, batch requests do not address point (c). If the uplink path suffers from relatively high packet error rates (perhaps due to congestion), requests will be lost and thus data packets will not be forwarded to the receiver. The situation is even worse if batch requests are lost; not only one, but many data packets will not be forwarded. In essence, if information is to be requested at a packet granularity, then the conditions of the uplink path will affect the perceived quality of service, regardless of the conditions in the downlink path.

We believe that these problems reflect a wider issue for ICNs: a mismatch in the traffic nature of *continuous media* and the *one request per packet* mode of operation. Although a user should receive information **only** if she has requested it, the efficient dissemination of continuous media requires more flexible mechanisms than separately requesting each data packet.

In this paper we argue that the efficient dissemination of

information requires different routing and forwarding mechanisms depending on the underlying traffic type. Looking back at the TCP/IP protocol stack, concerns regarding information transfer are left to the transport and application layer. However, in a network built around information, these concerns have to be addressed at the network layer. Initial ICN architectures [3], [4] propose to route and forward packets based solely on the requested information name. We argue that an ICN should not only recognize information by its name, but also by its traffic nature. Specifically, we propose that information dissemination should be classified by two characteristics: a) reliable vs. unreliable transfer b) real-time vs. on-demand delivery. The combination of these characteristics leads to three broad traffic types: a) *channels*, b) *on-demand documents* and c) *real-time documents*. We then focus on CCN [4], for which we propose two extensions to its routing and forwarding scheme. Our concepts are however also applicable to the PSIRP/PURSUIT architecture [6] We discuss how these extensions along with careful name selection can efficiently support the dissemination of the three identified traffic types.

II. Disseminating information in the Internet

To identify the various forms of information dissemination, we start by examining how information is disseminated in the Internet today. If we take a bottom-up look through the TCP/IP protocol stack, we see that the characteristics of information dissemination are not a concern until we reach the application layer. At the network layer, IP provides an unreliable, best effort packet delivery service. Although the IP header includes some type of service fields, in practice IP is unaware of what sort of information is carried in a packet. Transport protocols, namely UDP and TCP, are also oblivious to this information: even though they are used to provide end-to-end transport, their operation remains the same regardless of the kind of information carried. For example, the TCP implementation in a host applies the same flow and congestion control, whether the data carried represent a web page, a twitter update or a fragment of a voice conversation. Likewise, UDP operation at the end hosts is not faster or less prone to packet loss if it is used to carry streaming video rather than data for online games.

Internet applications and application layer protocols however are designed based on the kind of information they are meant to disseminate. At a first glance, most Internet applications care about reliable transfer (FTP, HTTP, SSH, SMTP, P2P to name a few). The majority of them transfer data over TCP, which provides error control and in order delivery. There are a few cases where applications avoid TCP due to the delays of TCP's flow and congestion control. Such applications resort to data transfer over UDP, applying error control and packet ordering themselves. On the other hand, delay sensitive, real-time media are tolerant to packet loss and thus do not require reliable transfer. For streaming media applications, UDP is again the primary choice. If an application requires a reliable channel for control signaling, e.g. RTCP, then an out-of-band TCP connection is usually established. Therefore, one classification of information dissemination is whether it requires reliable or unreliable transfer.

Information dissemination can be further classified by whether transfers are made in real time or on demand. By *real time* we denote information that is instantly transmitted by data sources at the moment it is generated. Users receiving real-time information are implicitly synchronized in the sense that they receive the same information simultaneously, regardless of the point in time they expressed interest for it. Continuous media applications like live TV and web radio are some examples of real-time traffic. In terms of IP, the optimal solution for delivering real-time information is to forward data over IP multicast. However, disseminating real-time information is not limited to live media streaming. Real-time information includes applications such as online gaming, twitter, chat rooms, emergency alerts, sensor network measurements etc. These are all applications where information has to be *reliably and simultaneously* delivered to a set of synchronized users. Ideally, these applications should be implemented on top of a reliable multicast transport protocol (e.g. [10], [11], [12], [13]). However, neither IP multicast nor reliable multicast transport protocols ever achieved wide deployment. In practice, reliable transport of real-time information is either implemented by multiple unicast connections (each user directly connected with the data source) or via an overlay multicast scheme.

Recently, HTTP was proposed as the thin layer in an ICN [14]. HTTP was selected due to its content-centric nature and its compatibility with firewalls and NAT boxes. HTTP transfers data over TCP, therefore HTTP transfers are reliable. Authors in [14], recognizing the inability of HTTP to support real-time information, introduce a new HTTP method, the *Subscribe-GET* (S-GET). When an HTTP client sends S-GET requests to a web server, the underlying TCP connection between the server and client is kept alive, unlike in the standard protocol where requests are served and then the connection is closed. When new information is published to the web server via HTTP PUT messages, the server immediately forwards it to all connected clients, thus delivering information in real time (Figure 1).

