Auto-Collaboration for Optimal Network Resource Utilization in Fixed IPv6 Networks

Nikolay Tcholtchev, Arun Prakash, Ina Schieferdecker
Fraunhofer FOKUS
Berlin, Germany
{*firstname.lastname*}@fokus.fraunhofer.de

Ranganai Chaparadza
IPv6 Forum, ETSI AFI
Germany
ran4chap@yahoo.com

Razvan Petre
Testing Technologies IST GmbH
Berlin, Germany
razvan.petre@testingtech.com

***Abstract*—Autonomic Networking is seen as one of the hopes to improve the Quality of Service (QoS) provided to end users in the Internet. Especially in the context of Video on Demand (VOD), content and network providers have to deal with the dynamics of network and server loads which can severely impair the services provided to end users. Our recent research efforts seek to provide a remedy for this issue by introducing an autonomic framework for optimal network resource utilization that allows content and network providers to cooperate, and take into account aspects of the network state as well as server loads while optimizing the services for their subscribers. The framework is specified, designed, and implemented in an IPv6 testbed that was setup by the participating partners in the course of a large scale European project. Finally, a scenario realized in this testbed is presented that illustrates the benefits of the proposed approach.**

***Index Terms*—Autonomics; IPv6; Auto-Collaboration; Load Balancing; GANA; ONIX; QoS; QoE; Video on Demand (VOD)**

I. Introduction

The Internet is one of the significant achievements of our times. Video-sharing, Internet Protocol Television (IPTV), Video on Demand (VOD), social networking, email, to name a few, are some of the facets that make the Internet both popular and indispensable. Unfortunately, it is this huge success that also contributes to its problems [1], necessitating in the development of new network architectures, infrastructures, and management mechanisms. Autonomic Networking [2] is one such new and emerging network management paradigm that addresses the issues facing the current Internet architecture. The focus of this paper is to showcase how Autonomic Networking favors and enables the optimal utilization and management of network resources.

The increasing number of devices capable of consuming rich and diverse media content from the Internet introduces new challenges for the content providers and especially for the network providers. For instance, it is very hard to predict *what* the users want, *when* they want it, and *how long* they will be interested in a particular video/show. Thus, when a large number of users consume rich media content in the form of high definition movie clips during the same time frame and from the same content provider, the quality of experience decreases dramatically for the users due to the high utilization and congestion of some paths from the network. This is true even if the content is replicated at several location and it is served in a distributed manner. It is important to note that this happens especially with services (e.g. youtube.com, BBC iPlayer, vimeo.com, etc) that offers their users with the possibility to choose from a very large set of movie clips and not in the case of classic IPTV. Furthermore, it makes the life difficult for the Network Operator (NO) as they cannot build a multi-cast tree due to the diverse nature of the content. In contrast, for IPTV it is easy to build a multi-cast tree and to provide the content with high standards of Quality of Experience (QoE) for the user.

Even though some Content Providers (CPs) have their own resource optimization mechanisms, these are load-based optimizations techniques on the server-side that do not consider the load characteristics of the network. As network load characteristics data, such as *network load* and *congestion points* are sensitive information, NOs are hesitant to share it with the CPs. Ultimately, the end user suffers due to the resulting bottlenecks and non-optimal use of network resources or due an explicit reduction of their bandwidth by the NO during *prime time* [3].

Thus, NOs are faced with a big challenge. They can invest to increase the network capacity. However, this is not only expensive but also economically inefficient as the bottlenecks occur only during a limited duration of the day, i.e., during the peak hours. A better solution would be to collaborate with the CPs and achieve a better utilization of network resources. This is feasible only when both NO and CP are operated by the same organization, but almost impossible when they are operated by different entities (and this is the case in most of the situations), since the NOs don't want to give sensitive information about the state of the network to third parties.

Thus, VOD is one of facets that requires optimal utilization of network resources and is the focus of our research. We demonstrate how an autonomic network based on the *Generic Autonomic Network Architecture (GANA)* [4] enables the auto-collaboration between network providers and content providers for an optimal utilization of the network resources towards providing high QoE standards to the end user.

