Self-management of Live Streaming Application in Distributed Cloud Infrastructure

PATRICIA ENDO, MARCELO SANTOS, JÔNATAS VITALINO, GLAUCO GONÇALVES, MOISÉS RODRIGUES, DJAMEL SADOK AND JUDITH KELNER, Federal University of Pernambuco
AZIMEH SEFIDCON, Ericsson Research

Currently, live streaming traffic is responsible for more than half of aggregated traffic from fixed access networks in North America. But, due to traffic redundancy, it does not suitably utilize bandwidth and network resources. To cope with this problem in the context of Distributed Clouds (DClouds) we present RBSA4LS, an autonomic strategy that manages the dynamic creation of reflectors for reducing redundant traffic in live streaming applications. Under this strategy, nodes continually assess the utilization level by live streaming flows. When necessary, the network nodes communicate and self-appoint a new reflector node, which switches to multicasting video flows hence alleviating network links. We evaluated RBSA4LS through extensive simulations and the results showed that such a simple strategy can provide as much as 40% of reduction in redundant traffic even for random topologies and reaches 85% of bandwidth gain in a scenario with a large ISP topology.

Categories and Subject Descriptors: **C.2.1 [Computer-Communication Networks]**: Network Architecture and Design—*Distributed networks*; **C.2.3 [Computer-Communication Networks]**: Network Operations—*Network Management*; **D.4.8 [General]**: Performance—*Simulation*

General Terms: Management

Additional Key Words and Phrases: Cloud Computing, self-management, live streaming, simulation

ACM Reference Format:

1. INTRODUCTION

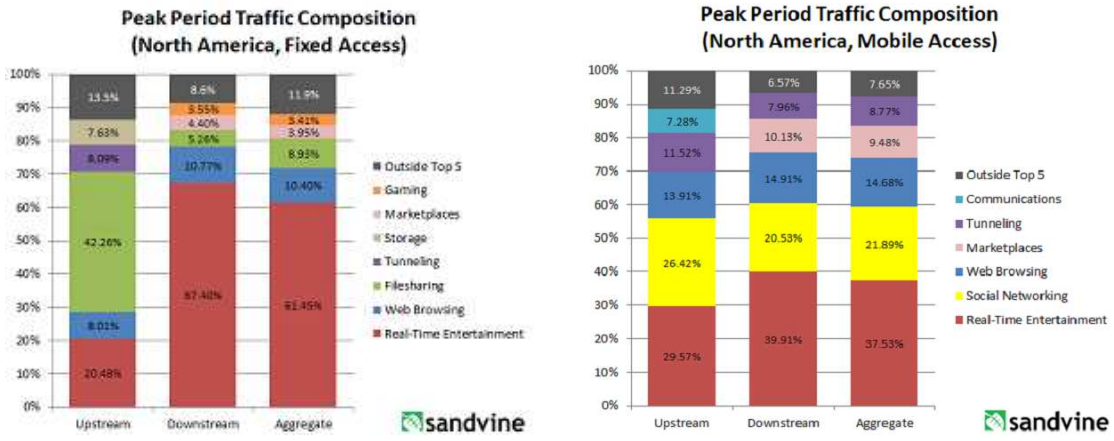
The Internet supports content dissemination towards the network edges. Its users may consume and produce shareable contents that should travel quickly over the core network. According to the Global Internet Phenomena Report [Global Internet Phenomena Report 2014], taking into consideration North America, the real-time entertainment traffic is responsible for around 61.45% and 37.53% of aggregated traffic in peak periods from fixed access network and mobile access network (Figure 1a and 1b, respectively).

In [Michel 2013], Kurt Michel, Director of Product Marketing at Akamai, stated that "*live video streaming has become an increasingly important part of the web content universe, as a variety of businesses and organizations attempt to capture a 'share of eyeball' and deliver richer, more HDTV-like experiences*". In [Zhuang and Guo 2011], authors also say that live streaming is becoming increasingly popular, and it is expected to be more pervasive in the near future. At the same time, the

This work is supported by the RLAM Innovation Center, Ericsson Telecomunicações S.A., Brazil.
Author's address: P. Endo, Av. Prof. Moraes Rego, 1235; email: patricia@gprt.ufpe.br; M. Santos, Av. Prof. Moraes Rego, 1235; email: marcelo@gprt.ufpe.br; J. Vitalino, Av. Prof. Moraes Rego, 1235; email: jonatas@gprt.ufpe.br; G. Gonçalves, Av. Prof. Moraes Rego, 1235; email: glauco@gprt.ufpe.br; M. Rodrigues, Av. Prof. Moraes Rego, 1235; email: moises@gprt.ufpe.br; D. Sadok, Av. Prof. Moraes Rego, 1235; email: jamel@gprt.ufpe.br; J. Kelner, Av. Prof. Moraes Rego, 1235; email: jk@gprt.ufpe.br; A. Sefidcon, Torshamnsgatan 21; email: azimeh.sefidcon@ericsson.com.
Permission to make digital or hardcopies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credits permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2010 ACM 1544-3558/2010/05-ART1 $15.00
DOI:http://dx.doi.org/10.1145/0000000.0000000

quality expectations of these ever-increasing audiences continue to grow. HD quality is becoming the de facto standard for all viewing experiences, from HDTV home viewing to "anywhere" viewing on mobile devices [Michel 2013].