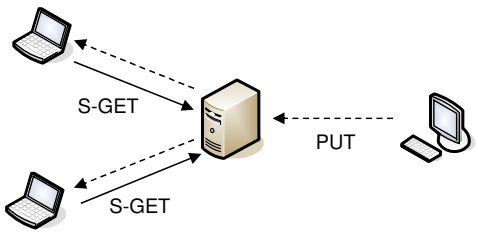

Fig. 1. HTTP clients send S-GET requests (solid arrows). TCP connections with the web server are kept active. New information is delivered to clients in real time (dashed arrows.

Our thinking follows the same rationale. A user's request for information should specify both the information name and the dissemination pattern. In terms of HTTP, this is denoted by the resource name and the selected HTTP method, GET or S-GET. However, HTTP's reliance on TCP makes it inappropriate for disseminating live media content. We propose instead a general purpose, packet switched, ICN architecture that can efficiently

deliver all kinds of traffic, including the loss tolerant but delay sensitive media streams.

III. Information dissemination types

In this section we present a more formal categorization of information dissemination types. The axes of classification as described in Section II are a) the requirement for reliable transfer and b) the requirement for real-time delivery.

A. Documents and Channels

We consider pieces of information that must be reliably transferred as belonging to **documents**. To obtain a document over an *unreliable information delivery service*, the user at the edge of the network must perform some kind of error control, e.g. a retransmission scheme as in TCP. This can be achieved by dividing a large document to named packets and then requesting each packet by name. If a packet is lost, the receiver will re-request the packet, in contrast to the sender-driven approach of TCP where retransmission occurs upon the sender's initiative.

We consider instead loss tolerant pieces of information as belonging to **channels**. As discussed in Section I, requesting each packet in a streaming channel is inefficient. To receive channel information, a user should *subscribe* to the channel once and then the network should forward each network packet belonging to the channel, until the user's interest seizes to exist.

B. Real-time and On-demand Dissemination

The second axis of classification regards the timing constraints of information dissemination. We consider as real-time any information transmitted to users at the moment it is generated, i.e. on the sender's initiative. Real-time information includes both continuous media (live TV, web radio) and real-time notifications (chat rooms, twitter updates, emergency alerts, etc). The fundamental difference between continuous media and real-time notifications is that continuous media are tolerant to packet losses (*channels*) while real-time notifications require reliable transfer (*documents*). When receiving real-time information, receivers are implicitly synchronized; they receive the same data at the same time, regardless of the when they expressed their interest in it. As discussed in Section II, disseminating information in real time applies to both channels (live TV, web radio) and documents (chat rooms, twitter updates, emergency alerts, etc).

On-demand information dissemination on the other hand includes transferring archived data (e.g. files) and point-to-point conversations (e.g. transactions, personalized content). Users receiving information on demand cannot be implicitly synchronized by the network. For example, if two users request the same file from a file server at different times, then at a certain point in time they are receiving different network layer packets, even if the two transfers are interleaved in time. To increase performance, an application could explicitly synchronize the receivers or implement an asynchronous multicast scheme through caching [15].

TABLE I
Applications classified according to the traffic type

	Channels	**Documents**
Real-time	Live TV, Web radio	Twitter updates, online gaming chat rooms, emergency alerts
On demand	VoIP, Skype	File download, email, YouTube

C. The Three Traffic Types

Based on the above, we propose that information dissemination should be classified as a) *channels*, b) *on-demand documents* and c) *real-time documents*. Table I presents a sample of applications classified according to their traffic type. Note that channels constitute a *single* category. This is because information represented as a channel requires the same routing and forwarding schemes, regardless of whether its dissemination is real-time (live media streaming) or on-demand (unreliable transmission of archived data). Details for the routing and forwarding channels follow in Section IV-C. Also note that YouTube is classified as an on-demand document application. YouTube videos are archived data stored in the service' servers, transmitted reliably over TCP on-demand (a transfer starts when a user explicitly requests a video). Therefore, YouTube videos are classified as on-demand documents.

IV. Extensions to CCN

In this section we turn our focus on CCN [4]. We discuss how CCN's basic model can efficiently transfer on-demand documents but faces problems when it comes to channels and real-time documents. We then propose two extensions to CCN to overcome these issues.