The rest of the paper is structured as follows: In Section II, we describe the ONIX system that provides an information exchange facility and allows data to be exchanged between the NO and CP in a standardized fashion. In Section III, we outline our proposed auto-collaboration framework that

978-1-4673-4941-3/12/$31.00 ©2012 IEEE

favors and facilitates the optimal usage of network resources. The implementation experience and results from our auto-collaboration demonstration scenario are detailed in Section IV. Finally, we provide our conclusions and insights in Section V.

II. Overlay Network for Information eXchange

The **O**verlay **N**etwork for **I**nformation e**X**change (ONIX) is an IPv6 based solution for scalable, fault-tolerant and secure information exchange in large-scale networks. The type of exchanged information includes: node and network configurations, general non-time-critical information, monitoring information, history of alarms and incidents etc

ONIX is a distributed system that forms an overlay network of nodes. The architecture of ONIX within a network node is illustrated in Fig. 1. The access to an ONIX server is realized over extensions of the DHCPv6 protocol [5], which we denote as DHCPv6++. These extensions were presented in our previous work [6]. Moreover, ONIX relies on traditional Distributed Hash Table (DHT) protocols, e.g. Chord [7], [8]. However, ONIX is designed as to be independent of the underlying DHT protocol since it requests the corresponding DHT functionality in an abstract way.

Following points constitute the core functionality of the ONIX system: 1) *Storing and Retrieving Information*: ONIX offers services (XML Parser & Key Generation Module, Query Module, and Replication Module) for storing and retrieving information. Both push and pull models as well as add, update and remove operations are supported. The information stored in ONIX is described using XML. 2) *Query for Information*: The Query Module of ONIX implements a query language similar to XPath [9]. 3) *Disseminating the information*: Information dissemination upon request, periodically or event triggered. The information dissemination features of ONIX allow other components to subscribe for receiving information of interest after it has been published to ONIX. This is realized by the Subscription Handler in Fig. 1.

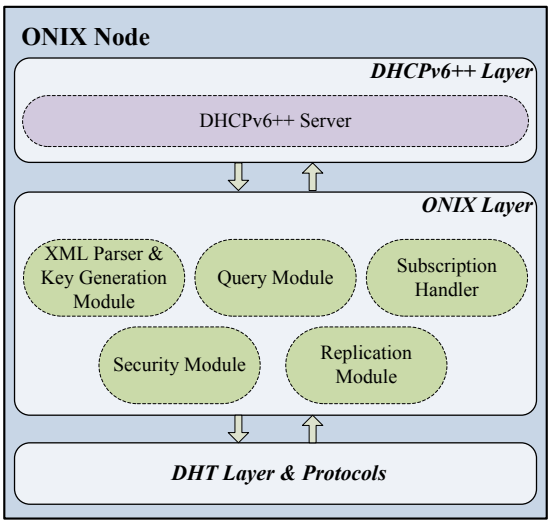

Fig. 1. Overview of the Architecture of an ONIX node

In addition to its core functionality, a number of non-functional requirements are implemented and realized by the ONIX system: 1) *Security aspects*: The Security Module of the ONIX system implements belonging mechanisms for Access Control, Authentication, and Trust based on special tokens which allow only trusted/authenticated nodes to participate and access information from the ONIX overlay. 2) *Improved reliability, fault-tolerance and accessibility*: These aspects are realized by the Replication Module in Fig. 1. Thereby, the DHT keys, the data and the subscription information are replicated inside the ONIX system and even in the case of failures of several ONIX servers, the information published into the ONIX system is still accessible.

Another interesting aspect is given by the way ONIX is discovered in an IPv6 network. Traditionally, DHCPv6 provides configurations only for on-link neighbors. However, recent research efforts and developments [10], [11] (that recently materialized into standards [12]) allowed us to use DHCPv6 relays for discovering and accessing ONIX DHCPv6++ servers over multiple hops.

III. Auto-Collaboration Framework for Optimal Network Resource Utilization

In spite of advanced device and service discovery standards and current practices available today, the network operator has a very limited number of possible solutions, and none with acceptable results for the end users. They cannot use multicast protocols as the content is very diverse and users' behavior hard to predict. There is no interface defined that can be used to exchange information in a standardized way between Internet Service Provider (ISP) and Content Providers (CPs) regarding the load in the network. Further, ISPs do not intend to provide sensitive information about their network topology or congestion points to the CPs, unless this is presented in an abstract manner and through a standardized interface.

