Exploiting OpenFlow resources towards a Content-Centric LAN

Kostas Choumas[◇], Nikos Makris[◇], Thanasis Korakis[◇], Leandros Tassiulas[◇] and Max Ott[†]
[◇]Department of Computer & Communication Engineering, University of Thessaly, Volos Greece
[†]National Informations and Communications Technology Australia (NICTA), Sydney Australia
e-mail: {kohoumas, nimakris, korakis, leandros}@uth.gr | Max.Ott@nicta.com.au

***Abstract*—The efficient utilization of emerging new technologies for the sake of implementing attracting and novel network architectures is a major research challenge. As traditional addressing schemes seem to be rather inefficient to cope with emerging Internet technologies, research concerning Content Centric Networks (CCNs) has received a lot of attention by the research community. CCNs are designed to treat content as a primitive and therefore overcome the obstacles posed by traditional addressing schemes. Utilizing Software Defined Networking (SDN) approaches can lead to a realistic implementation of CCN scenarios. In this paper, we exploit the OpenFlow (OF) technology, an SDN enabler, in order to efficiently create and manage CCNs which are backward compatible with already existing networking infrastructure. We evaluate our scheme by using different load balancing policies, based on the actual network state at every observation interval. The process is completelly transparent to the end user, making our approach an easy to integrate solution for all existing networking topologies. Our solution is implemented and evaluated under a real life scenario, utilizing distributed testbed resources, consisting of the NITOS wireless testbed and PlanetLab-Europe (PLE).**

I. Introduction

The motivation behind this work is inspired from the novel technology of the Software Defined Networking (SDN) [1] and the emerging idea of Content Centric Networking (CCN) [2]. SDN is a constantly growing networking approach that decouples network control (learning and forwarding decisions) from network topology (junctions, interfaces, and the way they peer). SDN equipment consists of special switches dedicated to the data forwarding plane, while the network control plane is assigned to decoupled from the physical device controllers. Data plane refers to the part of the routing process that includes the forwarding decisions of the packets arriving on the inbound interfaces, while control plane concerns to drawing the network map. OpenFlow (OF) [3], [4] is the most widely used enabler of SDN, with many networking equipment vendors supporting its development. As a proof of concept, Google's Intranet has recently been completely redesigned to run under OF. OF flexibility provides ease of design and deployment of novel network architectures, even those inspired by the CCN notion.

Content centric networks allow a user to focus only on the requested content (data or services), rather than referring to a specific host where that content will be retrieved from. A widely used example of CCN is the Content Delivery Network (CDN), which exploits the DNS protocol in order to map each content request to one of the available servers, rendering the user totally unaware of the actual content providing server. In fact, the network serves content to users with high availability and high performance, due to the existence of an underlying large distributed system of servers deployed in multiple different sites in the network. The mapping of each user content request to the most appropriate server is done in a transparent way from the user perspective, based on a variety of criteria including the server availability, process load and/or proximity.

In this work, we focus on the redesign of the underlying network plane by exploiting OF resources, towards creating a Content-Centric LAN (CCLAN), an easy to adopt solution for content based delivery. The extended use of Virtual Private Networks (VPNs), able to define LANs with expanded coverage area all over the world renders them an excellent example of a use case, where our proposed platform can be easily integrated. VPNs form LANs with an abstract sense of proximity, while the implicit geographical distances require the deployment of multiple servers for the same content and the application of various and diverse load balancing policies. Inspired by the CDN approach, we introduce a mechanism of content mapping based on the MAC layer and performed with the OF switches, rather than using the less interactive application-layer DNS based one.

The implementation and experimentation on the proposed CCLAN is based on the federated environment of the NITOS testbed [5] and PLE [6] resources. A CCLAN has been deployed among a Europe-wide distributed system of PLE computers and an OF enabled subnetwork of NITOS nodes, featuring a TAP based VPN interconnection (operates with MAC layer packets) among both sets of resources. As a proof of concept, we evaluate our scheme under different load balancing policies, that affect the selection of the most appropriate content providing server. The experimentation on a realistic setup with real measurements regarding the performance and efficiency of the evaluated load balancing policies is a significant effort towards the reliable comparison among them.

The rest of the paper is organized as follows. In Section II we introduce related work. Section III describes our proposed scheme and Section IV presents the implementation features in detail. Our evaluation of the overall solution is presented in Section V, while Section VI concludes the paper.

