CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	An SDN-based device-aware live video service for inter-domain adaptive bitrate streaming
Authors	Khalid, Ahmed;Zahran, Ahmed H.;Sreenan, Cormac J.
Publication date	2019-06
Original Citation	Khalid, A., Zahran, A. H. and Sreenan, C. J. (2019) 'An SDN-based device-aware live video service for inter-domain adaptive bitrate streaming', Proceedings of the 10th ACM Multimedia Systems Conference, Amherst, Massachusetts, 18-21 June, pp. 121-132. doi: 10.1145/3304109.3306229
Type of publication	Conference item
Link to publisher's version	https://dl.acm.org/citation.cfm?id=3306229 - 10.1145/3304109.3306229http://www.mmsys2019.org/
Rights	© 2019, Association for Computing Machinery. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in Proceedings of the 10th ACM Multimedia Systems Conference: https://doi.org/10.1145/3304109.3306229
Download date	2026-11-17 09:04:03
Item downloaded from	https://hdl.handle.net/10468/8171



UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

An SDN-Based Device-Aware Live Video Service For Inter-Domain Adaptive Bitrate Streaming

Ahmed Khalid
Department of Computer Science
University College Cork, Ireland
a.khalid@cs.ucc.ie

Ahmed H. Zahran
Department of Computer Science
University College Cork, Ireland
a.zahran@cs.ucc.ie

Cormac J. Sreenan
Department of Computer Science
University College Cork, Ireland
cjs@cs.ucc.ie

ABSTRACT

The emerging popularity of live streaming services poses a great challenge for the rigid and static traditional Internet architecture. The rise in adaptation of Software Defined Networking (SDN) by Internet Service Providers (ISP) and Content Delivery Networks (CDN) presents an opportunity to dynamically adapt and respond in real-time to high definition (HD) mega events or dynamic short-lived broadcast events. In this paper, we present an SDN-based system design that utilizes a communication framework between ISPs and CDNs to interact and thus enable a reliable and resource efficient live streaming service. We build and deploy an optimization model that can maximize the video quality for users while minimizing the resource utilization for both ISPs and CDNs. The model considers device capabilities, network constraints and the subscription level of users with the ISP/CDN. Our system is a network-assisted, cross-layer, approach that implements multicast at the network layer and can dynamically adapt the video bitrates that are served to each client at the application layer. We build a prototype of our proposed design and evaluate real-world scenarios with up to 500 users streaming multiple videos at different bitrates. Results show that our approach can increase average user goodput by up to 70% while almost eliminating frame drops by handling network congestion.

CCS CONCEPTS

• **Networks** → **Network design principles**; *Cross-layer protocols*;
• **Information systems** → **Multimedia streaming**;

KEYWORDS

Network-assisted solutions, ISP and CDN collaboration, software defined networking, live video, congestion handling, network layer multicast, optimization models, video streaming

ACM Reference Format:
Ahmed Khalid, Ahmed H. Zahran, and Cormac J. Sreenan. 2019. An SDN-Based Device-Aware Live Video Service For Inter-Domain Adaptive Bitrate Streaming. In *10th ACM Multimedia Systems Conference (MMSys '19), June 18–21, 2019, Amherst, MA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3304109.3306229

1 INTRODUCTION

The latest social media trends have increased the amount of user-generated live video content over the Internet through various streaming platforms such as Periscope , YouTube Live and Twitch, giving rise to frequent flash crowd events [18]. Additionally, more traditional TV programs such as news, sports events and political events are now streamed online at high definition (HD) qualities to large and ephemeral audiences, making it challenging to provision streaming resources in an efficient and flexible manner. The heterogeneous device capabilities of users, ranging from smart-phones and tablets to ultra-HD 4K TVs, further elevate the challenge to dynamically deliver the video streams at multiple bitrates that match the specific needs and requirements of each user [24].

When delivering live video streams inside an Internet Service Providers' (ISP) network, the Content Delivery Networks (CDN) rely on either IP unicast e.g. in DASH-based systems or overlay multicast e.g. P2P-based systems [22]. These approaches enable good control and management of end-devices by establishing end-to-end connections, but result in redundant transmissions in the network layer and waste system resources for both ISPs and CDNs. On the other hand, native IP multicast can help reduce resource consumption by eliminating packet duplication over network links and at content servers, but its adoption is stymied by lack of desirable features such as management, authorization and accounting [7]. Today, IP multicast is limited to intra-domain pre-provisioned services such as ISP-oriented IPTV [5].