Thus, to address this problem, we propose the introduction of the network metric called the *Ingress Egress Choice Metric (IECM)*. The IECM metric is computed for each possible combination of *ingress* points and *egress* points in the network, where at the *egress* point there is located at least one server from a CP. This metric encapsulates information about the *cost* of serving a request that comes into the network from the ingress point, by the CP's server located at the egress point. The *cost* is an abstraction of the current state of the path(s) from the *ingress* point to the *egress* point, including the *load* of the routers along the path(s), *link utilization*, *number of hops*, etc. However, since *IECM* is just a number with values in the interval [1,5], the ISP is able to compute a usable metric for the CPs, reflecting the state of the network, without exposing sensitive information like the congestion points in the network or how the routing is adjusted for CPs.

Fig. 2 showcases the proposed logical and physical architectural set up for demonstrating the auto-collaboration mechanism in an autonomic network. However, for the purpose of demonstrating the key novelties introduced by our approach and for maintaining the network set up within reasonable size

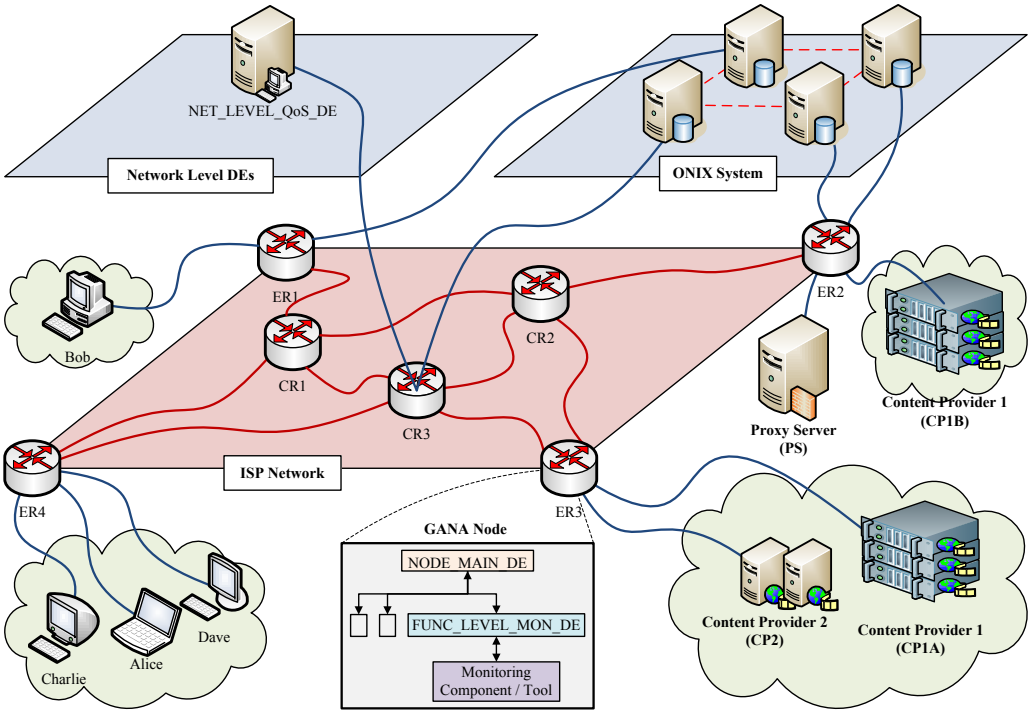

Fig. 2. Overview of the Architecture of an ONIX node

and complexity, some real world constraints have been relaxed. For instance, as it can be noticed there are several end systems that are directly attached to some core routers or edge routers, which is usually not the case in real world networks.

The autonomic network showcased here is based on GANA [4], all nodes, with the exception of the end users, namely the *home users* and CPs are assumed to be GANA conformant. GANA introduces the concept of a hierarchy of autonomic Decision Elements (DEs) both inside a network device and in the network as a whole. The hierarchy includes DEs at different levels of abstraction, such as Network-Level, Node-Level, Function-Level and at the Protocol-Level. A detailed discussion on GANA is available in [4], [13].