II. Related Work

CCN is recently gaining more and more respect from the research community. Since its introduction [2], a major thrust is evolving towards establishing a content centric Internet, as it is prominent from the existence of many content oriented research projects [7], [8], [9]. Many different aspects of implementation have been proposed up to now, concerning the CCN support by the current Internet infrastructure. However, most of them require a costly redesign of the Internet backbone or a complex setup of DNS servers in order to support content based routing schemes.

More specifically, authors in [2] propose the adoption of CCN based routers attached to current Internet backbone, without interfering with the existing working setup. However, these routers will be able to support only the link-state and not the distance-vector routing protocols, while the authors in [15] argue about the infeasibility of an Internet scale setup of CCN based on current router hardware. In [10], authors suggest either a novel protocol replacing IP or extending it, introducing a new header field containing the content identifier. This work has been further extended for the adoption of a new transport layer protocol in [11]. However, the common feature of the aforementioned architectures is the requirement for an extended Internet redesign, making them inapplicable for the time being.

On the other hand, CCN has come to life using the approach of CDNs, without requiring complex low layer modifications and relying on upper layer operations. In particular, CDNs rely on a DNS based deployment [14] or transport/application layer switches [12] to perform load balancing of content requests over the Internet. Nevertheless, the inherent architecture of CDNs renders any per packet load balancing policy inapplicable. Moreover, existing services that rely on IP rather than URL to retrieve content cannot ensure "off-the-self" operation with CDNs.

In order to deal with all aforementioned issues, we propose an innovative scheme that uses IPv4/6 address to characterize content. To implement our scheme, we involve OF in CCN. OF has attracted attention from both the research community and networking equipment vendors, since it enables experimentation with novel protocols using the networks we use everyday. OF is highly customizable enabling packet forwarding based on a variety of criteria except from the identification of the MAC destination. In our approach, routing of contents can be performed by using the unmodified TCP/IP Internet protocol stack and the widely available OF network equipment. Finally, we exploit the OF flexibility in order to apply different load balancing policies on the content requests.

III. Scheme Design

A. LAN preliminaries

Current network architecture is based on the TCP/IP protocol stack, where the packet process in a computer or a network device involves a separate subprocess in every layer. The content is identified by a known and human readable URL, which is translated through the application layer DNS to the network address (IPv4/6) of the corresponding server. Then, in order to map the network address to a specific physical address (MAC), the Address Resolution Protocol (ARP) is involved. The ARP operation is demonstrated in Figure 1(a). When a client wants to retrieve the MAC address of a server belonging to the same LAN, it broadcasts an ARP Request including the known server IP address. Then, the server listens this request, recognizes the including IP address as its own one and responds with an ARP Reply that includes its own MAC address. After this packet exchange, the client has all necessary knowledge to initiate the content request. Simultaneously, the network switches that establish the communication links are appropriately updated to forward the content request and reply.