Software-defined networks (SDN) significantly enhance the degree of control in IP networks [11] and can enable dynamically manageable network-layer multicast over the Internet. Increasing number of ISPs and CDNs are incorporating SDN in their domains [19], which serves as a motivation to rethink the design of live video streaming services. Some recent research such as SDM [20], LiveJack [25] and mCast [13] have proposed services and architectures that enable efficient inter-domain content delivery using network-layer multicast. However, these solutions have salient limitations that would restrict their practical deployment. Specifically, we believe that a practical solution must account for the heterogeneity and capability of end-user devices, support multiple video qualities and not ignore the adaptive bitrate aspect of live streaming, handle network congestion and offer CDNs the control and privacy essential for their business.

In this paper, we propose a Device-Aware Network-assisted Optimal Streaming service, Danos, that merges the flexibility and control of SDN with resource efficiency of network-layer multicast to enable inter-domain adaptive bitrate streaming. Danos involves minimal modification at the server and client side, to support transparent multicast for end-nodes. Danos is designed to maintain user and CDN data privacy by avoiding any deep packet inspection and provides CDNs with full control over their clients so they can perform authentication, authorization and accounting (AAA) functions. Through Danos, we have made four contributions:

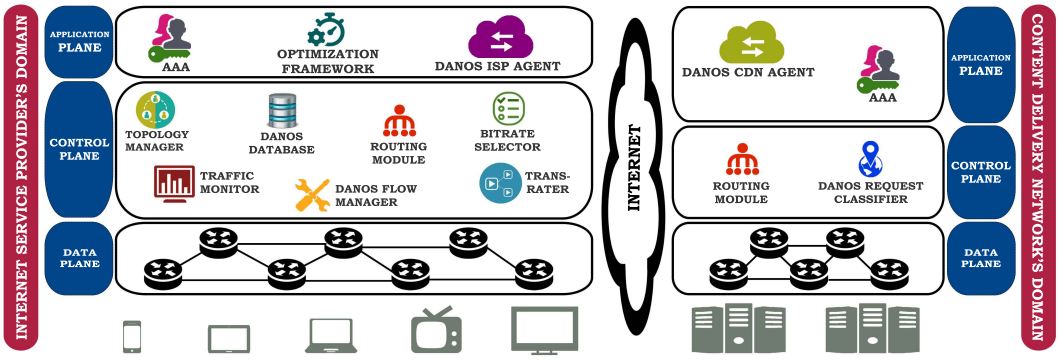


Figure 1: Architecture of Danos and all the essential components.

- A novel SDN-based architecture that enables an inter-domain adaptive bitrate live streaming service. We present all the essential components and various architectural design choices that can be made by CDNs or ISPs.
- The formulation of a novel multi-objective optimization problem to maximize the perceived video quality of all the users and minimize the utilization of the ISP network. Our formulation accommodates ISP or CDN operation constraints and scales with the number of users. The performance analysis of the model shows that it can be solved in the order of milliseconds for millions of users in the network, and can improve video quality for users in comparison to a state-of-the-art approach.
- Design features to address key challenges that arise by using optimized multicast and considering device-specific requirements when serving bitrate adaptive live video streams. As multicast sessions are UDP-based and, unlike TCP, there is no feedback loop or constant interaction between steaming servers and clients, we discuss how the bitrate of users can still be switched smoothly without causing frame jitter. We also look at how to synchronize various components in the parallelized architecture when optimizing the network and minimizing the start-up delays for clients.
- A prototype implementation of Danos for demonstration and evaluation, used for large-scale experiments with up to 500 clients streaming multiple videos at multiple bitrates. To show the feasibility, scalability, robustness and gains of Danos, we test for real-world scenarios including flash crowd events and cross-traffic and evaluate the performance of Danos against a state-of-the-art approach.

The rest of the paper is organized as follows. In Section 2, we present our proposed architecture and describe various system components. In Section 3, we address the key design challenges. In Section 4, we formulate our optimization problem and the real-time guided optimization technique. In Section 5, we present our prototype implementation and evaluate the performance of our solution. In Section 6, we share a review of the literature and related work and finally, in Section 7, we conclude our work.

2 DANOS ARCHITECTURE

Danos enables efficient delivery of bitrate-adaptive video streams in both CDN and ISP networks by leveraging SDN capabilities. Specifically SDN facilitates global network state, flexible data forwarding and standardized real-time route modification and packet editing. Danos deploys agents in CDN and ISP domains to allow unicast delivery of different streams from a CDN to an ISP network and network-layer multicast inside the ISP network. Such sessions would be denoted below as Danos sessions. We assume that the ISP offers such sessions as a service for interested CDNs. Based on their business requirements and decision models [13], CDNs may request such service, and ISPs would accordingly set up and manage multicast trees for Danos video streams.