As the emphasis of our work is on how auto-discovery can help the auto-collaboration between ISPs and CPs and how it can lead to more complex self-* functionality like self-adaptation and self-optimization, we limit our discussion to the following GANA conformant network components:

- *Network-Level Quality-of-Service Decision-Element (NET_LEVEL_QoS_DE)*: The main role of this DE is to compute the *IECM* for the network.
- *Node Main Decision Element (NODE_MAIN_DE)*: This DE aggregates and publishes the capabilities of the node to the ONIX system. It is also responsible for receiving monitoring requests from the NET_LEVEL_QoS_DE and to forward the request to the Function-Level Monitoring Decision-Element (FUNC_LEVEL_MON_DE). The communication between the Node Main DE and the NET_LEVEL_QoS_DE is realized using ICMPv6++.
- *FUNC_LEVEL_MON_DE*: The role of this DE is to self-describe itself and its Managed Entity (ME) (the monitoring component) and to forward the monitoring requests from the NODE_MAIN_DE to the ME.
- *Monitoring Component*: This provides the information required by the NET_LEVEL_QoS_DE for computing the Ingress Egress Choice Metric.
- *Proxy Server (PS)*: The proxy server for the content providers is responsible for choosing the best location from which to deliver content, based on the IECM, computed by the NET_LEVEL_QoS_DE, and the *Load Metric*, computed by the CP's server. It also includes a DHCPv6++ client module as it needs to communicate with the ONIX system.
- *ICMPv6++* [14]: The protocol used for DE to DE communication. In this scenario it is required for the communication between the NET_LEVEL_QoS_DE and the NODE_MAIN_DE.
- *DHCPv6++*: The protocol used by all network entities that need to communicate with the ONIX system.

A detailed description of functionality of all GANA's DEs is beyond the scope of this paper. They are described in [4] and associated literature available as project deliverables in the EFIPSANS project web page [15].

The auto-collaboration mechanism involves three phases: *Network Setup Phase*, *Service Initialization Phase* and *Operational Time Phase*. They are detailed below.

1) Network Setup Phase: This phase involves with the auto-discovery and auto-configuration of the network components. Thus, at the completion of this phase, all the network routers and devices are assumed to have been discovered

State	Ingress Egress Choice Metric (IECM)				CP Load Metric		
	ER1-ER2	*ER1-ER3*	*ER4-ER2*	*ER4-ER3*	*CP1A*	*CP1B*	*CP2*
State after Service Initialization	2	2	2	1	1	1	1
State after Alice Requests	2	2	2	**2**	**2**	1	1
State after Bob Requests	**3**	2	2	2	2	**2**	1
State after Charlie Requests	3	2	2	**3**	2	2	**2**
State after Dave Requests	3	2	**3**	3	2	**3**	2

TABLE I
METRICS VALUES OVER TIME

and configured. For simplicity without going into the details, we assume that the *Network Setup Phase* has been carried out. A detailed description of the auto-discovery and auto-configuration mechanisms in a GANA conformant network has been published [16].

2) Service Initialization Phase: In this phase, all the ONIX related subscriptions are carried out. On one hand, the NET_LEVEL_QoS_DE subscribes to the ONIX to receive notifications regarding new network devices that have monitoring and streaming capabilities. Thus, when a new streaming server or edge router with monitoring capabilities is available in the network, they publish their capabilities to ONIX, which in turn notifies the NET_LEVEL_QoS_DE. The NET_LEVEL_QoS_DE then requests the edge routers to activate a specific monitoring component and supply it with the monitoring data in a periodic fashion. When monitoring data from several edge routers are available and/or when network load characteristics change, the NET_LEVEL_QoS_DE (re)computes the IECM metric and publishes it to ONIX. On the other hand, the proxy servers (e.g. PS1) of the CPs (e.g. CP1) attached to the edge routers subscribes to the ONIX to receive the IECM metric when a new metric value is available.

3) Operational Time Phase: When ONIX receives a new value of IECM metric where the *egress* point is either CP1A or CP1B, it informs the proxy server about the new value. The proxy server computes then its own internal metric that indicates from which location to deliver content by taking into account both the IECM metric and also the information about the load of the media servers.