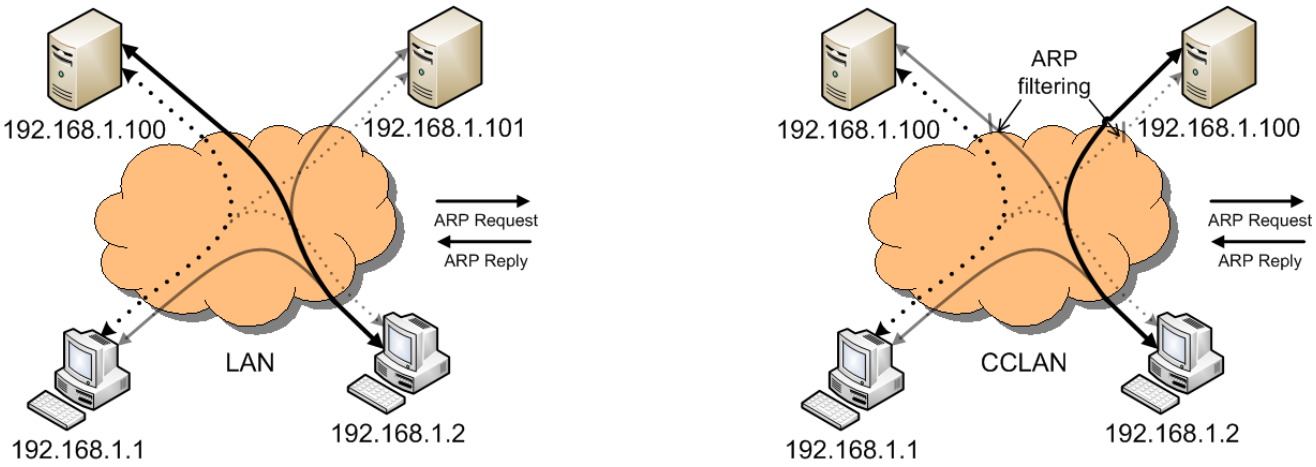
B. CCLAN presentation

The proposed scheme introduces a different operation of a LAN, using appropriately configured OF switches instead of Ethernet switches. The key novelty of our scheme is a differentiated manipulation of the ARP messaging process. We efficiently utilize the inherent intelligence of the OF switches towards controlling and filtering of the ARP process.

In particular, the existing protocol stack remains unmodified, in order to ensure both backward compatibility and less complex overall setup. Content identifying URLs are mapped via the DNS protocol to specific IP addresses, that are able to characterize content as well. Since we are talking about a distributed setup of servers, that all offer the same service, an IP address is no more a unique identifier for a host machine, but is used to denote the services offered by that specific host (or group of hosts). Therefore, a host machine may feature more than one IP address; This is feasible using multiple virtual interfaces. Moreover, the same IP address or content id, shall be used by many servers that provision the same service. Dynamic advertisement of available services from the content servers has been further discussed in [16], [17].

Apparently, when using the proposed setup in a LAN, an ARP Request shall trigger many ARP Replies, concluding in a conflict at the end user that has initiated the ARP process. However, the OF switches can upon request select which ARP Replies will be forwarded, thus imposing implicitly many load balancing techniques among the available content servers, as it is depicted in Figure 1(b).

In summary, the proposed process is described as follows; when a CCLAN end-point requests a content, that has been previously matched to an IP address from the corresponding DNS service, it issues an ARP Request, in search of the requested content server MAC address. The OF switches that intercept traffic in the LAN, will detect this ARP Request from the initiating host and subsequently forward this packet to the OF controller. The controller will examine the ARP packet, and let it pass through the network, to all the destined content servers with that matching IP address. The servers upon the reception of the ARP Request, will issue back an ARP Reply, reporting their MAC address to the host that initiated the communication. However, as the ARP Replies pass through



(a) usual LAN

(b) Content-Centric LAN

Fig. 1. ARP messaging

the OF switches, they are also sent to the OF controller, which will enable the forwarding of the most appropriate ARP Reply, indicating one of the available content servers based on the applied load balancing policy. In order to ensure no performance degradation over existing solutions, the OF switches are programmed to forward the first ARP Reply they receive, the time that our system is initiated, and subsequently impose our policies for the next ARP replies that arrive at the switch.

C. Load balancing policies

As it is already mentioned, the proposed scheme has been evaluated by running several experiments when imposing different policies that balance the client requests.

The first examined balancing policy adopts a per client-request scheme (Figure 2(a)), that maps every new client request to a fixed content server. This **Client-based policy** chooses to forward the ARP Reply of the least loaded server to any new client that initiates an ARP Request. If the client is not new, then the same server always replies to this user. Since we just use these policies only for evaluating our overall scheme, we expect that this shallow approach will be able to perform efficiently even in a system that the clients feature different bandwidth requirements.

The second evaluated policy (Figure 2(b)) is examining the overall load on the flows upon a switch, and tries to load balance traffic based on this criterion. Stemming from the statistics that every OF switch holds, the OF controller periodically checks the statistics of the amount of data sent through the existing flows and estimates the total load of traffic which is handled from each server. This **Load-based policy** tries to dispense traffic among the available content servers. Whenever it detects an overloaded server, it switches the most demanding content request to another less congested server. This approach manages to efficiently distribute traffic among all the available servers.

The last policy we decided to study, is a **Proximity-based** one (Figure 2(c)), that assigns a client to the server that most quickly responded, focusing on a first-come, first-served technique of forwarding the server ARP Replies. This balancing policy may only be efficient enough in the case of low network traffic. For the shake of this approach, we assume that no additional delays due to network traffic can occur during the ARP process.