A. CCN Overview

Content Centric Networking (CCN) is an ICN architecture proposed by Van Jacobson et al. [4] that places data at the thin waist of the network stack. In CCN, data names have a hierarchical structure, similar to file system pathnames, e.g. *"/christos/pictures/summer.jpg"*. CCN users request named data packets by issuing *Interest* packets. Interests are forwarded by CCN routers in a hop-by-hop manner. Upon receiving an Interest, a router first looks in its local cache and if a copy of the requested data packet is found, it instantly sends it back. Otherwise the router performs a *longest prefix match* on its *Forwarding Information Base* (FIB) and forwards the interest to the next hop towards the data source (Figure 2a). Routers keep track of each forwarded Interest in a data structure called *Pending Interest Table* (PIT), as shown in Table II.

When the Interest reaches a data source, the requested data packet is forwarded along the reverse path. At each hop, routers check their PIT for Interests whose name is an *exact* match of the data name. If a match is found, the data packet is forwarded and a copy of the packet is kept in a local cache for future use (Figure 2b).

Incoming data packets that do not have a match at the PIT are considered as unwanted traffic and are discarded. After a

data packet is forwarded, the router assumes that the Interest is *satisfied* and deletes its entry from the PIT. This way CCN ensures that a user receives at most one data packet per issued Interest. As we will describe in Sections IV-C and IV-C, we need to relax this constraint in order to efficiently support channels and real-time documents.

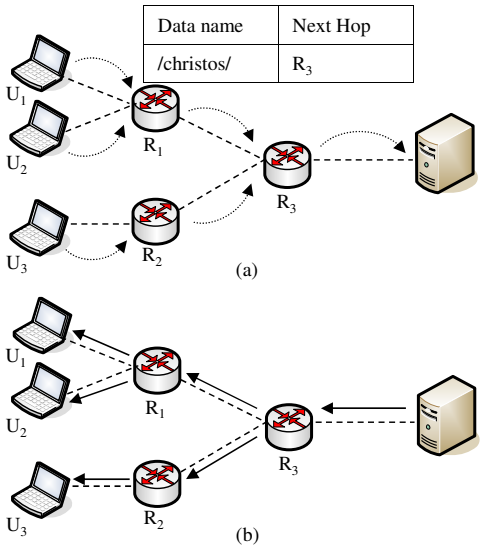

Fig. 2. CCN users send interest messages for "/christos/pictures/summer.jpg". (a) CCN routers propagate the interest towards the data store. (b) Data packets are forwarded following the reverse path.

TABLE II
PENDING INTEREST TABLE AT ROUTER R_3

Interest name	Forward to
"/christos/pictures/summer.jpg"	R_1, R_2

B. On-demand Documents

Reliable end-to-end data transfer over an unreliable network requires some form of error control. Usually this takes the form of either an *Automatic Repeat Request* (ARQ) or a *Forward Error Correction* (FEC) scheme implemented by the application. CCN's basic model of *one interest per data packet* can easily support both schemes, provided that they are implemented in a receiver-driven fashion. For example, consider a pull-based variant of *Stop and Wait* ARQ. Let R be a receiver that wishes to download document O. If O is too large to fit in a single network layer packet, O is split in n packets, each one with its own name, $O_1 ... n$. R initiates the transfer by requesting O_1 and then waits until the network delivers the corresponding data packet or a timer expires. If the timer expires, R re-requests O_1. Once the data packet for O_1 arrives, R proceeds to O_2 and resets the timer. The operation continues until R receives all packets comprising O. Note that R does not directly address the host to where the requests are sent and the network may route each request to a different location. To increase performance, R may implement *Selective Repeat* ARQ by pipelining requests. The scheme can be further extended to support some kind of flow and congestion control (*a la* TCP) by controlling the rate of sending the requests.

C. Channels

VoCCN [8] applied CCN's *one interest per data packet* scheme to the transfer of real-time voice streams. To minimize the end-to-end delays caused by this step-wise request-response process, at the beginning each receiver issues a number of pipelined Interests. Each Interest is routed to the channel source and remains there in a pending state. When a new data packet is generated, it is immediately forwarded to the receiver, consuming the correspondent Interest in intermediate CCN routers. Whenever a data packet is received, the end point issues a new Interest to replenish the consumed one.

As argued in Section I, explicitly requesting each packet in a media steam is inefficient. Sending an Interest for each data packet in a stream wastes uplink bandwidth and burdens routers with a large number of PIT entries. Furthermore, if an Interest is lost, the corresponding data packet will not be forwarded, therefore reducing the perceived quality of service.