In order to showcase the auto-collaboration mechanism, lets assume the following IECM metric values as shown in Table I for our topology shown in Fig. 2. Thus, for instance, if *Alice* requests for some content from CP1, according to the IECM metrics in Table I, she would get the content from CP1A. Now if *Bob* also wants content from CP1, according to the new IECM metrics, he can receive it from either CP1A or CP1B. But since, CP1A is already providing content to *Alice*, for load balancing reasons, CP1 may decide to provide from CP1B. Now lets assume that *Charlie* is also interested in watching a TV show, but provided by CP2. In this case, the path between the *ingress-egress* pair ER4-ER3 is loaded. Now lets assume that *Dave* wants to watch a TV show from CP1. The load balancing application used by CP1 would now take into consideration both metrics. The servers load is similar as both CP1A and CP1B have one user each. However, IECM metric for ER4-ER3 is greater than ER4-ER2. Thus, its wise to deliver content to *Dave* from CP1B.

Thus, through the auto-collaboration mechanism, the network resources can be optimally utilized through exchange of information between the network operator and the content provider, without compromising the business needs of one another. In the sections below, we discuss how we realized and validated our auto-collaboration approach.

IV. SCENARIO IMPLEMENTATION & DEMONSTRATION

A. Implementation and Testbed

The proposed components were implemented and evaluated in the scope of the EU-FP7 EFIPSANS project [15]. The dynamic aspects of the proposed architecture were evaluated through the realization of a concrete scenario (presented in detail in the next section) that exemplifies the benefits of our framework for auto-collaboration. The testbed and the components implemented for the scenario realization are shown in Fig. 3. This IPv6 based testbed was put together in the scope of the EFIPSANS project and consists of a number of Linux-Quagga [17] based routers running across the sides of various partners. The integration of the partners' networks is facilitated by VPN tunnels between the involved sides. The illustration in Fig. 3 highlights the network segment, which was realized by Fraunhofer FOKUS. The remote partner networks are attached by a VPN tunnel to Core Router 3 (CR3).

The ONIX system was implemented in Java and uses Chord and the belonging open source implementation "Open Chord" [8] as a DHT layer. The generation of the keys for the distributed hash table is based on SHA1 [18] and Rabin Fingerprints [19]. For the retrieval of information from the DHT layer we use Bloom filters [20] as implemented by the Java Bloom filter library [21]. Within our testbed (Fig. 3) there are three ONIX servers that build a distributed Chord overlay - ONIX-A, ONIX-B, and ONIX-C. These servers are accessed by content and network provider components over DHCPv6++ extensions transmitted over UDP, i.e. datagram sockets in Java.

The *FUNC_LEVEL_MON_DE* is implemented in Java and is running on the border routers (BR1, BR2, BR3, BR4) of the provider network. This piece of software is easily extensible by plug-ins [22] that wrap around off-the-shelf monitoring tools such as *iperf* [23], *ethtool*, *net-SNMP* etc. In order to measure the performance of the service provider network with respect to Quality of Service (QoS)'s *Key Performance Indicators (KPI)*, the FUNC_LEVEL_MON_DEs on BR2 and BR3 start an *iperf* server that is used by *iperf* client components,

