mCast: An SDN-based Resource-Efficient Live Video Streaming Architecture with ISP-CDN Collaboration

Ahmed Khalid
University College Cork
Cork, Ireland
Email: a.khalid@cs.ucc.ie

Ahmed H. Zahran
University College Cork
Cork, Ireland
Email: a.zahran@cs.ucc.ie

Cormac J. Sreenan
University College Cork
Cork, Ireland
Email: cjs@cs.ucc.ie

***Abstract*—The rise of Software Defined Networking (SDN) presents an opportunity to overcome the limitations of rigid and static traditional Internet architecture and provide services like network layer multicast for live video streaming. In this paper we propose mCast, an SDN-based architecture for live streaming, to reduce the utilization of network and system resources for both Internet Service Providers (ISP) and Content Delivery Networks (CDN) by using multicast over the Internet. We propose a communication framework between ISPs and CDNs to enable mCast while retaining user and data privacy. mCast is transparent to the clients and maintains the control of CDNs on user sessions. We developed a testbed and performed large scale evaluation and comparison. Results showed that mCast can improve the video quality received by clients and, for CDNs and ISPs in comparison to IP unicast, mCast can decrease link utilization by more than 50% and network losses to 0%.**

I. Introduction

The proliferation of demand[1] for high definition (HD), Internet-based video streaming has resulted in frequent short-lived events over the Internet, such as sports events, political events and crowd-source streaming. Additionally latest social trends have seen an extreme rise in individuals sharing live videos with groups of audience located across the globe. Services like Facebook Live[2] and Twitch[3] have changed the norms of live streaming where, Twitch alone saw a peak of over 2 million concurrent viewers in 2015. Consequently, a very dynamic and flexible provisioning of network and system resources has become a challenge.

Content providers usually have limited resources and they rely on Content Delivery Networks (CDN), to distribute streams globally. More than 60% of video traffic on Internet passes through CDNs[4]. CDNs use IP unicast to deliver streams to video clients. With IP unicast, the same stream is sent, potentially, thousands of times in parallel to clients located inside an Internet Service Provider's (ISP) network. This results in a wastage of system and network resources for both the CDN and the ISP.

[1]https://goo.gl/Gpd56l
[2]https://live.fb.com/
[3]https://www.twitch.tv/year/2015
[4]https://goo.gl/ySYurJ

To save resources, instead of IP unicast, a CDN can use either IP multicast or Application layer multicast (ALM) to distribute live streams. In multicast a stream is sent to a group of clients simultaneously in a single transmission, reducing the consumption of resources. However, multicast is rarely used in the Internet due to its rigid and static nature and other limitations specific for the aforementioned multicast approaches [1], [2].

Software-Defined Networking (SDN) is an emerging approach for network programmability, with the capacity to initialize, control, change, and manage network behavior dynamically via open interfaces [3]. A logically centralized controller, with a global view of network, can monitor every traffic flow, make forwarding decisions and install efficient rules at runtime. An SDN-based network possesses characteristics that are needed to deploy multicast over the Internet and are non-existent in a traditional IP network. The knowledge and awareness of network nodes and clients by the controller in SDN can be used to construct resource efficient trees in an ISP. The flexibility provided by SDN at network layer and the ability for inter-controller communication can be utilized to develop a multicast service for inter-domain video traffic coming from a CDN to an ISP network.

In this paper we propose mCast, a novel architecture for live streaming that merges the flexibility and control of SDN with resource efficiency of multicast to reduce inter-domain and intra-domain traffic for both ISPs and CDNs. Our modular architecture reduces the cost and complexity to implement network layer multicast for ISPs and provides a dynamic and scalable mechanism for multicast tree construction in real time. CDNs have full control over their clients and they get all the information necessary for management and billing.

The clients do not need to be modified because mCast installs rules on the last hop to convert the stream back to IP unicast, making the delivery transparent. These rules also help to identify the amount of traffic that goes to each user. The ISP can use this information to charge users based on their own data plans. As such, mCast solves one of the main problem of IP Multicast i.e. inability to bill users in a multicast group based on their individual billing plans.

mCast employs agents in both the ISP and the CDN. These agents are responsible for communication with each other as well as the control plane of SDN in their respective domain. A CDN can choose to switch from IP unicast to mCast when doing so will reduce its overall cost. We propose a decision model to help CDNs in making this decision. As ISPs are economically driven and will charge CDNs for availing mCast service, it is important for a CDN to know the exact cost to serve a certain video stream.

Our decision model not only identifies various cost factors but also presents them in quantifiable mathematical equations and if a CDN chooses to use mCast, this model will help the CDN to decide when switching to mCast will reduce the total cost to serve a stream. Once the decision has been made, CDN mCast Agent provides ISP mCast Agent with a list of its clients that are watching the stream and can be served with mCast. An ISP uses this information to construct a dynamic multicast tree using our extension of Dijkstra's Algorithm and installs forwarding rules in runtime.

For CDNs and ISPs, keeping their network infrastructure private is of utmost importance and they tend not to reveal information such as network topology, available bandwidth, or routing paths. We designed our framework in a way that no such information needs to be disclosed. Both ISPs and CDNs can manage their clients and network in their own way and just share the identity of clients to be served with mCast. This resolves any concern that a CDN might have in terms of user or data privacy.

For evaluation we developed a large scale testbed with video servers streaming real HD video content. We performed extensive evaluation to show the feasibility, scalability, robustness and gains of mCast. We compared mCast with standard IP unicast and results showed that in similar network conditions mCast, not only can save significant network and system resources for ISPs and CDNs, but also delivers a better quality of video to clients with lesser start-up delays and no dropped packets.