(a) (b)

Fig. 1. Peak period aggregate traffic composition from [Global Internet Phenomena Report 2014].

In the other side, the Internet yet presents challenges for live streaming delivery due to its own architectural design concept including the lack of end-to-end reliability or performance guarantees. Furthermore, traffic redundancy is a particular problem resultant from the way live streaming applications individually transfer information over the network to independent subscribers. Basically, when serving live video content the content provider creates single unicast flows from the video source to each client resulting in the transport of several copies of the same content packets over the network. Due to such traffic redundancy, expensive bandwidth and network resources are not adequately utilized [Rajkumar and Swaminathan 2013].

A solution to address some of these challenges lies in the caching strategy commonly utilized by Content Delivery Network (CDN) providers [Nygren et al. 2010]. According to [Global Internet Phenomena Report 2014], originally the CDN concept improved application performance by caching static and popular content at the edge of the Internet, close to end users, in order to avoid “middle mile” bottlenecks and minimize traffic redundancy as much as possible. However, applications like live multimedia streaming often generate dynamic and non-cacheable content, and therefore it is necessary to rethink the solution, just as major CDN providers (such as Akamai, Limelight, and Internap) have already evolved to support live streaming delivery [Zhuang and Guo 2011].

Alternatively, Cloud Computing can be seen as a solution to support live streaming applications, since it is possible to rent virtual servers hence offering a scalable backend infrastructure to support a variable number of live video viewers. One can also make use of a Distributed Cloud (DCloud) ([Gonçalves et al. 2012], [Church et al. 2008], [Alicherry and Lakshman 2012], [Endo et al. 2011], and [Valancius et al. 2009]) that represents a Cloud provider composed of small and distributed datacenters located at different geographical regions, to provide live streaming. But despite this, it is still necessary to implement solutions to deal with video distribution by mitigating traffic redundancy.

Since classical caching mechanisms and IP multicast present many issues to support live streaming application, looking for a new scalable autonomic strategy that focuses on reducing redundant traffic in a DCloud infrastructure can be seen as a potential field for research, development and innovation.

In this paper, we present an autonomic strategy based on a **Role-Based Self-Appointment** (RBSA) framework proposed in [Endo et al. 2014]. In a nutshell, RBSA is a generic framework that can be used for several management purposes in which nodes interact with their neighbors to exchange information, and decide by themselves which role they should perform. We adopted this framework in a scenario for managing live streaming application and called the resulting architecture **RBSA4LS**. RBSA4LS can be used by the DCloud infrastructure provider to offer video a streaming application while also optimizing its network resources usage autonomously. The main goal of this paper is to describe in detail how RBSA4LS optimizes link usage through an infrastructure provider that offers a live video streaming application by establishing, at the heart of our proposal, reflector and router roles. Moreover, we simulate RBSA4LS for different scenarios in order to gain insights on and discuss how it performs in terms of bandwidth reduction.

This paper is organized as follows: in Section 2, we describe the DClouds infrastructure in order to clarify the advantages obtained for working in this environment; Section 3 describes the RBSA instantiation special case to support a live streaming application; in Section 4 we present results from conducted simulations; we discuss RBSA4LS performance in Section 5; related works are described in Section VI, and final considerations and future works are delineated in Section 6.

2. DISTRIBUTED CLOUD

Current Cloud Computing providers mainly rely on large and consolidated datacenters in order to offer their services. The physical infrastructure is composed of many nodes (processing elements and network entities) to which users gain access through some virtualization technology. At the same time, small and geographically distributed datacenters can also be attractive for Cloud providers since this type of datacenters can offer a cheaper and low-power consumption alternative that reduces the costs of large and centralized ones. These small and distributed datacenters can be built at different geographical regions and can be connected to form a Distributed Cloud (DCloud). We believe that this scenario gives a good opportunity to develop mechanisms to optimize the usage of resources appropriated to this type of application in an innovative way.

We decided to use DClouds because we obtain more flexibility to deal with the resources within the infrastructure. We require such flexibility as we are looking for applying an autonomic management strategy.

The main objective of a video streaming application is to carry video signals to viewers while assuring minimum quality. Such objective can be supported by well-dimensioned Streaming Servers (SS) and a content distribution network that transports video information through several geographically distributed servers in a timely fashion. One can consider three distinct stakeholders in this scenario: the Application Provider, the DCloud Infrastructure Provider, and the Viewer.