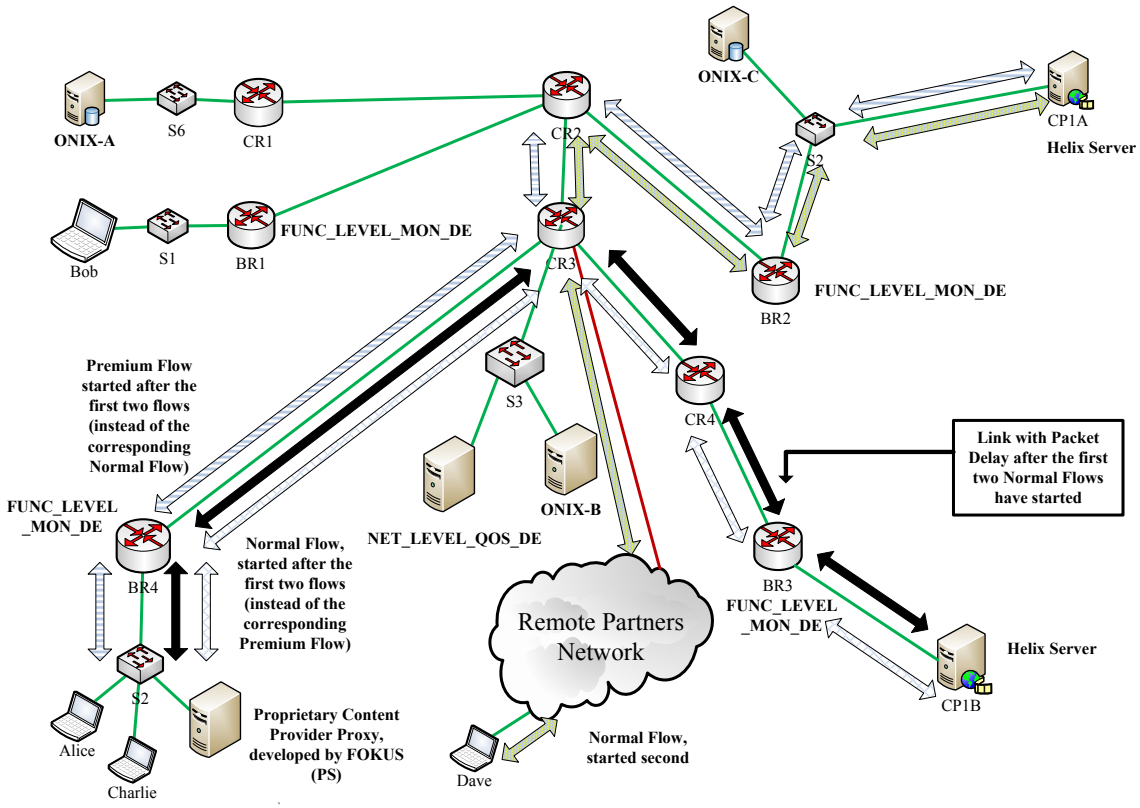

Fig. 3. Realization of the Auto-Collaboration Scenario in the Integrated EFIPSANS Testbed.

orchestrated by the Monitoring DEs on BR1 and BR4, in order to periodically measure the performance of the network. The obtained metric values for *jitter*, *delay*, *bandwidth*, *packet loss*, and *out-of-order packets* are communicated over ONIX to the NET_LEVEL_QoS_DE running in the S3 sub network attached to CR3.

This Java based NET_LEVEL_QoS_DE calculates the IECM based on the measured KPI values and stores them into ONIX using the nearby ONIX-B server and the corresponding DHCPv6++ extensions. The IECM values are communicated over the Chord ring to the Java based Content Providers (CPs)'s PS in the S2 subnet on the left side. This proxy (PS) is referred by clients every time they want to establish a connection to one of the Helix Video Servers on the right side of Fig. 3. Thereby, the proxy (PS) dispatches the requesting client to the appropriate server by taking into consideration the load (number of clients) on the servers and the IECM values provided by the NET_LEVEL_QoS_DE. In order to make the client software communicate with the proxy (PS), we developed wrappers around the Linux version of the *mplayer*. These wrappers are responsible for communicating with the proxy and setting up the video session.

B. Demonstrated Scenario

Based on the implementation and deployment of the proposed architectural components as described in Fig. 2 and Fig. 3, a concrete scenario illustrating the benefits of the proposed architecture was realized. The scenario distinguishes between normal and premium network users. Normal users use traditional load balancing that relies only on the load (number of clients) on the servers for dispatching the clients, while for premium users both the load of the servers and the network load towards the servers (IECMs) are considered. The scenario starts with a normal flow (black color) originating from the S2 network on the left side in Fig. 3. This flow is dispatched by the proxy (PS) to CP1B Helix Server (at the start both servers are completely free). Secondly, Dave (from the partners' networks on the bottom) starts another normal flow that is dispatched to the CP1A Helix Server.