The rest of this paper is structured as follows. In Section 2, we discuss the existing literature and related work. In Section 3, we present the proposed architecture and decision model. In Section 4, we present our testbed and evaluation results and we conclude the paper in Section 5.

II. Literature Review

IP multicast does not provide CDNs with enough control over clients because it lacks support for features [1] such as group management, authorization, billing policies, data privacy and security. These features are of crucial importance to the business and financial model of CDNs and hence for live streaming, CDNs do not adopt IP multicast. Also, ISPs have well managed topologies with carefully planned routing policies [4] and they avoid inter-domain IP multicast traffic which is unpredictable and difficult to handle. Due to all of its limitations, the use of IP multicast to deliver video has been restricted to IPTV services within a single domain, where an ISP can pre-configure the network with static routes to paying customers.

Another approach to reduce resource consumption is overlay multicast, also known as ALM. End devices organize themselves into an overlay topology and distribute the stream in an efficient manner. Multicast is handled by end nodes rather than routers. Although, ALM is not the ideal choice for CDNs, as it propagates slowly and usually incurs additional latencies [2], active involvement of end users does reduce the system load for servers in CDN or content provider. On the other hand, the situation further deteriorates for ISPs as additional unicast flows are introduced by clients of ISPs to facilitate ALM, consuming even more bandwidth than IP unicast. These problems, along with the rising demand for HD streaming, serves as an incentive, for CDNs and ISPs alike, to come up with a better and more resource efficient system for delivering live streams.

Over the past few years, research has been conducted to improve the mechanism of IP multicast using SDN. For example in [5], an SDN-based system is proposed that allows fast failure recovery for IP multicast trees. In [6] an approach is presented where a centralized SDN controller manages IP multicast. All IGMP messages are sent to the controller that calculates multicast groups and configures network accordingly. While such solutions improve the management of IP multicast, other issues such as handling inter-domain traffic remain unaddressed. In Lcast [7], a network-layer inter-domain multicast framework, is proposed that creates a router overlay to connect multicast hosts in different domains. Solutions, like Lcast, enable inter-domain multicast but fail to present CDNs with enough control over clients to provide a viable live streaming service.

A few SDN-based network layer multicast frameworks have been proposed to offer content and/or network providers with sufficient control to realize a video streaming service. In [8], SDM is proposed to enable ISPs to support resource efficient peer-to-peer streaming. A virtual peer is created inside an ISP allowing an external streaming source to gain a presence. ISP then distributes traffic to its clients using NAT-like rules. Other works exploit Scalable Video Coding (SVC) [9] and Multiple Description Coding [10] to divide a video stream in separate flows and multicast groups and ensure a minimum quality for users. These works give a very good idea of what SDN can do to create dynamic networks, save network resources and improve video quality. Our proposed architecture bears some resemblance to SDM, with a key difference that we provide a framework between ISPs and CDNs that can be used to develop live streaming services with multiple video channels and features like channel switching.

Other works propose some elaborate multicast routing algorithms [11], [12] that can be used in SDN. SDN controller can receive network statistics from all the network nodes and can use this global information to construct efficient paths. These SDN-based routing algorithms are complimentary to our work and can be implemented in our architecture to obtain optimized routing paths and construct optimal multicast trees based on

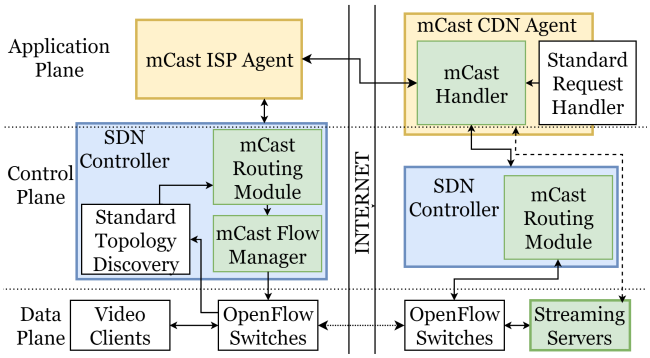

Fig. 1. Architecture of mCast.

the requirements of the underlying network.

III. Proposed Architecture

Our proposed architecture for live streaming focuses on two main goals: reducing the resource utilization in ISPs and CDNs, and providing CDNs with full control over their clients even when using multicast.

We consider that both ISP and CDN are SDN-enabled. The growth of SDN in market and industry is on the rise. SDN is already transporting 23 percent of traffic in data centers, growing to 44 percent by 2020 [13]. A 2016 survey indicates that 75% of the respondents had either implemented SDN in their network or were planning to do so[5]. Generally, we believe that ISPs and CDNs would implement SDN for various use cases such as efficient resource utilization and ease of management.

In our architecture, we exploit flexible data forwarding and mangling capabilities of SDN. This is essential for an ISP to take advantage of the routing capabilities of SDN. For a CDN, the SDN controller can be replaced by a server, capable of communicating with an ISP and managing clients on its own. However we used SDN for the CDN to show how two controllers can communicate and manage their underlying networks more efficiently.