Figure 1 illustrates the architecture of the Danos system and its various essential components. In the remainder of this section, we first describe the functionality of each individual component. Then, we present the overall workflow of Danos (Section 2.9) using examples of important messages exchanged between the different components to initiate and maintain a Danos session.

2.1 Request Classifier

When a client wishes to watch a video stream served by a CDN, it sends a session join request which is typically delivered to AAA modules that authenticate requests and handle legitimate requests. In Danos, after handling the request, AAA forwards it to a Request classifier module, which is located in the control plane of the CDN. The classifier identifies the client's ISP network and forwards the request to the Danos CDN Agent. Such classification can be performed using geo-location databases.

2.2 Danos CDN Agent

Danos CDN Agent is implemented at the application plane of the CDN's SDN-based architecture and decides how content is served. On reception of the user request, Danos CDN Agent determines whether it would be served as a unicast stream or would be part of a Danos stream. Generally, the request would be served as unicast if it is not from a Danos-based ISP or the CDN policies to initiate a Danos session have not yet been met.

Unicast-case: Danos CDN agent prepares and sends a typical content request message to the selected server and configures the CDN forwarding nodes for the stream delivery. The configuration is done by sending messages from the SDN controller to the forwarding nodes, e.g., *flow_mod* messages in OpenFlow. The server starts streaming the content to the user. The routing of this flow is

handled normally by the CDN routing module. We assume that a CDN performs destination NAT at the edge router.

Multicast-case: If the requested stream is already served by a Danos session, then the CDN Agent will ask the ISP Agent to add the new client to the multicast group of the stream. This request includes client's IP address, port number, the content server's IP address and a randomly generated source port number which will be used for smooth switch-over if reverting back to a unicast stream for that client. This *add-client* request also includes device specification and CDN subscription level for the user.

Danos CDN Agent implements multicast policies that determine when a stream should be switched to multicast in an ISP. In this case, Danos CDN and ISP Agents interact to initiate the transition process as presented below. Typically, a multicast policy depends on the economics of bandwidth utilization and the ISP charges for multicast delivery in the latter network, e.g. [13].

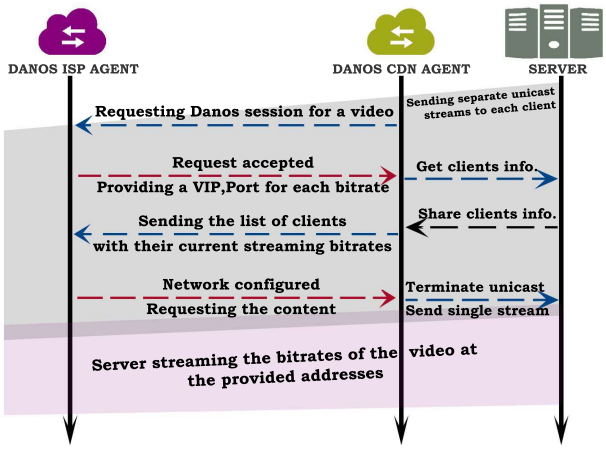


Figure 2: Message exchanges between a CDN and an ISP to initiate a Danos session for a live video content.

2.3 Danos ISP Agent

The Danos ISP Agent is implemented at the application plane of the ISP network and handles the requests coming from CDNs. It performs two main functions: interfacing with Danos CDN Agent using SDN east-westbound interface and orchestrating multicast operations in the ISP network using northbound interface.

Figure 2 illustrates the message exchanges that take place between Danos ISP and CDN agents. When a CDN wants to trigger multicast for a video stream, it sends a session aggregation request to the ISP Agent. After analyzing the request and performing initial assessment, the ISP Agent responds by creating an identifier for each bitrate of the stream. The identifier is composed of an IP address and a port number, denoted as $V_{(IP,\ Port)}$. An ISP can choose an address from a pool of IPv4 addresses that it reserves for Danos service. It can then use port numbers to distinguish video streams or bitrates.

Upon reception of $V_{(IP,\ Port)}$, a CDN sends *add-client* requests with details of the clients located in the ISP network and watching that video. These details include the header fields, their current streaming bitrates, and maximum and minimum allowed bitrates for each user. Such information is used to identify the target users in the ISP network and ensure a good video quality for users.

The Danos ISP Agent then instructs the modules, discussed in following sections, to create multicast trees, configure the network and ensure transparent delivery to all the clients. To provide complete control of its clients to a CDN, the ISP does not intercept any *session-join* or *session-leave* requests from clients even after a multicast stream has been initiated using Danos. These requests are instead delivered to the CDN to be handled as described above. The Danos CDN Agent may ask the Danos ISP Agent to remove a user from a Danos session when needed (e.g., channel switching or application termination).