After these two flows, we use the *netem* tool [24], in order to introduce *packet delay*, *jitter* and *packet re-ordering* on the link CR4-CR3. This network impairment is immediately reflected in the QoS measurements performed by the FUNC_LEVEL_MON_DEs, and correspondingly the IECMs calculated by the NET_LEVEL_QoS_DE worsens for the conditions towards CP1B. Subsequently, in order to illustrate the impact of our framework, a normal flow is started by one of the clients from within the S2 sub network on the left side.

This flow is dispatched only based on the server load and is assigned to the CP1B, server since the proxy (PS) selects CP1B first in case of equal server load. This normal flow really suffered from the bad network conditions towards CP1B and was very often not able to establish a video connection to the Helix Sever. In case that instead of a normal flow, a premium

flow is started (again from within S2), then the network conditions are taken into account and the premium client is dispatched to CP1A that results in the smooth establishment of a video connection and good quality. Hence, we can observe how our framework improves the session establishment phase for premium users, based on a combination between the server load and the load of the network towards the involved. servers.

V. Conclusion

In this paper an approach to realizing auto-collaboration between network and content providers towards optimal resource utilization and improved Quality of Experience (QoE) in fixed IPv6 networks was presented. Based on the Generic Autonomic Network Architecture (GANA) reference model a set of components was defined that enable the content and network provider to exchange information regarding the state of the network paths towards content provider servers. This makes it possible to efficiently dispatch video streaming clients and cope with the dynamics of network and server load.

The key players that benefits from the innovations introduced by our approach are the Network Operators (NOs) and end users. The NO benefits during operational time, through the optimal utilization of the network resources under given Quality of Service (QoS) constraints even during peak hours. Furthermore, auto-collaboration favor overall load balancing in the whole network, leading to revenue maximization for the NO. Finally, the NO is able to share sensitive information in an abstract manner without compromising their business goals and strategies for their own benefit. From an end user point of view, the Content Providers (CPs) are able to utilize network specific information in their load balancing mechanisms to provide the required quality, enabling them to satisfy and maintain their current subscriptions while looking for new content and customers to increase their revenue streams, instead of investing in additional hardware or servers. On the other hand, *home users* receive the desired service quality justifying their subscriptions to an Internet service.

Acknowledgment

The authors would like to thank their colleagues from the EFIPSANS [15] project for their support in completing this work.

References

[1] D. Hausheer, P. Nikander, V. Fogliati, K. Wünstel, M. Á. Callejo-Rodrígues, S. R. Jorba, S. Spirou, L. Ladid, W. Kleinwächter, B. Stiller, M. Behrmann, M. Boniface, C. Courcoubetis, and M.-S. Li, "Future Internet Socio-Economics - Challenges and Perspectives," in *Future Internet Assembly*, 2009, pp. 1–11.

[2] Z. Movahedi, M. Ayari, R. Langar, and G. Pujolle, "A Survey of Autonomic Network Architectures and Evaluation Criteria," *IEEE Communications Surveys & Tutorials*, vol. PP, no. 99, pp. 1 –27, 2011.

[3] S. Hansell, "BTs Battle Against the BBCs Online Video," Jun. 2009. [Online]. Available: http://bits.blogs.nytimes.com/2009/06/12/bts-battle-against-the-bbcs-internet-video/

[4] R. Chaparadza, "Requirements for a Generic Autonomic Network Architecture (GANA), suitable for Standardizable Autonomic Behavior Specifications for Diverse Networking Environments," *International Engineering Consortium (IEC), Annual Review of Communications*, vol. 61, 2008.

[5] R. Droms, "Dynamic Host Configuration Protocol," RFC 2131 (Draft Standard), Internet Engineering Task Force, Mar. 1997, updated by RFCs 3396, 4361, 5494. [Online]. Available: http://www.ietf.org/rfc/rfc2131.txt

[6] R. Chaparadza, R. Petre, A. Prakash, F. Németh, S. Kuklinski, , and A. Starschenko, "IPv6 and Extended IPv6 (IPv6++) Features that enable Autonomic Network Setup and Operation," in *Access Networks: 5th International ICST Conference on Access Networks, AccessNets 2010 and First ICST International Workshop on Autonomic Networking and Self-Management in Access Networks, SELFMAGICNETS 2010*, ser. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, R. Szabó, H. Zhu, S. Imre, and R. Chaparadza, Eds., vol. 63. Springer Heidelberg Dordrecht London New York, 2010.