A. Architecture Overview

Figure 1 illustrates key mCast architectural elements in application, control, and data planes[6]. At the data plane, SDN switching nodes are used to forward user data within ISP and CDN networks, according to mCast control plane functions. mCast employs agents in both ISP and CDN networks to perform application plane functions. The role of these architectural elements is detailed below.

mCast CDN Agent: This component has three main functions: monitor the client requests, classify clients and trigger mCast. In a standard IP unicast streaming service, a request handler receives channel requests, authenticates clients and responds with the IP and port address of the streaming server. A client can then send a content request to the streaming server.

mCast CDN Agent extends a standard request-handler in traditional CDN systems to enable efficient content delivery using multicast. A standard request handler authenticates content requests and forwards legitimate content requests to their corresponding severs according to the CDN policy. mCast extensions include request classifier and multicast management functions. The request classifier identifies users located in a single ISP while watching the same content. Such classification can be performed using geo-location databases. Once a group of clients satisfy a pre-specified criterion (discussed in Section 3D), the request classifier triggers mCast handler to start multicast operation.

The multicast management functions include interfacing with: mCast ISP agent to request mCast service and share client details; mCast CDN routing module to install routing entries for multicast and; mCast streaming server to improve the resource utilization by aggregating a group of flows into a single flow as detailed below.

mCast ISP Agent: ISP plays a passive role in mCast, as in, it does not trigger the mCast request. To be the trigger, an ISP would need to perform deep packet inspection and decoding to identify what streams are watched by clients and whether they are served from the same CDN. As CDNs are a better judge of their clients, it is simpler and less resource consuming if a CDN triggers mCast request.

This component represents the application plane module at the ISP and holds key importance in mCast architecture. It performs two main functions, interfacing with CDN and orchestrating multicast operations in the ISP. It receives session aggregation requests from CDN including source and destination addresses and port numbers for every user session intended for multicast. Note that such information is used to identify the target users in the ISP network and ensure transparency for end clients.

Upon reception of such requests from CDN, mCast ISP agent first creates an identifier for the mCast stream composed of an IP address and a port number, denoted as $V_{(IP,\ Port)}$. Then, it instructs the mCast ISP routing module to create a multicast tree for the intended users starting at the ISP gateway. Once a tree is successfully constructed, ISP Agent shares $V_{(IP,\ Port)}$ with CDN Agent which uses it to configure CDN network for mCast delivery.

As users join or leave the session, mCast CDN agent informs the ISP agent to dynamically update routing entries in the ISP switching nodes. Such interaction is not possible in traditional switching nodes due to the lack of a centralized control as in SDN.

mCast Enabled Streaming Server: This server typically listens for content requests and streams the requested content using RTP/UDP/IP unicast sessions. To support mCast, the server implements an API to communicate with mCast CDN agent. Over such an interface, the server would be instructed to pause transmission for a group of sessions and alternatively send the content to the provided destination address and

[5]https://goo.gl/CEyzBs

[6]https://www.opennetworking.org/sdn-resources/sdn-definition

port $V_{(IP, Port)}$. To avoid packet losses, the new connection is activated before terminating the old unicast sessions. While this sequence ensures no packet loss, it may lead to duplicate packets at the client. However these packets would be ignored by the client after inspecting the RTP header.

mCast CDN Routing Module: The routing module is programmed to forward content requests to the request handler for authentication. Once authenticated, the routing module is provided a target server to forward connection requests using its predefined routing policy (e.g., shortest path or least loaded path). In the presence of mCast agent, it is consulted before proceeding with the default routing. If the multicast criteria is satisfied, multicast routing is activated and the new content request is served through the new multicast stream.

mCast ISP Routing Module: This module identifies the routes of different flows from the ISP gateway to their end points. Due to the global view of SDN controller, a typical topology manager in SDN is aware of all of its network nodes and clients. mCast ISP Routing module probes the topology manger to obtain a graph representation for ISP network. It also communicates the estimated routes to the flow manager that interfaces with the switching nodes. The ISP routing module implements the ISP routing policy and is extended by an additional function to support multicast routing, which is triggered by the mCast ISP agent. The mCast ISP agent also dynamically instructs the routing module to update video multicast trees as clients leave and join.

Although the main focus of the paper is to present the architecture and components essential for realizing a live streaming service using inter-domain multicast, we developed an extension of Dijkstra's algorithm to implement on the mCast ISP Routing Module. As the architecture of mCast is modular, to implement any other tree construction algorithm, mCast Routing Module can be replaced with that algorithm without modifying rest of the modules in the architecture. As part of our future work we plan to look specifically on the tree construction algorithms and either propose or identify from existing work [11], [12], an optimal multicast tree construction algorithm.

In our extension of Dijkstra's algorithm, When the mCast ISP Routing Module receives a request from mCast ISP Agent, it calculates the shortest path for the first client. Then it sets the weight of all involved edges to zero before calculating the shortest path for the next client. This prioritizes the used edges and paths over others and reduces link stress in the network. The process is repeated until a path is determined for all the clients. This information is then passed on to the mCast Flow Manager.

mCast Flow Manager: A Flow Manager in an ISP installs rules on SDN-enabled switches based on the information that it receives from the Routing Module. mCast Flow Manager installs multicast entries in network nodes with higher priority than IP unicast, ensuring that clients are served with mCast whenever possible. In addition, mCast Flow Manager installs transparency rules on the egress switch. Before forwarding a packet to the client, this rule modifies the $V_{(IP, Port)}$ to the IP and Port address of the video client, so the client receives the packet just as it would in IP unicast, hence the transparent delivery. These rules also help the ISP to identify the amount of traffic that goes to each user. An ISP can use this information to charge users based on their individual billing plans.