Note that the propagation delay for the control is comparable for unicast clients and clients added to Danos sessions. Moreover for data delivery, Danos offers lower start-up delays than a conventional streaming service, as the content is already available near the client through Danos multicast streams. Such interaction and dynamic updating of the multicast trees is facilitated by the centralized control and global view of the SDN and is not achievable in a traditional Internet architecture.

2.4 Danos Database and Bitrate Selector

Once a Danos session has been initiated for a video stream, the CDN Agent sends details of clients to the ISP Agent to add to the stream. In a Danos session, the best streaming bitrate for each client should be served. This rate depends on different factors:

- The receiving devices can have different specifications and capabilities resulting in different bitrate limitations.
- CDNs offer different subscription levels to users which may limit the maximum bitrate that a user can be served with.
- ISPs offer different packages to users with different bandwidth limitations.
- Based on the current cross-traffic being generated or received by a user, the bitrate that a user can be served may vary.

Our architecture gathers the information provided by the CDN and collected by different SDN modules and stores it in a database. When an *add-client* request is received by the Danos ISP Agent, it updates the database with any new received information and instructs the bitrate selector to choose a bitrate for the user. For clients switching from unicast to multicast, the Danos CDN Agent provides their current streaming bitrates to the ISP Agent and to enable smooth switch-over these clients are initially served with their current bitrates and are optimized later as explained in Section 4.

For any new clients, the bitrate selector accesses the Danos database and uses all the available information to find the highest bitrate that the user can support and passes on that information to the Danos ISP routing module for further processing.

2.5 Danos Routing Modules

In the CDN network, the routing module implements the CDN routing policies and installs relevant flow rules for both unicast and Danos streams. In the ISP network, the routing module receives a message from Bitrate Selector that includes the highest bitrate for a client. The routing module accesses the information saved in the database by the Topology Manager module and finds a path

from the ISP entry point to the user's egress switch and assigns the highest bitrate that is available on the path and can be assigned to the user. It then adds the user to the multicast tree of that bitrate and passes on that information to the Danos Flow Manager to configure network nodes and deliver the stream to the client.

2.6 Danos Optimization Engine

To minimize the start-up delays for users, Danos ISP routing module finds paths and configure network for clients as soon as an *add-client* request is received from the CDN Agent. However, the bitrate chosen for a client might not be feasible or optimal depending on the current network state. The Danos Optimization framework runs as a network application on the application plane of the ISP network and uses the global knowledge acquired by Danos database to find the optimal solution for all the videos and clients that are currently being served in the network using Danos. This framework also considers any cross-traffic on the links on paths used for multicast. Such information is stored in the database by the Traffic Monitor module of ISP. Danos optimization engine can function in two different modes:

Event-based: The network and client states can be monitored and the optimization module initiated when certain events occur; e.g., a new Danos session being added to the network, the amount of cross-traffic on the multicast paths exceeds certain threshold, and/or a certain number of clients have been added to or removed from a video stream.

Periodic: The optimization framework can run at regular intervals, make the optimal decisions and update the network paths and user bitrates. Re-configuring the network too often can result in excessive signaling overheads, therefore when running the framework periodically, the network and users may only be re-configured if the gains of doing so increase beyond a certain percentage.

2.7 Danos Flow Manager

The ISP Routing Module and Optimization Engine choose bitrates for clients, decide the paths to be used and instruct the Danos Flow Manager to configure the forwarding nodes. The Flow Manager uses the south-bound interface to install rules on SDN-enabled switches. The multicast entries in network nodes are installed with higher priority than IP unicast, ensuring that clients are served with Danos streams whenever possible. In addition, Danos Flow Manager installs transparency rules on the egress switch.

Before forwarding a packet to the client, the transparency rule modifies the source and destination IP address and port number to match the client-specific details, so the client receives the packet just as it would in IP unicast, hence the delivery is transparent. These rules ensure that the clients do not need to be modified to support Danos streams. 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.

2.8 Danos Streaming Server and Client

A typical live streaming server streams the requested content via unicast-based RTP/UDP sessions. To support Danos, we implement an API in the server to communicate with the Danos CDN Agent.