To overcome these issues, we propose an extension to CCN routing and forwarding: *Persistent Interests* (PIs). In contrast to plain Interests, CCN routers store PIs in their PIT for a period of time. PIs are not deleted after a matching data packet is forwarded; instead, they remain in the PIT until users explicitly unsubscribe from a channel or their lifetime expires. Users issue PIs periodically so that state in routers is refreshed. PIs that have not been refreshed for a while are discarded as stale.

In the data plane, each data packet in a channel still has its own name, in order to distinguish data packets, but they all share a common prefix, the channel name. For example, if a channel is named *"SportsTV"*, its data packets could be called *"SportsTV/Packet1"*, *"SportsTV/Packet2"* and so on. Channel data packets are specially marked so that *forwarding is performed based only on the channel name and not on the packet's full name*. When CCN routers receive data packets belonging to channels, they extract the channel name from the packet name and search their PIT for a matching PI. Once the match is found, the data packet is forwarded and the PI is kept in place.

The dissemination of real-time channel information is fairly easy. An application registers a name for the channel and advertises it to users. In practice, the name must be carefully selected so that CCN routers will propagate the PI to the right content provider. If multiple users send PIs for the same channel, CCN can implicitly group all users into a single multicast tree and forward the same data packets simultaneously to the subscribed users.

Apart from real-time channels, PIs can be used for disseminating on-demand channels as well. An example of an on-demand channel would be a content provider that unreliably streams prerecorded media on user demand. In this case, the application must be careful in naming these channels; otherwise users will receive invalid data. For example, user U_1 in Figure 2 may ask for last night's soccer game by sending a PI for "Soccer match, April 2nd". A few moments later,

user U_3 may also ask for last night's soccer game, sending a PI for the same channel name. When the PI sent from U_3 reaches R_3, R_3 will match it with the previous PI sent by U1 and will not push it to the data source. In addition to that, R_3 will aggregate the PI into its existing PIT entry and forward upcoming data packets towards both users. As a result, U_3 will have just missed the first minutes of the soccer game.

To avoid such confusions, disseminating on-demand channels can be achieved by creating new channel names on demand, e.g., *"*Soccer match, April 2nd,*ordered by Bob"* and *"*Soccer match, April 2nd,*ordered by Alice"*. Users willing to receive information represented as on-demand channels must first negotiate with the content provider a customized channel name, exchange the name through an out of band mechanism and then subscribe to their custom named channel. In this manner, different channels will be created and data packets will not be correlated by CCN routers. Of course, the provider can group multiple such requests together so as to offer a near video on-demand service via multicast.

D. Real-time Documents

The last traffic type is real-time documents. In this category, users want to reliably receive information generated in real time. Real-time documents can be viewed as a synthesis of (real-time) channels and error control applied at the end hosts. The ACK implosion problem [10], [11], [12], [13] faced by many reliable multicast protocols is not an issue in CCN, since lost packets can be retrieved from caches in intermediate routers. In this sense, ICNs provide an ideal communication substrate for reliable multicast transport.

The problem with real-time documents is how to notify the receivers that new information is available, so that they may request it. We can illustrate this with an example. Consider a fire alert application where a fire detector signals alarms over the network. Fire alerts must be reliably delivered to interested users, e.g. the fire service, the police, the local hospital etc. Assume that the fire alert is represented by only a few bytes and fits into a single network packet. In case of fire, the sensor creates a single data packet and pushes it to the network. If the fire alert is lost in a congested link, there is no way for the receivers behind that link to identify the loss and re-request the data packet in time (Figure 3). Receivers could bypass this by issuing periodic requests, but this solution suffers from two weaknesses: a) repeated requests will cause extra network overhead and b) if information is generated right after a periodic request and the packet is lost, it will only be recovered after the next periodic request.

Looking at how the Internet delivers real-time information, brings the sender back to the center of attention. Real-time information is actually sender-driven: the data source sends packets and awaits receivers to acknowledge the proper reception of data; otherwise the sender takes the initiative to retransmit unacknowledged packets. How could this be emulated in a receiver-driven way? In CCN, data sources are unaware of receivers and cannot take the initiative of retransmitting packets.

To solve this problem, we propose the use of *Reliable Notifications* (RNs). RNs are special data packets sent by

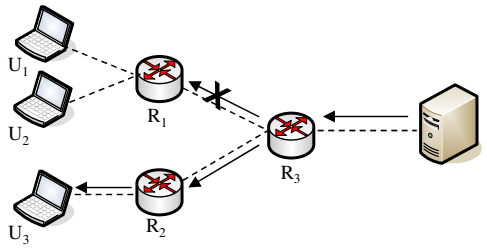

Fig. 3. Real-time information fits in a single data packet. If the packet is lost in the link R_1-R_3, receivers U_1 and U_2 cannot identify the loss.

data producers to notify receivers that real-time information is available. RNs are hop-by-hop reliably propagated to receivers. When a router receives an RN, it immediately sends an acknowledgement to the previous hop. RNs that are not acknowledged in time are retransmitted. Once a receiver gets an RN, it waits for new data to arrive. If no packets arrive, or individual packets are lost, the receiver issues interests for the missing data.