(1) **Application Provider**: is the entity that requests to the DCloud Infrastructure Provider (computational and network) resources to host a video streaming application. The streaming application is characterized by the Streaming Server (SS) harnessed to a specific geographical region. Beyond this, the Application Provider is responsible for producing the content of the video streaming application;
(2) **DCloud Infrastructure Provider**: represents the provider responsible for accommodating different types of services on its physical infrastructure. The DCloud Provider can make use of virtualization technologies to deal with this heterogeneity of services. These services can be an IaaS, PaaS, or SaaS. However, the focus of this paper is a DCloud Infrastructure Provider selling a specific network configuration for use by video streaming applications; and

(3) **Viewer**: is the end-user of the video streaming application that is served by the DCloud Infrastructure Provider. The viewer (or service subscriber) accesses the application though an edge router.

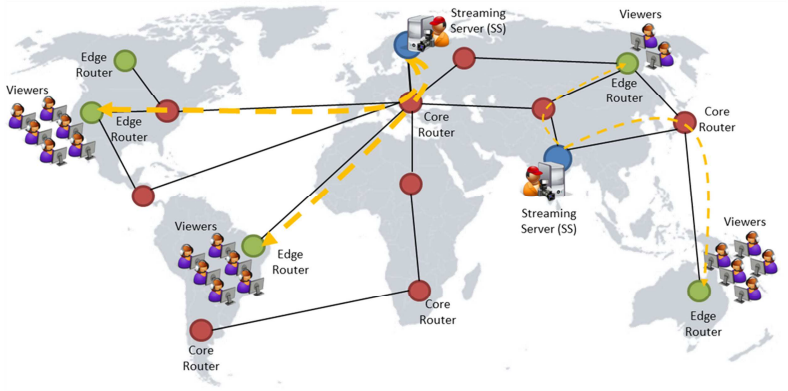


Fig. 2. Live streaming application over DCloud.

Figure 2 shows an example of a DCloud hosting service using virtualization technology. The SSs are responsible for providing the origin server and for leading with video signals. DCloud is responsible for the actual distribution of such signals to the subscribers who are accessing the SSs through edge routers. Any node in the infrastructure can be assigned as an edge router; if all nodes are edge routers, then this means that all viewers can directly access the head-end through any node in the network. If a node is not an edge router or an SS, then it is a core router. In Figure 2, we are considering five edge routers, two SSs providing different live videos, and eight core routers. The flows from SSs to viewers are assigned in yellow; the width of the flow represents the network traffic in the path taken.

This scenario supports several alternatives for managing the video streaming application. Among these is the creation of a virtual overlay network in the DCloud infrastructure. In this case, the virtual network would be seen as a virtual forest-like structure: the SS would be the root server of this forest while the core and edge routers would form the live streaming delivery tree. In the next Section, we describe an autonomic mechanism based on RBSA, where nodes are able to appoint themselves, when needed, roles autonomously depending on network and service performance.

3. ROLE-BASED SELF-APPOINTMENT FOR LIVE STREAMING (RBSA4LS)

The proposed RBSA [Endo et al. 2014] is a mechanism executed by a node in order to choose by itself a suitable new role it should play in order to optimally manage resources in a distributed networked environment, such as Distributed Cloud.

In a general way, we can describe the node's behavior as follows: nodes monitor their resources continuously and, when predetermined conditions are met, for example through trigger fires or other observed indicators, a communication process is initiated in order to achieve an optimized state performing aggregation over video flows. A trigger is considered as an event that indicates context state changes. These could reflect possible operational problems, such as bottleneck, or performance loss. Next, nodes exchange messages and analyze their states. Processing of the interchanged information takes then place to autonomously reach decisions upon which to act. In the context of our approach, a node may even chose to spin a new role or maintain its current one based on the acquired intelligence. Roles are then mapped into local actions, according to pre-established rules. Since nodes do not have a global knowledge about the whole system they only take decisions that have local scope

effects. One may describe the resulting behavior where a node assumes a specific role without seeking prior agreement from the others as self-appointment, a special type of self-configuration.

Authors in [Endo et al. 2014] provide some examples of RBSA applicability such as live streaming optimization, caching in CDN, and virtual machine management. In this Section, we will describe an RBSA instantiation scenario (called RBSA4LS) used to deal with our special problem: to provide a live streaming application that focuses on the reduction of network resource usage.

To this end, we introduce a new processing role some nodes may tale, namely, that of a **reflector**. The reflector acts as a flow concentrator with the sole responsibility of reducing redundant flows generated from for the same video server in the network. The main idea is that a node with many redundant flows from the same video streaming should be able to aggregate their traffic and redirect it to its subscribers. For this, our nodes in a network are constantly running RBSA4LS that will decide when to become a reflector based on two variables: a) the amount of redundant flows crossing itself; and b) the distance in hops to the server.

Table I shows the adaptation made in RBSA to comply with the requirements of the video streaming application. We established that the desired global behavior is to reduce the redundant traffic in the network. When the number of flows passing through a node reaches a specific value (called **threshold**), the node becomes a candidate for aggregating such flows and announces itself as one, i.e., it will send messages informing that it wants to be a reflector. The metric used to choose the best candidate is the distance in hops to the video server. The distance value can be obtained through some existing routing protocol and nodes do not need to know the distance of all others. Hence towards the end of the communication period (called **cycle**, i.e., the maximum amount of time units used to take a decision after messages were exchange was started), the candidate node to become reflector will be the one that experiences "considerable" load and is the farthest away from the existing servers.