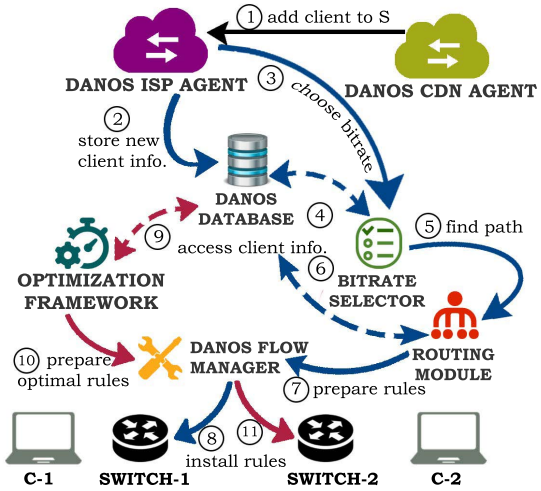


Figure 3: An example workflow of adding clients to an established Danos session.

When the CDN agent wants to switch between unicast and multicast, it uses this interface to perform the following tasks:

Acquire the current state of the clients: This information is helpful for a smooth transition from unicast to multicast and providing a seamless video experience to active clients.

Terminate individual client transmissions and initiate a single transmission instead: Along with this instruction, the CDN Agent provides a destination IP and port, $V_{(IP, Port)}$ to the server for each bitrate and the server establishes connections. To avoid any lost or skipped frames at the client device, the new connection is activated before terminating the old unicast sessions. This sequence avoids any skipped frames; however, it may lead to duplicate packets at the client, which will be ignored by the client after inspecting the RTP headers.

Terminate single transmission per bitrate and initiate individual client transmissions: When switching back from multicast to unicast, the CDN Agent will send a list of clients (IP,port) to the server with an instruction to terminate the connection with $V_{(IP, Port)}$ and the server will act accordingly.

Due to the transparent delivery mechanism of Danos, any standard UDP-based live video client can be used with a minor consideration at the application layer. To enable smooth switch-over between different video bitrates, we propose a design in Section 3.1 and a Danos client should incorporate this design to correctly handle and decode video frames.

2.9 Danos Work-flow

We now present an example (Figure 3) to explain how different components in the ISP domain interact with each other.

The Danos CDN Agent decides to deliver a video stream S, available at bitrates 1-Mbps and 2-Mbps, through a Danos session. It initiates this session by following the steps shown in Figure 2 and starts delivering both bitrates to the ISP using single streams. Client C-1 sends a *session-join* request to the CDN and receives an IP address of a content server after AAA functions have been performed. When C-1 sends a *content-request* at the provided address, the

Danos CDN Agent receives it and sends an *add-client* request to the Danos ISP Agent (1). Along with C-1's IP address and port number, this request includes information such as the maximum bitrate that the client can support and any CDN user subscription details. The ISP Agent stores the newly learned information in the Danos database (2) to be accessed by other modules and sends an instruction (3) to the Bitrate Selector to find an appropriate bitrate for C-1.

The bitrate selector accesses the information stored in the database (4) by the Traffic Monitor, Danos ISP Agent and AAA modules of ISP and chooses 2-Mbps as the best bitrate for C-1. It instructs the routing module (5) to find a suitable path for C-1. To locate the client in the ISP network, the routing module retrieves the information stored in the Danos database (6) by Topology Manager module. It then uses its bitrate-specific graphs to find the least expensive path to add C-1 to the multicast tree and decides to deliver the stream to C-1 through Switch-1. The routing module saves the newly updated graphs back to the database, to be used later by the optimization framework and sends a request to the Flow Manager (7) to configure the forwarding nodes and enable stream-delivery for the client.

The Danos Flow Manager installs or updates forwarding entries on the nodes in the path to deliver the bitrate to Switch-1 and installs or updates transparency rules on Switch-1 (8) to modify the packet headers before forwarding them to C-1. At this point, C-1 will start receiving the requested video quality at 2-Mbps. The process is repeated when the Danos ISP Agent receives a new *add-client* request from the CDN Agent for C-2 and C-2 starts receiving 2-Mbps bitrate through Switch-2.

The optimization framework runs as a separate application than the bitrate selector. It gains the global information of clients and network state from the Danos database (9) and solves the optimization problem to maximize a system utility while respecting network, user and device constraints. In this example, the optimization framework detects congestion on the path to C-2 and realizes that C-2 will experience frame losses if it kept on receiving 2-Mbps and decides to stream 1-Mbps to C-2 instead. It instructs the Flow manager (10) to reconfigure the forwarding nodes and Switch-2 accordingly. The Flow Manager updates the forwarding entries (11), using smooth switching technique discussed in the next section, and switches the bitrate of C-2 to 1-Mbps which now receives a better quality of video by eliminating frame drops.