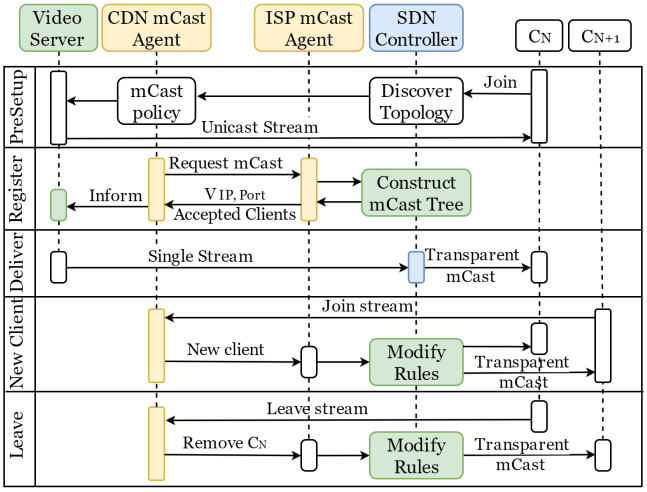

Fig. 2. Important message exchanges to establish mCast.

B. Functional Description

Figure. 2 illustrates the exchanged message-sequence between different components to setup mCast. In this section, we explain these messages and give an overview of mCast operations. For simplicity of illustration, we consider that all the requests belong to a single ISP. The same operations would be applied to clients belonging to a different ISP.

Pre-setup: We use our decision model (Section 3D) as the criteria to initiate mCast. C_N represents the client that satisfies the decision model. Any client that joins the stream before C_N, is served with IP unicast. For these clients, the mCast CDN routing module installs unicast rules for the client and forwards the request to the streaming server which sends a unicast stream to the client.

When a content request is received from C_N, along with sending a unicast stream to C_N, CDN requests mCast service from ISP and sends a list of clients to be served with mCast. The list includes full tuples i.e. server's address $S_{(IP, Port)}$ and the client's address $C_{(IP, Port)}$. ISP mCast Agent receives this request and assigns a virtual IP and port $V_{(IP, Port)}$ to the video stream. ISP can choose an address from a pool of IPv4 addresses that it reserves for mCast service. It can then use port numbers to distinguish streams, allowing up to 65535 streams per address.

Registration and Delivery: List of clients and $V_{(IP, Port)}$ are passed to the mCast ISP routing module. The mCast Flow Manager then installs rules on the switches to: forward the traffic coming for $V_{(IP, Port)}$ to the switches in mCast tree and; on egress switch replace $V_{(IP, Port)}$ with $C_{(IP, Port)}$ and the source with $S_{(IP, Port)}$.

The SDN controller then informs the CDN mCast Agent which sends a message to the streaming server to terminate IP unicast streams of accepted clients and replace them with a single stream destined for $V_{(IP, Port)}$. As the ISP network is all set up for mCast, when the single stream from the server enters the ISP ingress switch, it is sent only once on every link on the tree until it reaches all the clients.

New Client Requests: When the CDN receives a new content request from a client, instead of sending that request to the server, it is first sent to the ISP. The port number $S_{(IP, Port)}$ is set to the one at which the client made the request i.e. the listening port of the server. ISP adds this client to the mCast tree by installing or modifying forwarding rules and the client starts receiving the stream instantly and transparently.

Client Leave Requests: When a client decides to switch channel or terminate the service, it sends a leave request to the CDN. The CDN mCast agent receives this request, updates the client' state and also informs the ISP. The mCast Routing module in ISP updates the mCast tree by traversing backwards from the egress switch and removing mCast entries, until it reaches a node that serves more than one client. As clients joining and leaving can result in unoptimized multicast paths, a global update of mCast tree can be scheduled, to optimize the routing paths based on the currently joined clients.

C. Performance Analysis

We measure the performance of mCast with the consumed network and system resources. For IP unicast, the bandwidth consumed in the network increases linearly with every new client, regardless of the number of shared links in the network. mCast uses network layer multicast and avoids duplication of traffic over any link. As the number of clients increases in the network, the amount of bandwidth consumed at a link stays constant. This reduces the amount of bandwidth consumed when one or more shared links exist in the network. This also avoids bottleneck links when a large number of clients share a link, as the bandwidth consumption per link is independent of the number of clients in mCast.

The system resources consumed in an SDN network include the flow entries or rules installed at switches and the messages shared, between SDN controller and the switches, to install these flow entries. For a single live stream, mCast installs only one entry per switch. When a new client joins mCast, the SDN controller sends a message to the switches on the path to this client and further actions are added to the flow entry. Depending on the approach used by an ISP, mCast offers savings in system resource consumption, for example when an ISP implements service differentiation [14]. Instead of having one rule at entry and exit point of each tunnel for a live streaming service, mCast will install only one rule per switch in the network, saving the system resources.

Other ISPs will save system resources by mCast, as the number of packets to be processed by a switch will be very low in comparison to IP unicast and hence the processing cost and time will be reduced. This is reflected in our evaluation results, where we can see that for a large number of clients, when IP unicast overloads a switch's processing capability, mCast maintains a very low CPU consumption.

D. Decision Model for mCast

As ISPs are economically driven and will charge CDNs for availing mCast service, it is important for a CDN to know the exact cost to serve a certain video stream. In this section, we identify various cost factors and present an optimized mathematical cost model for the decision of switching from unicast to multicast for a specific stream. The model is a distinct complementary contribution. mCast does not rely on it however if a CDN chooses to use mCast, this model will help the CDN to decide when switching to mCast will reduce the total cost to serve that stream.