When this node starts on its new role as a reflector, one expects to see a local reduction in streaming traffic hence alleviating network bandwidth resources. It is important to highlight that we are not interested in ensuring mutual exclusion; then it means that more than one node can become reflector at same time, and improve the bandwidth together.

In the same way nodes decide to become a reflector, they also can decide to return to router role. When nodes observed that the number of flows is less than the defined threshold, they can autonomously become a router and work normally.

Table I. RBSA Instantiation for a Live Streaming Application

GLOBAL BEHAVIOR	Reduce redundant traffic and alleviate links in the network
MONITORED VARIABLE	Number of flows of a video streaming from the same video server passing through a node
METRIC	Distance in hops to the original video server
ROLES	Router: it routes live streaming flows normally Reflector: it aggregates redundant flows and performs multicast

4. RESULTS

In order to analyze the RBSA4LS, we choose to use the NetLogo simulator[1] that is a multi-agent programmable modeling environment as our underlying simulation tool. We divided the analysis in two parts: the first set of simulations was carried out with the initial simple goal to configure the RBSA4LS. To achieve this, we used a random topology network composed of 50 nodes and executed as many as 25000 simulations of random topologies. The second set of simulations was conducted by

[1] http://ccl.northwestern.edu/netlogo/

using a real topology of a Tier 1 Internet Service Provider. The objective of the simulations now is to evaluate the RBSA4LS behavior; more specifically, we intend to evaluate the following metrics:

(1) **Bandwidth**: the goal is to measure the efficiency and gain in the reduction of link load. The bandwidth is measured as the sum of flows over each link in the network per time unit; and
(2) **Messages**: the main idea is to evaluate the overhead caused by exchanging messages between nodes to give these sufficient information to self-organize in an optimal way. This metric is calculated as the mean of all messages per time unit.

4.1 Configuration Scenario

The configuration scenario was used as a calibration phase to analyze the influence of the threshold and the cycle parameters that are previously configured. The topology is a random network composed of 50 nodes plus five edge nodes and one video server that are allocated randomly in each simulation. We built 50 different topologies generated according to the Barabási-Albert approach with only one initial node and $\Delta m=1$ [Lewis 2009] generating a tree-based topology. Each combination of these factors composes an experiment that was subsequently repeated 50 times for each combination of the present factors and their levels shown in Table III (totalizing 22600 simulations) to obtain statistical confidence.

A scenario is simulated as follows. The clients arrive to the system and are attached randomly (using a uniform distribution) to one of the five edge nodes. Each client starts one video flow and remains viewing the video to its end. The video server sends video streams (a flow per subscriber) to clients located near the edge nodes. The time unit in NetLogo is measured in ticks. Table II shows the parameters used in the simulations and their respective values.

Table II. Parameters and Levels

PARAMETER	LEVEL
Number of nodes	50
Number of edge nodes	5
Number of video streaming server	1
Cost between links	1
Mean of video streaming duration (exponential distribution)	5 ticks

Table III shows the factors of the experiment and their respective levels. A scenario with the absence of RBSA4LS self-appointment support was used as our reference comparison. The subscriber arrival rate follows a Poisson distribution. Note that this also represents the rate of creation of new video flows in the system because each client starts only one video flow.

Table III. Factors and Levels

FACTORS	LEVELS
Threshold	15, 20, 25 flows
Cycle	5, 7, 10, and 15 ticks
Arrival rate (Poisson distribution)	10 and 20 clients per tick

The stop condition was calculated based on the statistical accuracy considering 95% of confidence on our two metrics: the average consumed bandwidth per tick and the average number of messages exchanged between nodes per tick.

The bandwidth accuracy is calculated as $\frac{1.96*s_x}{\bar{x}*\sqrt{n}}$, where 1.96 is the value of the 0.975 percentile of the normal distribution, $\bar{x}$ is a moving average made over the consumed bandwidth, s_x is the standard deviation of this average, and n is the size of the time window used to compute the average that is chosen to be 500 times the cycle length. The message accuracy is calculated similarly, but the statistics are made over the number of messages exchanged between nodes. Thus, to obtain a low variance, the simulation finishes when the accuracy drops below 5% for both metrics.

Regarding consumed bandwidth, we concentrated all results in Figure 3 and Figure 4 and compare with the reference baseline case, i.e., without RBSA self-appointment. Each point of the graphics represents a mean of all simulations made for each topology, with its respective factors and levels in the legend. The confidence interval was omitted because it is very small and does not offer additional insights to the analysis.