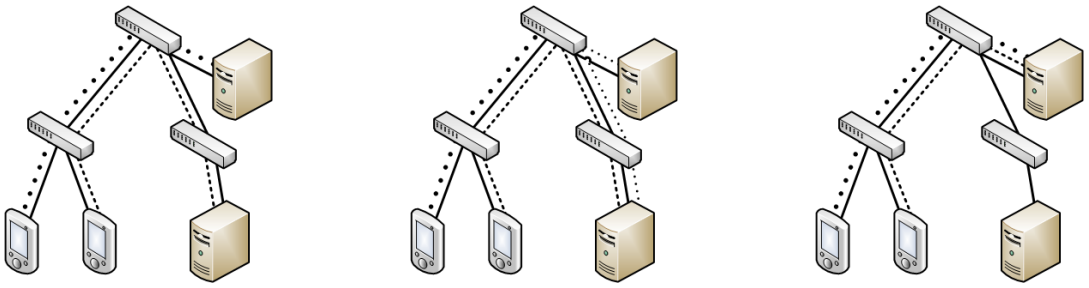
IV. Implementation Details

A. Experimentation platform

For the implementation of the proposed scheme of CCLAN, a variety of software tools is used for setting up the infrastructure and managing the experimentation on top of this. More particularly, an extended LAN spanning multiple European countries is built, using VPN connections between the participating resources, as it is illustrated in Figure 3.

Concerning the hardware engaged in our experimentation, we used two widely used European testbeds; The NITOS wireless testbed in Greece, and the PlanetLab Europe, with resources distributed all over Europe. NITOS is an outdoor deployed, large-scale wireless testbed, currently consisting of 50 operational WiFi nodes in the premises of a University of Thessaly campus building, interconnected through two Pronto 3290 OF switches. PlanetLab-Europe (PLE) is the European portion of the publicly available PlanetLab testbed, a global facility, offering a total of 1000+ nodes worldwide. Each node is a dedicated server, that runs components of PlanetLab services, a setup that is very useful for our experimentation. Both PLE nodes and NITOS server have access to the Internet with a public IP address.

The NITOS server and the NITOS nodes are straightly connected through two OF switches. Half of the NITOS's nodes are connected to one OF switch, while the rest are connected to the second OF switch. The NITOS server is connected only to one of the two switches, which are properly interconnected through a single link. Using a private VPN network, based on the OpenVPN application, we managed to interconnect via Ethernet-bridging all the PLE resources committed to our experiments, by placing them under one TAP



(a) Client-based (b) Load-based (c) Proximity-based

Fig. 2. Load balancing policies

tunnel. A VPN server is set up at the NITOS server, while the PLE nodes are VPN clients, located at several different countries spanning Europe. Bridging the NITOS server VPN interface with the experimental interface that connects to one of the two OpenFlow switches, all the resources operated in our experiment have the illusion of being in one single LAN. The OF switches behavior is adjusted from an external OF controller, located at the NITOS server.

The topology of our experimentation is illustrated in Figure 3. The two physical OF switches are interconnected through a single link, while one of the NITOS nodes behaves as a third OF switch, with the aid of the Open-vSwitch (OvS) software [18]. The three switches shape a tree topology, where one of the physical OF switches is the root. The node operating the OvS software features a wireless interface too, able to bridge wireless with wired traffic. We focused on a realistic experimental scenario, where the content clients with bandwidth requests which do not exceed the wireless link capacity are wireless connected to the OvS and the rest of them are connected to the leaf OF switch. All the content clients request services from geographically distributed content servers across the Europe. Such a scenario could be applied in a real world situation, such as in the case of the VPN of a European institution, with moving users who should be efficiently served by multiple distributed servers.

Finally, our overall experimentation is orchestrated by the cOntrol and Management Framework (OMF) [19]. NITOS and PLE testbeds are already federated by exploiting the federation capabilities of OMF, thus allowing for large scale experimentation such as the one for evaluating our scheme. Using OMF, we configure transparently the experiment topology, assign IP addresses, raise the OF controller and generate traffic requests. Using OMF's corresponding library, OMF Measurement Library (OML), we managed to get measurement aggregation from measurement points defined in the applications running, into a SQLite database, in order to assess the results of our experiments.