Two main factors that add up to the cost for a CDN serving a channel to N clients are: the load on servers or power consumption $P(N)$ and the outgoing traffic or bandwidth consumption $B(N)$. For $P(N)$, we use the Power model from [15] which states a linear increase in power consumption with the increasing number of clients. The maximum power consumed by a server when fully utilized is represented by$P_{\max}$. An idle server consumes approximately 70% of $P_{\max}$ while the remaining 30% increases linearly with the number of clients. From this model we derive our equation for the total power consumed by all the servers to serve N clients that are watching a particular channel. If one CDN server can support N_o number of clients then to serve N number of clients with IP unicast $P(N)$ will be

$$P(N) = P_{\max}\left(0.7\left\lceil\frac{N}{N_o}\right\rceil + 0.3\frac{N}{N_o}\right). \quad (1)$$

As an example, if there are 1200 clients (N) watching a channel and one server can support 500 clients (N_o) then we need three servers, where two servers are fully utilized and the third one consumes 70% of $P_{\max}$ for running idle and an additional 12% to serve 200 clients. The total power consumed can be found by substituting N and N_o in Equation 1, yielding $P(N) = 2.82P_{\max}$.

$B(N)$ is the Internet transit cost that a CDN will have to pay to the Internet Exchange Point (IXP) for serving a channel to N clients. Transit volume is the amount of traffic that goes out of a network domain. Internet transit is typically metered and priced in $/Mbps. The industry standard to measure transit cost is the 95th Percentile method. In this method, a network can avail of pricing discounts by relying on commit volume. Commit volume is a certain volume of traffic that network domains can agree in advance to pay for, regardless of the actual volume that they consume. Let V_T be the 95th percentile transit volume and V_C be the commit volume that the CDN committed to an IXP, [16] models the transit cost as

$$\text{Transit cost} = \max(V_T S_T, V_C S_C), \quad (2)$$

where S_C, the single unit price for commit volume is lower than S_T, the single unit price for transit volume. We use this model to calculate $B(N)$. Further details of this model can be found in [16].

We represent the volume consumed by N clients by $V(N)$ and hence $B(N) = \max(V(N)S_T, V(N)S_C)$. If a CDN chooses the first method i.e. transit volume then $V(N)$ increases linearly with the number of clients and can be calculated as $V(N) = \alpha N B_i$, where B_i is the average bit-rate of channel i and $\alpha > 1$ accommodates for the variable video bit-rate and helps avoiding large queuing delays. With $V_C = 0$, $B(N) = \alpha N B_i S_T$

If a CDN uses the second method i.e. commit volume to an IXP then the cost for V_C is a fixed value and does not vary with the number of active clients. To get an estimate of the cost for N clients watching a single channel, we divide the cost equally among all active clients. Let X_i be a client watching a channel i at the average bit-rate B_i, then $V(N) = NB_i / \sum X_i B_i$. This represents the portion of the cost for N clients. With $V_T = 0$, $B(N) = V_C S_C N B_i / \sum X_i B_i$. Combining these two results and substituting in Equation 2 gives us

$$B(N) = \max\left(\alpha N B_i S_T, \frac{V_C S_C N B_i}{\sum X_i B_i}\right). \quad (3)$$

In general, CDNs use the second method i.e. commit volume to an IXP as this more predictable. The total cost that a CDN will incur to stream a channel to N clients using IP unicast is denoted by $U(N)$ and equals to

$$U(N) = P(N) + B(N). \quad (4)$$

With $V_T = 0$, substituting Equations 1 and 3 in Equation 4 gives us

$$U(N) = P_{\max}\left(0.7\left\lceil\frac{N}{N_o}\right\rceil + 0.3\frac{N}{N_o}\right) + \frac{V_C S_C N B_i}{\sum X_i B_i}. \quad (5)$$

Now we calculate the cost to serve a single channel to N clients using mCast. For mCast, the outgoing traffic from a CDN server and the load on it is independent of the number of clients and is equal to the cost of one client i.e. $U(1)$. In addition, CDN will have to pay a certain charge to the ISP for availing mCast service. An ISP can charge CDN with either a variable-rate based on the number of clients that joined a particular stream or a flat-rate based on an estimated value. This charge provides an extra incentive for ISPs to use mCast because in addition to saving resources, an ISP can increase its revenue by using mCast. This charge should represent the cost that an ISP incurs to provide mCast service and is mainly based on forwarding data to the clients that are in the multicast tree.

Because mCast avoids packet duplication on any link in the ISP topology, once a link is added to the tree to serve a client, the cost to serve any additional clients over that link is essentially zero. This means that once all the links in the ISP topology have been traversed and added to the multicast tree, an ISP will incur no additional cost for the increasing number of clients. Probabilistically, such a situation occurs with just a fraction of the total number of clients that an ISP can serve. As mentioned in [17], an ISP that can serve 100,000 clients can have all of its links, added in the multicast tree, with just 500 randomly distributed clients. Hence, an ISP can charge CDN a geometrically decreasing cost for every client that joins the stream. Therefore the total cost $M(N)$ that a CDN will have to pay for N clients using mCast will be

$$M(N) = U(1) + C\frac{1 - r^N}{1 - r}, \quad (6)$$

where C is an initial cost that an ISP charges CDN and r is the ratio for decreasing cost of every new client. Note that the small increment for every new client, regardless of no increase in the link cost, is justified by other minor costs that an ISP incurs such as number of forwarding entries and actions in the network nodes; managing clients and multicast trees; and interacting with the CDN.