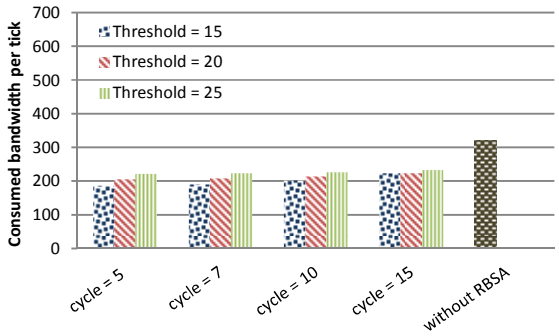


Fig. 3. Mean of bandwidth for arrival rate 10.

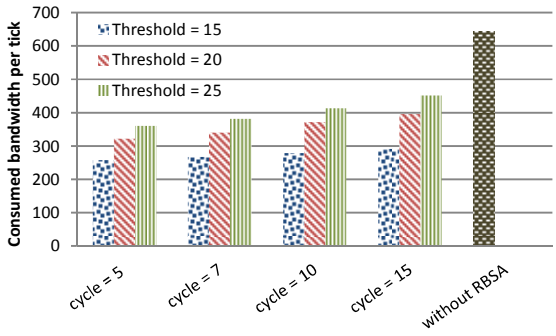


Fig. 4. Mean of bandwidth for arrival rate 20.

The results show a reduction of overall bandwidth consumption provided by the usage of RBSA4LS and such reduction is more accentuated when more clients enter the system. As expected, doubling the clients' arrival rate (from 10 to 20 clients per tick for example), also doubles the usage of the links in the network when RBSA4LS is not used. Thus, one expects to see a sharper resource savings than with the scenario with a smaller arrival rate 10. Note that the results show that a best case scenario gain is obtained for the configuration (cycle = 5 ticks, threshold = 15 flows, and arrival rate = 20 clients per tick). Here the reduction was as much as 60%.

A second aspect to be noted is that for all the values of cycle length, the bandwidth consumption of RBSA4LS with threshold 15 is statistically lower than RBSA4LS with threshold 20 and RBSA4LS with threshold 25. This occurs because the lower the threshold value is, the earlier nodes start to exchange messages and to decide when to become a reflector. In other words, if the threshold value is set high, the nodes would have to transfer more flow loads before triggering the self-appointment selection process and hence would inject more redundant streaming traffic in the network.

Regarding the number of locally exchanged messages, we focused our study to the results in Figure 5 and Figure 6 while comparing these against the worst case scenario. We are considering that under the worst case scenario all nodes send messages to all their neighbors at the beginning of a cycle. Each bar of the graphics represents a mean of all simulations made for all experimented topologies, with its respective factors and levels listed in the legend.

With a 10 clients per tick arrival rate one note that, for fixed cycle value, when the threshold value increases, the mean of messages per tick decreases. See that this occurs for all cycles (5, 7, 10, and 15 ticks). As mentioned earlier, this means that the threshold causes a direct impact over the number of

messages exchanged. This result was expected since it represents the amount of time to take a decision once message exchange is started by some node.

However, when the arrival rate is 20 clients per tick we can see that the threshold value has a low impact on the amount of exchanged messages per tick. This occurs because when arrival rate is 20 clients per tick, we have more active video flows in the network, consequently nodes will reach the defined threshold value quickly. Anyway, it is clear that the cycle length influences the number of messages.

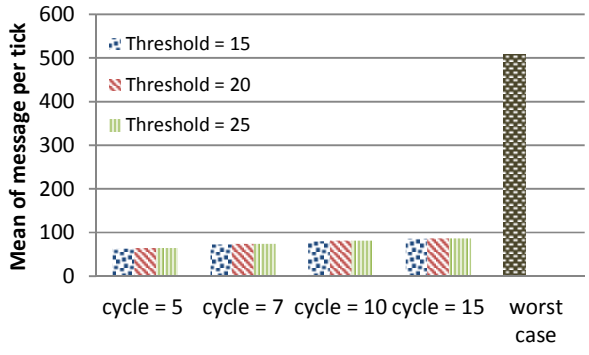


Fig. 5. Mean of messages for arrival rate 10.

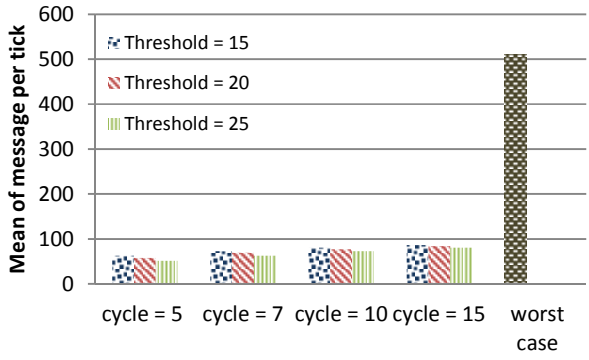


Fig. 6. Mean of messages for arrival rate 20.

We note that there is a variation when the cycle length is increased. This increase is expected since the cycle length represents how long nodes have to wait to take the decision to become a reflector, i.e., the higher the cycle length, the higher the time to initiate the message exchange phase and, consequently, the higher is the number of messages exchanged in the system. For all values of cycle and threshold, the number of messages is always much better than the worst case.