B. OF controller outlines

The OF controller is the most essential part of our design and is implemented using the Trema framework [20]. The OF controller receives indications from the switches, every time the switches receive a frame of an unknown flow. Then, the controller sets up an appropriate flow entry in the OF switch, allowing the switch to forward immediately the following packets without further controller involvements. The flows in the proposed CCLAN are specified by the destination MAC address and Ethernet type. If the Ethernet type is different from ARP, the controller imposes the switch to forward the packets in the same way that a normal switch does. In any other case, the controller forwards or filters the ARP packets accordingly, implementing one of the strategies that previous Section III describes.

In the **Client-based policy**, the OF controller maintains an appropriate hash table that maps each content identifier (server IP address) to another hash table, which stores a mapping between the clients and the servers. Each time that a new ARP Request is initiated from a new client, the OF controller checks the number of clients served by a specific server, and chooses the one of the available ones with the least client load. The controller is aware of all the possible servers in the system, since it intercepts all the transmitted ARP Replies (except for the initialization phase of our system, when the controller chooses the server that responds first). Once the new ARP Replies get transmitted, the OF controller will propagate only the one of the previously selected server, filtering the rest ones.

Concerning the implementation of the **Load-based policy**,which is harder to implement, a periodical check of the statistics held by the OF switches is required. The main differentiation of this policy compared to the first one, is the extended data storage structure that holds more information, except for the client-server mapping, while it enforces the periodical reformation of this mapping. The client-server mapping is done using the same building blocks that the previous policy follows. However, after a specified time interval (10 secs for our experiments) the OF controller collects the OF statistics and estimates the current traffic load of each server, based on the amount of total per port traffic. We remind that the OF switches have an internal entry for each port, which maintains its aggregate traffic load. Therefore, when the inspector OF controller detects an overloaded server, it may reassign the most demanding client from the overloaded server to the least loaded one.

The last **Proximity-based policy** is the most straightforward implemented algorithm, since it does not require any complicated data storage structure and the filtering decisions are taken based on the first-come-first-served algorithm. More

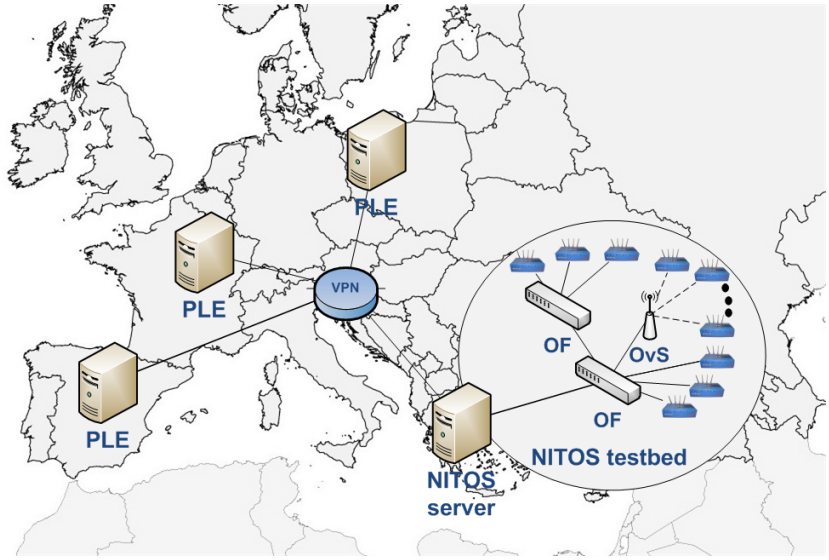


Fig. 3. Experimental Network Topology

specifically, the OF controller does not take into consideration any traffic load statistics or number of clients per server, but just forwards the first received ARP Reply, presuming that it has been generated by the nearest located content server. It can be easily concluded that this approach would be very efficient in terms of response time, in case that the network is not congested. However, in any other case, this specific policy is actually ignoring any load balancing effort.