3 SYSTEM DESIGN

We address and resolve three key design challenges that arise by using optimized multicast and considering device-specific requirements when serving bitrate adaptive live video streams.

3.1 Smooth switching of client bitrates

To avoid video glitches and image jitters, the bitrate of a client should only be changed after it has received a complete group of pictures (GOP) i.e. at the next I-frame. In unicast transmissions this is not a problem because each client has its own separate connection with the streaming server and the server would know when a GOP has ended. As existing multicast-based solutions only consider single bitrate schemes, they do not address this issue.

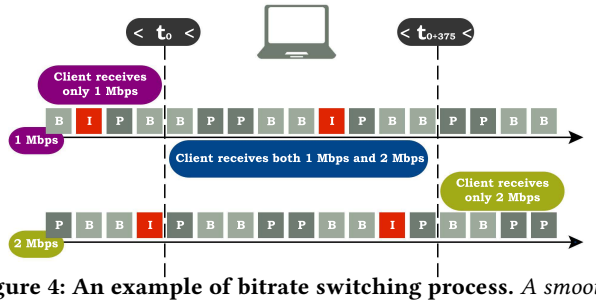


Figure 4: An example of bitrate switching process. *A smooth switch is ensured by delivering both rates for a duration of a GOP.*

However, in our adaptive bitrate multicast-based solution, when the ISP optimizer module decides to switch the bitrate of a client based on the current network state, it must prevent the client from losing a GOP and experiencing jitter.

The decisions made by the optimization framework translate into three types of actions:

- change the network path for one or more bitrates of a stream
- serve one or more clients with a new bitrate and
- stop serving those clients their current bitrate.

These actions are then translated into flow-table entries and are installed on the forwarding nodes by Danos Flow Manager. To ensure a smooth bitrate switch for clients, we handle these actions in different ways. If a decision only involves changing the path and re-routing the traffic, it is installed on the forwarding node immediately. However, if a decision involves switching the bitrate for a client, the action of delivering the new bitrate to the client is installed immediately but the action to stop serving the current bitrate is taken with a delay of a duration of one GOP.

Figure 4 shows an example to demonstrate a switching action. A GOP of 8 frames is considered with the video encoded at 24 frames per second. This implies a GOP duration of $333ms$. For a client, the video is delivered at 1-Mbps until time t_0. At time t_0, the optimizer decides to switch this rate to 2-Mbps and initiates the switching process. The action to deliver 2-Mbps is taken immediately and the client starts receiving frames at both bitrates.

We wait for a duration of $375ms$ during which time the client receives 9 frames at both bitrates. This will ensure that the previous GOP has been completely received by the client and the user does not experience any jitter in the video. The event where a client receives frames from two distinct bitrates can serve as an indication that a bitrate switch is taking place and the client can decode the correct frames by inspecting the RTP headers while discarding the additional unnecessary frames from both bitrates.

This design choice avoids the need for deep-packet inspection to detect the start and end of a GOP in the ISP domain and maintains content provider's and user's data privacy. It also avoids the need of constant signaling between ISP and CDN Agents to share the start and end of each GOP. The CDN Agent will instead share the duration of a GOP when initiating a Danos session for a stream and the Danos Flow Manager will use that information for a smooth transition of bitrates for clients.

Multiple rates of an adaptive bitrate live video are always transmitted synchronously. This prevents users from experiencing frame skipping or playout delays when switching between bitrates. A

minor time difference, due to the dynamic nature of network or different amount of data for different bitrates, can be handled by the client's playout buffer.

During the smooth switching process, the forwarding nodes will be transmitting two distinct bitrates (R_1 and R_2) of the same video stream. If any of the links on the path is congested, it might result an input rate (R_{in}) higher than the out rate (R_{out}) of the network buffer in the respective forwarding node. An ISP must plan ahead for such events to avoid buffer overflows and ensure reliable packet delivery. We provide a simplified example on how to calculate the extra amount of buffer length needed for a stream. Assume that a video is being streamed at full-HD at a bitrate of $R_1 = 8Mbps$ and 4K at $R_2 = 20Mbps$, and on the path to a client, one of the network buffer can support R_{out} up to $20Mbps$.

The optimization framework chooses to switch the client to 4K and instructs the Danos Flow Manager to initiate smooth switching. The total R_{in} during the switching period will be $R_1 + R_2 = 28Mbps$ which is $8Mbps$ more than R_{out} of the bottleneck buffer and will result in packet drops. Assuming that the video is encoded at 24fps and 8 frames per GOP, the additional bytes that will arrive in one GOP duration ($8/24 * 1000 = 333ms$) will be $8Mbps/8/334ms = 333KBytes$. Considering a maximum transmission unit (MTU) of 1500 bytes, $333KBytes/1500 = 222$ additional packets will arrive at the network buffer during the switching period. Similar exercise can be conducted for each video with its respective highest bitrates to get the total size of extra queue length needed.