A business model of a CDN for live streaming involves individual clients where serving each client incurs some cost on the CDN as discussed above. To minimize the total cost for the duration of the stream, a CDN should switch to mCast when the cost for mCast becomes lower than IP unicast. i.e.

$$M(N) < U(N). \quad (7)$$

For stability, only those clients should be considered for initiating mCast that have been watching a particular channel for a certain amount of time. The clients with very dynamic behavior such as the ones which are switching channels should be ignored. This will also keep the cost to minimum when an ISP is charging CDN with variable-rate based on the number of clients that joined a particular stream.

IV. Evaluation and Results

Our performance evaluation is based on real-testbed implementation. The main goal of the testbed is to show the feasibility, scalability, robustness and gains of mCast. While comparison with IP unicast is straightforward, we evaluate IP unicast to set a benchmark for various performance metrics. We first present our testbed setup and then present our performance evaluation results.

A. Experimental Setup

Figure. 3 illustrates our testbed setup in Mininet[7]. We consider two separate SDN-based domains, for ISP and CDN, connected over a high speed link. For the CDN, we used a tree topology with a depth and fan-out equal to one. For the ISP domain, we used two real residential ISP topologies from Topology Zoo database[8] including KREONET (approximately a STAR topology) and AT&T network (a MESH topology), to show that the potential of mCast is independent of the underlying network. Since topology zoo information does not include link rate, we set the internal link bandwidth to 200 Mbps.

Each domain is managed by a separate python-based Ryu controller[9] over which control and application planes are implemented. We considered OpenFlow 1.3 protocol for the SDN southbound interface. We implemented mCast CDN

[7]http://mininet.org/

[8]http://www.topology-zoo.org/

[9]https://osrg.github.io/ryu/

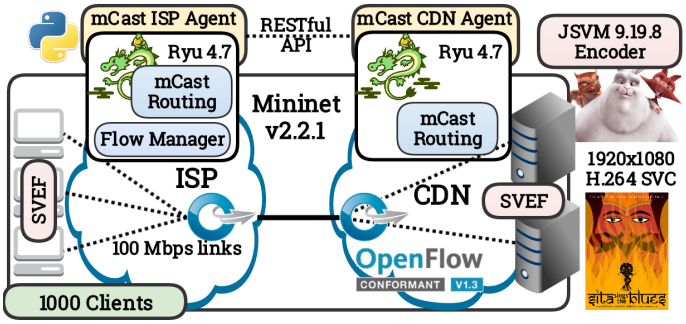

Fig. 3. Experimental Setup

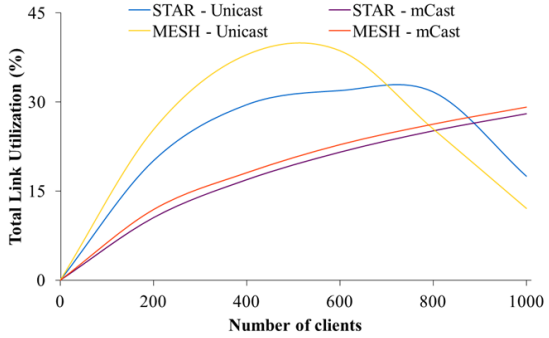

Fig. 4. Link Utilization (%) vs Number of clients

Agent as a separate module running over the CDN controller. We implemented mCast ISP Agent as a web server using a RESTful API. CDN can communicate with ISP using HTTP requests that are received by the RESTful API in mCast ISP Agent and passed to the SDN controller of ISP using Web Server Gateway Interface (WSGI).

We used two open source videos "Big Buck Bunny" (*bbb*) and "Sita Sings the Blue" (*sstb*) to be streamed from the CDN. We encoded the raw videos using JSVM framework[10]. We encoded nine minutes of each video at 1920x1080 HD resolution with a Group of Picture (GOP) size of eight at a frame rate of 25 fps, yielding an approximate bitrate of 2 Mbps.

We used C++ implementation of video client and server from the scalable video evaluation framework (SVEF)[11]. However, we modified the server implementation to act as a live streaming server that is capable of dynamically streaming the content to more than one client as it receives content requests. Additionally, we implemented an API for the interaction between content servers and the mCast CDN agent. We took advantage of the simple client implementation, that postpones video decoding to a post processing phase, to increase the number of clients to 1000. Each client was randomly attached to one of the ISP switches and requested one of the two available streams according to a uniform distribution at a random time.

Our key performance metrics include link utilization and dropped network packets. We also capture the percentage of decodable frames and start-up delays as an application layer metric. To analyze the cost of using mCast, we measured the number of additional Open-Flow rules and messages generated when using mCast instead of IP unicast. The shown results represents the average of five runs of experiments.

B. Results

We evaluated our architecture with extensive tests and calculated various network and application metrics to show the benefits of our proposed architecture and the cost to achieve it.

[10]https://goo.gl/fti0bu

[11]http://svef.netgroup.uniroma2.it/

1) Dropped network packets and video frames: In Table I we present results for STAR topology and in Table II we show the results for MESH topology. The results can be understood better by splitting them in two parts: when the system is not overloaded i.e. less than 600 clients and when it gets overloaded i.e. more than 600 clients. In the first case, IP unicast resulted in congestion in the network due to high bandwidth consumption with increasing number of clients. mCast reduced the consumed bandwidth by avoiding packet duplication and hence avoided congestion. This can be seen with zero packet loss when using mCast.