V. Experimentation Results

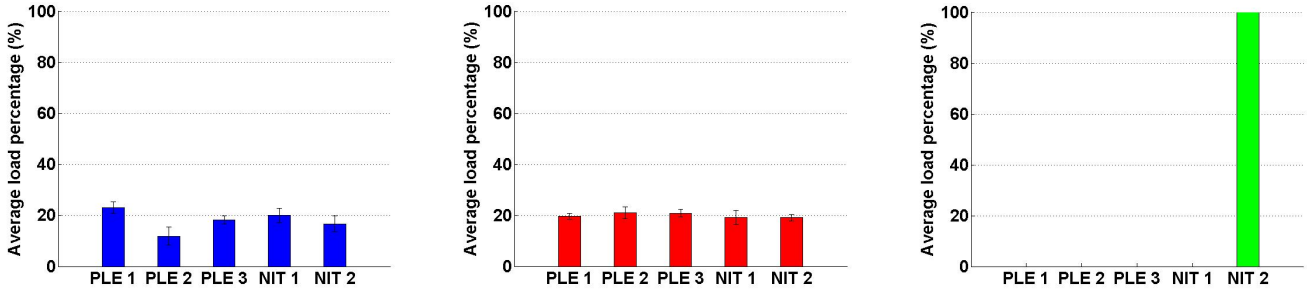
Our experiments are targeting at evaluating both the compared policies as well as the validity of our CCLAN implementation. On one hand, we collected some accurate measurements regarding the real performance of each implemented policy, while on the other hand, we proved that our CCLAN implementation is working properly and can be efficiently integrated in any production network. A total number of 15 nodes and 2 OF switches were committed to our experiment, 12 nodes from the NITOS testbed (9 clients, 2 servers and one acting as an OvS) and the two OF switches, and 3 nodes from the PLE testbed. In order to differentiate the response times of the server nodes that are located closely to each other, we injected artificial delay on their incoming and outgoing links, using the ”netem” application for the NITOS nodes, and the ”netconfig” application for the PLE resources. Apart from the some remarkable switch behavior that we noticed, on which we elaborate later, the experimental results were expected.

For our experiments, we set up all the 5 servers in our system with one IPv4 address, emulating that all of them serve the same content. We used the ”iperf” application for generating traffic to the server nodes, and cleared all ARP entries that the clients in our system where holding. Three of our clients send out traffic of 5Mbps, three of 10Mbps and the remaining three of 15Mbps. The clients that issue a 5Mbps load (15Mbps in summary) are assigned to the OvS, which operates at IEEE 802.11a mode, in channel 100, thus ensuring no external interference on the wireless link. We assume that the DNS service responsible for mapping the content identifiers to an IP address is running at a different network segment, and thus we focus on the procedure after this mapping. Each experiment was run 5 times, in order to ensure the validity of our results. Our implementation proved to be a solid one, since in every run we kept getting the same results in the way client requests were allocated to the servers.

Concerning the evaluation of the Client-based policy (Figure 4(a)), the requests are always assigned to a corresponding server, depending on the way that they arrive at the OF controller. The fact that *PLE 2* resource was more distant than the other servers (ping delay from the NITOS server to *PLE 2* had a 10ms extra overhead than *PLE 1* and 5 ms than *PLE 3*) resulted in the assignment of less client requests to it, in some of our experiment runs. Since we have 9 clients and 5 servers, we were expecting 2 requests per server would have been assigned, apart from one server that would serve only one client. The experimental results validate our assumption.

Regarding the Load-based policy, we can see that the overall load was almost equally distributed among the servers in our system (Figure 4(b)). The measurement of this policy has been the hardest to measure, since clients were continuously moving from the different servers each time that the ARP timeout expired, and an ARP request was issued from the corresponding client. The ARP timeout was set to 10 seconds for our experiments. The total aggregate throughput that each client has transmitted was measured by summing up the total bandwidth that the servers have received from each one.

Finally, concerning the third Proximity-based policy, all the requests were assigned to the most proximate server, the one operating on the NITOS side (Figure 4(b)). Remarkable is a switch dependent behavior that we observed when evaluating this policy. Since that we always select the server that has the least ARP response time, we would expect that the similar response times of the NITOS side servers would have resulted in the overall load to be balanced between these two servers.



(a) Client-based policy (b) Load-based policy (c) Proximity-based policy

Fig. 4. Load balancing comparison among the three policies

However, after adapting our approach, we repeated the experiments by using only NITOS nodes as content servers. The results we got were similar, and after injecting delays on the outgoing links of each server we concluded the following; the switch scans the port serially based on their port identification number. Therefore, if the ARP Replies arrive at the same time (or within a time interval that the switch is not aware of, the switch will assign the requests based on the port identifier.