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 149–160, Aug. 2001. [Online]. Available: http://doi.acm.org/10.1145/964723.383071

[8] K. Loesing and S. Kaffille, "Open Chord," Apr. 2008, version 1.0.5. [Online]. Available: http://www.uni-bamberg.de/en/pi/bereich/research/software_projects/openchord/

[9] (1999, Nov.) XML Path Language (XPath). W3 Consortium. Version 1.0. [Online]. Available: http://www.w3.org/TR/xpath/

[10] F. Németh and G. Rétvári, "Address auto-configuration methods in ipv6 networks," in *HSN Workshop*, May 2009.

[11] C. Simon, F. Németh, F. Uzsak, G. Rétvári, F. Ficsor, and R. Vida, "Autonomic DHCPv6 Architecture," in *GLOBECOM Workshops*, 2011, pp. 620–624.

[12] S. Ooghe, W. Dec, S. Krishnan, and A. Kavanagh, "Lightweight DHCPv6 Relay Agent," RFC 6221, May 2011. [Online]. Available: http://tools.ietf.org/html/rfc6221

[13] R. Chaparadza, S. Papavassiliou, T. Kastrinogiannis, M. Vigoureux, E. Dotaro, A. Davy, K. Quinn, M. Wódczak, A. Toth, A. Liakopoulos, and M. Wilson, *Towards the Future Internet - A European Research Perspective*. IOS Press, 2009, ch. Creating a viable Evolution Path towards Self-Managing Future Internet via a Standardizable Reference Model for Autonomic Network Engineering, pp. 313–324.

[14] R. Chaparadza, R. Petre, S. Cheng, L. Xin, and Y. Li, "ICMPv6 based Generic Control Protocol (IGCP)," IPv6 Maintenance (6man) Internet Draft, Mar. 2010.

[15] EFIPSANS - Exposing the Features in IP version Six protocols that can be exploited/extended for the purposes of designing/building Autonomic Networks and Services, "EC FP7-IP Project," www.efipsans.org, 2008-2011, INFSO-ICT-215549.

[16] A. Prakash, A. Starschenko, and R. Chaparadza, "Auto-discovery and Auto-configuration of Routers in an Autonomic Network," in *AccessNets*, 2010, pp. 311–324.

[17] Quagga Routing Suite. [Online]. Available: http://www.quagga.net/

[18] D. Eastlake and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)," RFC 6234, May 2011, iSSN: 2070-1721.

[19] M. O. Rabin, "Fingerprinting by Random Polynomials." TR-CSE-03-01, Center for Research in Computing Technology, Harvard University, Tech. Rep., 1981.

[20] G. Koloniari, Y. Petrakis, and E. Pitoura, "Content-Based Overlay Networks for XML Peers Based on Multi-level Bloom Filters," in *Databases, Information Systems, and Peer-to-Peer Computing*, ser. Lecture Notes in Computer Science, K. Aberer, M. Koubarakis, and V. Kalogeraki, Eds. Springer Berlin Heidelberg, 2004, vol. 2944, pp. 232–247. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24629-9_17

[21] "java-bloomfilter," Sep. 2011, v1.0. [Online]. Available: http://code.google.com/p/java-bloomfilter/

[22] N. Tcholtchev and R. Petre, "Design, Implementation and Evaluation of a Framework for Adaptive Monitoring in IP Networks," in *EASE 2012, 9th IEEE International Conference and Workshops on the Engineering of Autonomic & Autonomous Systems*, 2012.

[23] A. Tirumala, L. Cottrell, and T. Dunigan, "Measuring end-to-end bandwidth with Iperf using Web100," in *Web100, Proc. of Passive and Active Measurement Workshop*, 2003, p. 2003.

[24] S. Hemminger, "Network Emulation with NetEm," in *LCA 2005, Australia's 6th national Linux conference (linux.conf.au)*, M. Pool, Ed., Linux Australia. Sydney NSW, Australia: Linux Australia, Apr. 2005. [Online]. Available: http://www.linux.org.au/conf/2005/abstract2e37.html?id=163