For more than 600 clients, the network switches and streaming servers became overloaded and a lot clients failed to connect to the server. These failures were due to overloaded Mininet and Open vSwitch as shown by the CPU Utilization in Table I and Table II . Real network switches have far more capacity than Mininet but the number of clients are also far higher. This testing scenario is shown to represent situations where the number of clients is high enough to surpass system resources.

In case of mCast, the load on network switches decreased significantly due to no packet duplication. Similarly the load on servers reduced, as only one stream was transmitted for one video channel regardless of the number of clients. Consequently all the client requests and traffic was handled perfectly with all the clients connecting to the servers and no packet loss in the network. This shows the robustness of mCast when the system resources are limited.

Similar trends can be seen with video packets at the application layer of the receiving clients. We calculated the total number of video frames dropped by all the clients throughout the streaming duration. The frames that were received by a client but had errors or lost dependencies were also considered dropped. These packets are actual representative of the number of frames that are not decodable and will result in a decreased quality for the user. As the results show, mCast provides a better video to users by avoiding congestion in the network and eliminating the dropped video packets and un-decodable frames.

2) Link utilization and bandwidth consumption: We determine link utilization as the percentage of link capacity of all

TABLE I
RESULTS OF EXPERIMENTS FOR STAR TOPOLOGY

Clients	Failed Client Connections		Network Packet Loss (%)		Video Frame Loss (%)		Open vSwitch CPU Util. (%)	
	Unicast	mCast	Unicast	mCast	Unicast	mCast	Unicast	mCast
200	0	0	0.33	0	1.22	0	16.67	4.92
400	0	0	3.41	0	5.98	0	31.42	6.68
600	56	0	23.56	0	38.78	0	68.14	10.15
800	470	0	1.33	0	2.52	0	86.45	11.88
1000	844	0	0.64	0	1.75	0	96.58	14.72

TABLE II
RESULTS OF EXPERIMENTS FOR MESH TOPOLOGY

Clients	Failed Client Connections		Network Packet Loss (%)		Video Frame Loss (%)		Open vSwitch CPU Util. (%)	
	Unicast	mCast	Unicast	mCast	Unicast	mCast	Unicast	mCast
200	0	0	0.37	0	1.38	0	15.93	6.24
400	0	0	5.62	0	9.10	0	31.51	9.55
600	191	0	13.02	0	22.91	0	72.91	12.04
800	677	0	5.61	0	9.10	0	94.11	13.94
1000	938	0	1.15	0	3.88	0	99.63	18.17

the links in an ISP network, utilized over a given amount of time. We measured link utilization against the number of clients that were actively receiving a video stream (Figure 4). We compared the results of mCast and IP unicast for both STAR and MESH ISP topologies.

As expected, in case of IP unicast the link utilization increased linearly with the number of clients. Duplicate packets passed through same link for each client, increasing the link utilization linearly, until congestion occurs and the links are saturated. As in MESH topology, a stream has to traverse more links to reach the client, link utilization in MESH was higher than that of STAR topology. For mCast, the amount of traffic generated in ISP over a certain link remains constant avoiding any bottlenecks in the network. For a large number of clients, IP unicast overloaded network nodes and content servers, resulting in clients failing to connect to the servers. This adds unreliability to both the network and the streaming service. In mCast, such situation did not occur as the content server was not overloaded even when the number of clients was very large i.e. 1000.

In addition to the decrease in intra-domain bandwidth consumption and link utilization, it is also important to notice that in mCast, the stream enters the ISP network only once. Inter-domain traffic is usually more expensive and valuable for both ISPs and CDNs. Using mCast, for a 1000 client, this traffic got reduced to just one stream in mCast from a 1000 streams in IP unicast.

3) Start-up Delays: Start-up delay is the time a client has to wait from sending a content request until it starts receiving the stream. We measured start-up delays for 600 clients in STAR topology for unicast and mCast. For IP unicast, the client request goes to the streaming server which establishes a connection and starts streaming. For mCast, this request is intercepted by mCast CDN Agent which requests ISP to deliver stream to the client and only if ISP fails to do so, does the request goes to the streaming server.

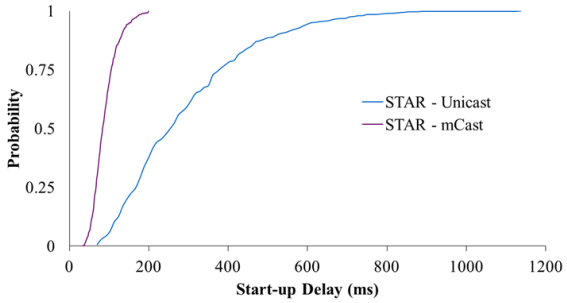

Fig. 5. Cumulative Distribution Function of Start-up Delays

We plot the results (Figure 5) as Cumulative distribution function (cdf) with delays in ms. As results show, the delay for mCast stayed below 200ms while in unicast, it went up to 1140 ms. This is due to the load on the server and the network, in case of unicast. As mCast eliminates these loads, the requests get responded very quickly which is shown in the results. The decreased start-up delays along with no dropped packets improves the video quality significantly and enhances the overall user experience.