An additional advantage of an SDN-based architecture is the centralized control provided by the SDN control plane that can be used to perform these calculations dynamically and reconfigure buffer lengths based on the policies and priorities of the ISP network.

3.2 Synchronizing Optimization Framework and Routing Module

The optimization framework is activated by the Danos ISP Agent to run periodically or based on some events. It runs as a separate application and optimizes a system utility and the network link utilization by solving an optimization problem. One key challenge for running the framework as a separate application is handling the clients that arrive during the computation span i.e. after the problem starts computing until the optimal solution is found and the framework is ready to re-configure the network.

If any new clients arrive during the computation span, the solution found by the optimization framework may not be optimal anymore, depending on where the new clients are located in the network and their specifications. Although this problem is natural for real-time systems and cannot be avoided, the decision on what paths or what bitrates to choose for these clients impacts the system performance. There are two ways to handle these clients:

- Add the new clients to the multicast paths as soon as the request arrives
- Delay serving new clients until the end of computation span of the optimization framework

As mentioned in Section 2, to minimize start-up delays, the Danos ISP Routing Module finds paths and requests network configuration as soon as an *add-client* request is received. Delaying clients until the end of computation span would imply pausing the routing

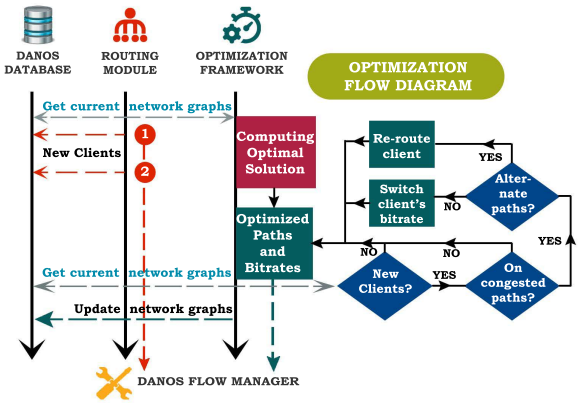


Figure 5: Flow diagram of Optimization Framework *Clients that arrive during computation will be switched to lower bitrates only if their current rates cannot be served.*

module and increasing start-up delays which has a negative impact on user experience. On the other hand, adding the clients as soon as they arrive might lead to the routing module choosing paths with congested links, which the optimization framework would have resolved by solving the constrained-problem globally.

We resolve the issue by proposing a hybrid approach as illustrated in Figure 5. At the start of the computation span, the optimization framework accesses network paths and graphs from Danos database and saves a copy of it. It then proceeds with solving the problem and making decisions which translate to either re-routing or switching bitrates for some clients. During the computation span, the routing module keeps adding new clients to the session.

After solving the optimization problem, the framework accesses the database again and gets the current network paths and graphs to determine whether any new clients have been added during the computation span and what paths and bitrates have been assigned to them. If the new clients happen to be on the optimal paths and receiving the optimal bitrate already, then no further action is needed and the optimization framework can configure the network based on its solution. Similarly, the optimization framework leaves the clients on un-congested paths untouched and does not update its solution to alter their paths.

If a client is assigned a certain bitrate on a congested path and the paths decided by optimization framework are able to deliver the same bitrate to the client through another path, then the action to reroute that client is added to the optimal solution. However, if no alternate paths are available then the optimal solution is updated with actions to initiate smooth switching for the client.

This approach avoids additional start-up delays for clients and handles any network congestion that might occur due to an un-optimized path. The final solution is then sent to the Danos Flow Manager and the network is re-configured accordingly.

3.3 ISP Transrating Services

Danos provides a resource efficient delivery mechanism for CDNs using multicast and the CDNs can save their energy cost and Internet transit cost by serving all the clients in an ISP with one stream per bitrate of a video. To further maximize on the savings, we propose a transrater module in the ISP network that can be rented by

the CDNs. After establishing multicast for a live video session, if a CDN opts for transrating services then it will deliver one stream per video through the Internet Exchange Point (IXP) into the ISP network where it will be received at the transrater module. This stream can then be requested by the forwarding modules of the ISP as needed.

This design is beneficial for both ISPs and CDNs. For a CDN the costs for serving a stream at multiple bitrates will be reduced. For an ISP, in addition to the monetary gains, the ISP network will be capable of accessing the streams in a more dynamic fashion by requesting the transrater to generate only the bitrates that are currently being served to the active clients in the network.