To receive documents in real time, users send Persistent Interests, just as in channels. When new information is generated, the data source first creates a reliable notification. The notification is named after the channel, carrying in its payload the pair $< N, Name_1 >$ where N is the number of packets comprising the document and $Name_1$ is the name of the first data packet. The notification is forwarded as a streaming packet, with the addition of hop-by-hop acknowledgments. Once a user receives a notification, she is aware that N packets are to follow, with the first packet named $Name_1$.

After transmitting the notification, the source transmits the data packets. As in channels, data packets are marked so that they are forwarded based on their channel name prefix. If the first or any subsequent data packet is lost, receivers request it explicitly via a plain Interest, using the packet's full name. The Interest is routed towards the data source and is served either by a router's local cache or the data source itself.

V. Conclusions

Early development on information-centric networks suggests that end-to-end information transfer is performed on a *request per packet* basis. However, this mode of operation does not fit all kinds of traffic. In the case of streaming, requesting information on the granularity of packets is inefficient as it wastes network resources and may reduce perceived quality of service. The efficient dissemination of information requires different routing and forwarding mechanisms. ICNs should address these issues in the design of their network layer rather than leaving them to upper layers. Efficient end-to-end transport requires that users request information not only by its name, but also by specifying the nature of the underlying traffic. We explored this concept in CCN and proposed two extensions to CCN's routing and forwarding for disseminating information represented as channels and real-time documents. We believe that similar extensions would be valid and useful of other ICN architectures, including the publish/subscribe architecture advocated by PSIRP/PURSUIT.

ACKNOWLEDGMENTS

The work reported in this paper was supported by the FP7 ICT project "Publish Subscribe Internet Technology" (PURSUIT), under contract ICT-2010-257217. The authors would like to thank Pantelis Frangoudis for his useful comments and remarks.

REFERENCES

[1] M. Gritter and D. R. Cheriton. An architecture for content routing support in the Internet. In *USENIX USITS*, 2001.
[2] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica and M. Walfish. A layered naming architecture for the Internet. In *ACM SIGCOMM*, 2004.
[3] T. Koponen, M. Chawla, B. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker and I. Stoica. A data-oriented (and beyond) network architecture. In *ACM SIGCOMM*, 2007.
[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking Named Content. In *ACM CoNEXT*, 2009.
[5] D. Trossen, M. Sarela and K. Sollins. Arguments for an information-centric internetworking architecture. In *ACM Comput. Commun. Rev.*, 2010.
[6] N. Fotiou, D. Trossen and G.C. Polyzos. Illustrating a publish-subscribe Internet architecture, In *Springer Telecommun. Syst.*, 2011.
[7] J. H. Saltzer, D. P. Reed and D. D. Clark. End-to-end arguments in system design. In *ACM Trans. Comput. Syst.*, 1984.
[8] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, P. Stewart, J. D. Thornton and Rebecca L. Braynard. VoCCN: voice-over content-centric networks. In *ACM ReArch*, 2009.
[9] C. Stais, D. Diamantis, C. Aretha and G. Xylomenos. VoPSI: Voice over a Publish-Subscribe Internetwork. In *Future Networks and Mobile Summit*, 2011.
[10] J.C. Lin, S. Paul. RMTP: a reliable multicast transport protocol. In *IEEE INFOCOM*, 1996.
[11] H. W. Holbrook and D. R. Cheriton. IP multicast channels: EXPRESS support for large-scale single-source applications. In *ACM Comput. Commun. Rev.*, 1999.
[12] L. Rizzo. PGMCC: a TCP-friendly single-rate multicast congestion control scheme. In *ACM SIGCOMM*, 2000.
[13] J. Gemmell, T. Montgomery, T. Speakman and J. Crowcroft. The PGM reliable multicast protocol. In *IEEE Network*, 2003.
[14] L. Popa, A. Ghodsi and I. Stoica. HTTP as the narrow waist of the future Internet. In *ACM Hotnets*, 2010.
[15] K. Katsaros, G. Xylomenos and G. C. Polyzos. MultiCache: An overlay architecture for information-centric networking, In *Elsevier Comput. Netw.,* 2010.