4) OpenFlow rules and messages: Results showed that mCast can drastically decrease the network resource consumption for both ISPs and CDNs. However to setup mCast, extra OpenFlow rules and messages are needed in the network. The goal of mCast is to minimize all types of resource consumption, therefore we designed mCast in a way as to minimize the number of OpenFlow rules and messages needed. The number of OpenFlow rules (Figure 6) depends on the number of switches involved in mCast. For every switch,

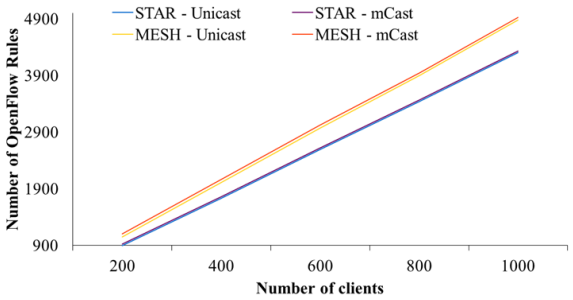

Fig. 6. Number of OpenFlow rules vs Number of clients

mCast needs only one extra rule per stream to match the incoming packet's destination IP and port with $V_{(IP, Port)}$ and forward it on the relevant physical ports. The mCast rule at egress switch has additional actions to modify the destination IP and port to the client's address before forwarding. In OpenFlow 1.3 these actions can be defined in one OpenFlow rule.

To add every client, the SDN controller calculates the path for that client based on the mCast tree and then sends an OpenFlow message to all the switches that need to add or modify their rules. While these control messages cause an overhead for the mCast service, these packets are usually very small and the overhead caused is negligible in comparison to the amount of bandwidth saved due to the data packets.

V. Conclusion

In this paper, we proposed mCast as a novel scalable architecture for live streaming. mCast provides a framework for communication between ISPs and CDNs. The CDN sends only one copy for a video channel towards the ISP leading to a significant reduction in CDN egress link. Additionally, ISP reduces the bandwidth utilization by significantly reducing redundant transmission in its network. mCast is transparent to the client and maintains the CDN control on the content distribution. We presented a decision model that can be used by CDNs to determine when switching to mCast can be profitable for them. We performed a large scale evaluation with up to 1000 clients, emulated in Mininet. Our results show that mCast decreases link utilization by more than 50% in comparison to IP unicast. Additionally, it reduces start-up delays to less than 200 ms and eliminates video frame loss which is up to 39% in case of IP unicast.

Acknowledgment

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number: 13/IA/1892.

References

[1] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the ip multicast service and architecture," *IEEE Network: The Magazine of Global Inter-networking*, vol. 14, no. 1, pp. 78–88, January 2000.

[2] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas, "A survey of application-layer multicast protocols," *IEEE Communications Surveys & Tutorials*, vol. 9, no. 3, pp. 58–74, September 2007.

[3] E. Haleplidis, K. Pentikousis, S. Denazis, J. Salim, D. Meyer, and O. Koufopavlou, *Software-Defined Networking (SDN): Layers and Architecture Terminology*, January 2015, https://www.rfc-editor.org/rfc/rfc7426.txt.

[4] P. Gill, M. Schapira, and S. Goldberg, "A survey of interdomain routing policies," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 1, pp. 28–34, January 2014.

[5] D. Kotani, K. Suzuki, and H. Shimonishi, "A design and implementation of openflow controller handling ip multicast with fast tree switching," *IEEE/IPSJ 12th International Symposium on Applications and the Internet (SAINT)*, July 2012.

[6] C. Marcondes, T. Santos, A. Godoy, C. Viel, and C. Teixeira, "Castflow: Clean-slate multicast approach using in-advance path processing in programmable networks," *Proc. of the 2012 IEEE Symposium on Computers and Communications (ISCC)*, pp. 94–104, July 2012.

[7] F. Corasa, J. Domingo-Pascuala, F. Mainob, D. Farinaccib, and A. Cabellos-Aparicioa, "Castflow: Clean-slate multicast approach using in-advance path processing in programmable networks," *Elsevier Journal Computer Networks*, vol. 59, pp. 153–170, Feburary 2014.

[8] J. Ruckert, J. Blendin, and D. Hausheer, "Software-defined multicast for over-the-top and overlay-based live streaming in isp networks," *Journal of Network and Systems Management*, vol. 23, no. 2, pp. 280–308, April 2015.

[9] E. Yang, Y. Ran, S. Chen, and J. Yang, "A multicast architecture of svc streaming over openflow networks," *IEEE Global Communications Conference (GLOBECOM)*, December 2014.

[10] K. Noghani and M. Sunay, "Streaming multicast video over software-defined networks," *Proc. of the 2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 551–556, October 2014.

[11] J. Jiang, H. Huang, J. Liao, and S. Chen, "Extending dijkstra's shortest path algorithm for software defined networking," *16th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, September 2014.

[12] S. Zhang, Q. Zhang, H. Bannazadeh, and A. Leon-Garcia, "Routing algorithms for network function virtualization enabled multicast topology on sdn," *IEEE Transactions on Network and Service Management*, vol. 12, no. 4, pp. 580–594, August 2015.

[13] "Cisco global cloud index: Forecast and methodology, 20152020 (white paper)," Cisco Public, Tech. Rep., 2016.

[14] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, *An Architecture for Differentiated Services*, December 1998, https://www.rfc-editor.org/rfc/pdfrfc/rfc2475.txt.pdf.

[15] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, November 2010.

[16] W. B. Norton, *The Internet Peering Playbook: Connecting to the Core of the Internet*. DrPeering Press, 2012.

[17] J. C. Chuang and M. A. Sirbu, "Pricing multicast communication: A cost-based approach," *Journal of Telecommunication Systems*, vol. 17, no. 3, pp. 281–297, July 2001.