4.2 Cogent Topology

Cogent[2] is a multinational Tier 1 Internet Service Provider ranked, according to its website, as one of the top five networks in the world. Its network covers 37 countries in North America, Europe, and Asia. In this scenario, we use a version of the 2010's Cogent topology available at the Topology Zoo site[3]. Such topology covered only America and Europe and it consisted of 197 connection points and 243 links. Such representation was treated to remove duplicated links and redundant paths using out proposed self-appointment technique.

We located the server in a node with just one link, and located edge nodes by using the same rule, totalizing 22 edge nodes. Since a server could be located at any of the different nodes, we repeated the experiment 50 times to obtain statistical confidence. Other parameters and levels were maintained as in the last scenario. However, the factors were adjusted and considering the results, their configuration followed Table IV: the threshold is 20, the cycle length is 5 ticks, and the arrival rate is set to 20 clients per tick.

The stop condition was calculated based on the statistical error of 5% on our two metrics: the average consumed bandwidth and the average number of messages exchanged between nodes (in the same way as in the first set of simulations); or 10000 ticks of simulation.

[2] http://www.cogentco.com/
[3] http://www.topology-zoo.org/

Table IV. Parameters and Levels For Cogent Topology

PARAMETER	LEVEL
Number of nodes	197
Number of edge nodes	22
Number of video streaming server	1
Cost between links	1
Mean of video streaming duration (exponential distribution)	5 ticks
Threshold	20 flows
Cycle	5 ticks
Arrival rate (Poisson distribution)	20 clients per tick

For comparison, this time we created a static solution (called **static reflectors**), in which reflectors are previously located in the Cogent topology since the creation of the network and remain active throughout the simulation lifetime with no new reflectors being spawn.

ALGORITHM 1: Static Reflectors Solution Algorithm

input: *topology*
output: *reflector*

```
foreach node in topology
    if (node.isEdge) do
        reflector.add(node)
    else if (node.amountLinks – 1 > threshold) do
        reflector.add(node)
    end
end
```

This algorithm uses a centralized strategy to allocate reflectors in static way by in an initial scenario. The rationale behind this algorithm is that reflectors created near of edge nodes can minimize bandwidth consumption for more links in the path between reflectors and the video server. Furthermore, the algorithm also considers that nodes very connected may become a bottleneck and should be alleviated in preference to others. Please recall that this solution is not adaptive.

Figure 7 shows the mean of consumed bandwidth representing both the total of consumed network bandwidth and the one occupied in the core of the network while disregarding the edge links. RBSA4LS successfully decreases the consumed bandwidth by as much as 85% when compared with the scenario without the presence of RBSA4LS reflectors. Note that even the static solution decreases resource occupation by as much as 77% when compared with the baseline configuration.

Since the static reflectors solution do not engage in the exchange of messages by definition, we show no comparisons here but the mean of messages per tick when RBSA4LS was used in the Cogent topology reached 89.43 message per tick.

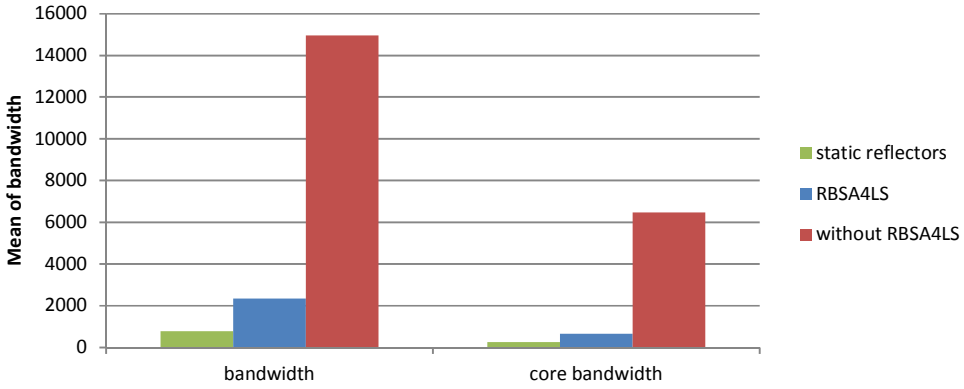


Fig. 7. Mean of bandwidth in Cogent topology.

In this last scenario, besides observing bandwidth and messages exchange, we introduced a new metric: that of the number of reflectors present by cycle. The goal of this metric is to evaluate the number of nodes performing the reflector role per cycle compared to the static case. This metric is important because reflectors represent a cost (hardware and software) and it could turn the solution impracticable due to the total cost. Figure 8 represents the mean of such number per simulation (RBSA4LS and static reflectors solution) as well as the maximum and the minimum number of reflectors per simulation using RBSA4LS.