TABLE I
AVERAGE RESPONSE DELAY COMPARISON

Default ARP process	3.8 ms
CCLAN ARP process	8.5 ms

Concluding our evaluation, noteable are the extra delays that our OF based system may impose in the ARP process. We managed to measure the OF overhead that was imposed on our scheme, after sniffing ARP packets that were using our policies or the default ARP process. Our results show that our process was suffering from an extra 3-5 milliseconds delay, during the ARP process. The average delay we measured for each ARP process is presented in Table I. This happens due to the decoupled nature of the OF controller from the physical switch, and the extra communication overhead that is imposed between the controller and the switch for every unknown flow. However, the results from our setup show that apart from this initial mere overhead, no other degradation in the overall system performance was imposed.

VI. CONCLUSION

In this paper we propose and implement a content-centric network architecture for LANs, inspired by the widely used OF technology and the insights behind the CCN approach. We designed and implemented three load balancing policies, considering all possible desirable features that several applications may require. We analyzed the performance of these policies in terms of both load balancing and time response. We intent to extend the current work towards two directions: The first one is to apply the content-centric logic in more extended networks, approaching the limits of the Internet. The second one is to analyze further the evaluated policies, researching deeper in the appropriate tuning of their several configuration parameters, as well as to propose even more sophisticated policies.

REFERENCES

[1] D. Staessens, S. Sharma, D. Colle, M. Pickavet, and P. Demeester. Software defined networking: Meeting carrier grade requirements. In *Proceedings of IEEE LANMAN*, 2011.
[2] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In *Proceedings of ACM CoNEXT*, 2009.
[3] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: enabling innovation in campus networks. In *Proceedings of SIGCOMM*, 2008.
[4] Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, and Guru Parulkar. Can the production network be the testbed? In *Proceedings of USENIX OSDI*, 2010.
[5] Nitlab: Network implementation testbed laboratory, http://nitlab.inf.uth.gr/NITlab.
[6] Ple: Planetlab europe, http://www.planet-lab.eu/.
[7] Pursuit: Publish-subscribe internet technology, http://www.fp7-pursuit.eu/PursuitWeb/.
[8] 4ward: Architecture and design for the future internet, http://www.4ward-project.eu/.
[9] Convergence, http://http://www.ict-convergence.eu/.
[10] Andrea Detti, Nicola Blefari Melazzi, Stefano Salsano, and Matteo Pomposini. CONET: a content centric inter-networking architecture. In *Proceedings of ACM SIGCOMM*, 2011.
[11] Stefano Salsano, Andrea Detti, Matteo Cancellieri, Matteo Pomposini, and Nicola Blefari-Melazzi. Transport-layer issues in information centric networks. In *Proceedings of ACM ICN*, 2012.
[12] M. Day, B. Cain, G. Tomlinson, and P. Rzewski. A Model for Content Internetworking (CDI). RFC 3466 (Informational), February 2003.
[13] A. Vakali and G. Pallis. Content delivery networks: status and trends. *IEEE Internet Computing*, 7(6):68–74, Nov/Dec 2003.
[14] Alexandros Biliris, Chuck Cranor, Fred Douglis, Michael Rabinovich, Sandeep Sibal, Oliver Spatscheck, and Walter Sturm. CDN brokering. *Comput. Commun.*, 25(4):393–402, Mar 2002.
[15] Diego Perino and Matteo Varvello. A Reality Check for Content Centric Networking. In *Proceedings of ACM ICN*, 2011.
[16] Yan Chen, Randy H. Katz, Y H. Katz, and John D. Kubiatowicz. Dynamic Replica Placement for Scalable Content Delivery. In *Proceedings of IPTPS*, 2002.
[17] Chengdu Huang, Gang Zhou, Tarek F. Abdelzaher, Sang Hyuk Son, and John A. Stankovic. Load Balancing in Bounded-Latency Content Distribution. In *Proceedings of IEEE RTSS*, 2005.
[18] Open virtual switch, http://openvswitch.org/.
[19] Thierry Rakotoarivelo, Maximilian Ott, Guillaume Jourjon, and Ivan Seskar. OMF: a control and management framework for networking testbeds. *SIGOPS Oper. Syst. Rev.*, 43(4):54–59, Jan 2010.
[20] Trema: Full-stack openflow framework in ruby and c, http://trema.github.com/trema/.