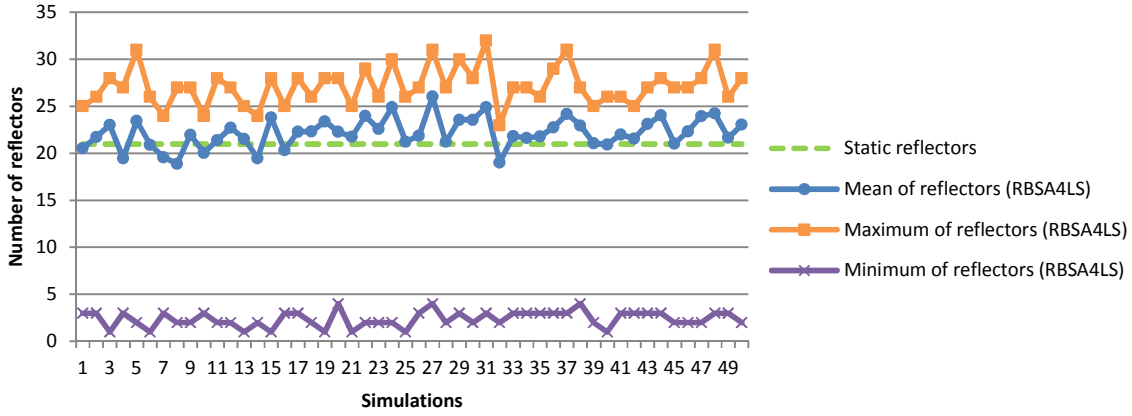


Fig. 8. Mean of reflectors per cycle in Cogent topology.

5. DISCUSSION

RBSA has many configurable parameters such as the cycle value it defines and the threshold condition continuously tested. We analyzed through simulations the impact of each one on our observable output metrics, namely, bandwidth and additional message exchange overhead.

When we increased the cycle length value, we noted an increase in the number of exchanged messages. As mentioned earlier, this behavior is expected since the cycle length represents the time in which RBSA4LS exchanges messages to take a decision. The results also showed that a low threshold has a significant impact on consumed bandwidth. A low threshold reduces the consumed bandwidth

because the entire system can react earlier and waiting a long time to introduce a new reflector into the network.

In bandwidth consumption, a low threshold will activate the message exchange earlier and it will promote the reduction in bandwidth consumption earlier. In the other hand, with a low threshold, message exchange will be activated more times hence increasing their overhead.

In a real topology, we compared the RBSA4LS with a scenario where static reflectors were pre-configured. Despite the algorithms bandwidth gain was lower as opposed to the gain against the scenario without RBSA4LS, the mean number of simultaneous reflectors per cycle for RBSA4LS remained close to that of the static case.

These results point the trade-off between centralized and autonomic distributed solutions. Centralized solutions can have a better performance, but require knowledge about all the elements of the network. Sometimes, the decision time is crucial and it is difficult to collect such information of all elements in the system. Anyway, the important key concept advocated by RBSA4LS is role creation, and we can show that it provides a good performance close to the static solution.

Finally, the technical feasibility for RBSA4LS implementation is currently provided by technologies such as Network Virtualization (NV) and Software Defined Networking (SDN) due to its capacity of managing flows over network. Using these approaches one can construct prototypes for future testing with RBSA4LS.

6. RELATED WORKS

According to [Sharma et al. 2003], multicast is a key enabler to transfer high-bandwidth multimedia broadcasts and seminars on IP networks, since multicast promotes reduction of network traffic and video server load. However, there are factors that contribute to turn IP-layer multicast into an unfeasible solution for live streaming. For instance, its commercial deployment has been very limited, mainly because in order to support IP multicast all components in the infrastructure (video source servers, video client devices, and intermediate routers) should be made multicast-aware. Thus the video clients and servers must have support to IP multicast reception in the TCP/IP protocol stack, and the network routers must be able to build packet distribution trees that allow sources to send packets to all clients [Cisco 2013]. Authors in [Rajkumar and Swaminathan 2013] also say that IP multicast and Application Layer multicast create duplicate packets which deteriorates the redundant network traffic.

Another technical aspect contributing to the unfeasibility of IP-layer multicast is that the traditional multicast routing protocols, such as Distance Vector Multicast Routing Protocol (DVMRP), Multicast Open Shortest Path first (MOSPF), Protocol Independent Multicast Dense-Mode (PIM-DM), and Protocol Independent Multicast Sparse-Mode (PIM-SM) suffer from scaling problems [Handley and Crowcroft 1999]: DVMRP and PIM-DM initially send data everywhere and also require routers out of the distribution tree to hold prune state to prevent this flooding from persisting; MOSPF requires all routers to know where all receivers are located; and PIM-SM needs pre-distribution of information about the set of core routers called Rendezvous Point (RP). Since traffic needs to flow to the RP, a RP cannot handle too many groups simultaneously; therefore many RPs are needed globally.

Despite the existence of academic works having proposed infrastructure dedicated for the support of live streaming using CDNs (such as [Zhuang and Guo 2011] and [Li et al. 2013]), none of them is focused on a Cloud virtualized infrastructure. Furthermore, there are proposals for the usage of centralized strategies for creating overlay multicast with the surrogates or servers that are part of the CDN infrastructure. Our main contribution in this paper was the proposal of an autonomic management of live streaming application over DCloud infrastructure, focusing on minimizing redundant traffic and on creating of reflectors

7. FINALCONSIDERATIONS AND FUTURE WORKS

In this paper, we presented RBSA4LS an autonomic strategy to manage creation of reflectors for reducing redundant traffic in a live streaming application over Distributed Cloud infrastructures. Such strategy is based on the RBSA framework that provides self-organizing solutions for node autonomic management in a network.

In RBSA4LS the network nodes continually sense the network and check forwarding packets to assess if the live streaming flows of a content source have achieved a high utilization level. If so, the nodes communicate and appoint a new reflector node, which will transparently start to multicast video flows to clients while alleviating the network links that goes through the content servers and reducing the redundant flows in the network.

We evaluated RBSA4LS through extensive simulations and the results showed that such simple strategy can reduce 40% of the redundant traffic in random topologies with 50 nodes. We also simulated RBSA4LS in a large ISP topology obtaining a reduction of 85% in the redundant traffic, which is close to the best result obtained by a static strategy that positions reflectors in the border of the network. Nevertheless, RBSA4LS autonomously appoints reflectors where they are needed and adapts such roles according to viewers' fluctuations.

As future work we are planning to take in consideration the cost of nodes and the cost of changing roles. We would like to study when reflector´s should cease to exist as streaming traffic diminishes in the network as well as their life migration impact. We are also prototyping the RBSA4LS in a physical and virtual testbed in order to verify the applicability of the proposal in a real environment. Moreover, we intend to apply the RBSA framework in other contexts, such as autonomic caching in CDNs.

REFERENCES

Global Internet Phenomena Report, 2013. Available at https://www.sandvine.com/downloads/general/global-internet-phenomena/2013/2h-2013-global-internet-phenomena-report.pdf. Last access in January, 2014.

K. Michel. "Live Video Streaming that Can Handle Traffic Spikes - The Challenge". Available at https://blogs.akamai.com/2013/01/live-video-streaming-that-can-handle-traffic-spikes-the-challenge.html. Last access in January, 2014.

Z. Zhuang, and C. Guo. "Optimizing CDN infrastructure for live streaming with constrained server chaining", in IEEE ISPA, 2011.

K. Rajkumar, and P. Swaminathan. "Eliminating Redundant Link Traffic for Live Multimedia Data over Distributed System", in International Journal of Engineering and Technology (IJET), vol. 5, pp. 1202-1206, 2013.

E. Nygren, K. Sitaraman, and Jennifer Sun. "The akamai network: a platform for high-performance internet applications", in ACM Operating Systems Review (SIGOPS), p. 2-19, 2010.

G. Gonçalves, P. T. Endo, A. Palhares, M. Santos, J, Kelner, and D. Sadok. "On the load balancing of virtual networks in Distributed Clouds", in ACM SAC, pp. 625-631, 2012.

K. Church, A. Greenbreg, and J. Hamilton. "On Delivering Embarrassingly Distributed Cloud Services". In V Workshop on Hot Topics in Networks (HotNets), Citeseer, 2008.

M. Alicherry, and T. V. Lakshman. "Network Aware Resource Allocation in Distributed Clouds". In IEEE INFOCOM, pp. 963-971, 2012.

P. T. Endo, A. V. A. Palhares, N. N. Pereira, G. E. Gonçalves, D. Sadok, J. Kelner, B. Melander, J. E. Mangs, "Resource allocation for distributed cloud: concepts and research challenges". In IEEE Network Magazine, vol. 25, pp. 42-46, July 2011.

V. Valancius, N. Laoutaris, L. Massoulie, C. Diot and P. Rodriguez. "Greening the Internet with Nano Data Centers". In International Conference on Emerging Networking Experiments and Technologies, pp. 37-48, 2009.

P. Endo, A. Palhares, M. Santos, G. Gonçalves, D. Sadok, J. Kelner, A. Sefidcon, and W Fetahi. "Role-Based Self-Appointment for Autonomic Management of Resources", in IEEE NetMM, 2014.

Lewis, T. G. Network Science: Theory and Applications, Wiley, 2009.

P. Sharma, Edward Perry, and Radhika Malpani. "IP multicast operational network management: design, challenges, and experiences", in IEEE Network, p. 49-55, 2003.

Cisco Systems, IP Multicast Deployment Fundamentals, Design Implementation Guide, 1999. Available at http://www.cisco.com/en/US/tech/tk828/tech_brief09186a00800e9952.html. Last access in May, 2013.

M. Handley, and J. Crowcroft, "Internet Multicast Today", in The Internet Protocol Journal, v. 2, n. 4, 1999.

B. Li, Z. Wang, J. Liu, and W. Zhu. "Two decades of internet video streaming: A retrospective view." ACM Transactions on Multimedia Computing, Communications, and Applications, pp. 33:1 – 33:20. 2013.