If an ISP provides Transrating services, the Danos Flow Manager will interface with the Transrater module to facilitate efficient resource utilization in the ISP network for the CDNs that are renting transrating services. When installing flow entries on the switches, the Flow Manager will check if a new bitrate is to be served to any clients in the network. For such cases, it will request the Transrater to make that bitrate available in the network. Similarly, when removing entries from switches, the Flow Manager will check if a certain bitrate is no longer served to any client in the network. In this case, the transrater will be informed to stop transrating the video on that bitrate.

4 GLOBAL OPTIMIZATION MODEL

For our Danos Optimization Framework, we formulate a novel multi-objective optimization problem with the primary goal to maximize the system utility of all the Danos clients in the network and a secondary goal of minimizing the network link utilization. Our model can solve the problem for any system utility that is a function of user-bitrates or bitrate-switches or both. This makes the model generic and independent of any particular Quality of Experience (QoE) or fairness function.

As discussed in Section 3.2, a longer computation time for the model can result in a sub-optimally configured network. In this section, we first present our model and then propose a scalable guided-optimization approach for the Danos Optimization Engine that can solve the problem in real-time regardless of the number of users in the network.

4.1 System Model

We consider an ISP network with a number of forwarding nodes used for Danos to serve a set of videos V to multicast users in set M. Each video v is encoded and streamed at a set of bitrates, denoted by R_v and a node i may receive one or more bitrates of v from any of its previous hop node $j \in L_i$. The capacity on the link from node i to j is denoted by c_{ij} and for a user m this capacity will be limited based on the user's ISP subscription level. Each link may have some cross-traffic and a constant α_{ij} represents the percentage of the capacity available or allowed for multicast sessions. Note that even for links with no cross-traffic a small percentage of total link capacity can be made inaccessible to multicast sessions, to avoid large network queuing delays.

A video v enters the ISP domain at a forwarding node e_v. A user m may request one or more videos at the same time. For m watching a video v, a maximum bitrate $m_{v_{max}}$ that can be assigned to the user is defined based on its device specification and CDN user subscription level. The model also accepts a minimum bitrate guarantee $m_{v_{min}}$ that might be offered to m for video v. Table 1 summarizes all the notations used for the model.

Table 1: Notations For Optimization Model

Symbol	Description
INPUTS	
V	Set of active video sessions served by Danos
M, M_v	Set of multicast users (M) subscribed to video v (M_v)
R_v	Set of bitrates available for video v (arranged in ascending order)
$m_{v_{max}}$	Index of maximum bitrate allowed or possible for user m watching video v
$m_{v_{min}}$	Index of minimum bitrate guarantee for user m watching video v (Set to lowest if not specified)
N	Set of all the network and user nodes
L_i	Set of previous hop nodes to reach node i
c_{ij}	Capacity on the link between node i and j
α_{ij}	Maximum fraction of link capacity allowed for Danos between node i and j
β_{ij}	Priority weight for the link between node i and j
$f(m,r)$	A system utility function that takes user m and bitrate r as an input
VARIABLES	
B_{ijvr}	Binary variable to determine if bitrate r of v has been chosen to be transmitted from node j to node i

4.2 Problem Formulation

Our problem is an Integer Linear program (ILP) with the primary objective of maximizing the cumulative system utility for all users. We design our problem in a way that is agnostic to the utility function being used. The model accepts any metric as a utility function that takes a bitrate as an input and returns the achievable utility value for a user, and the goal is to maximize the sum of the utility function for all users of all videos.

Problem 1: Optimal user bitrate assignment.

$$max \sum_{v \in V} \sum_{m \in M_v} \sum_{r \in R_v} \sum_{j \in L_m} f(m,r) \cdot B_{mjvr} \tag{1a}$$

subject to

$$\sum_{r \in R_v} \sum_{j \in L_m} B_{mjvr} = 1, \ \forall v \in V, m \in M_v \tag{1b}$$

$$\sum_{r=R_v[m_{v_{min}}]}^{R_v[m_{v_{max}}]} \sum_{j \in L_m} B_{mjvr} = 1, \ \forall v \in V, m \in M_v \tag{1c}$$

$$\sum_{v \in V} \sum_{r \in R_v} r \cdot B_{ijvr} + \sum_{v \in V} \sum_{r \in R_v} r \cdot B_{jivr} \leq \alpha_{ij} \cdot c_{ij} \ \forall i \in N, j \in L_i \tag{1d}$$

$$B_{ijvr} - \sum_{k \in L_j} B_{jkvr} \leq 0 \ \forall v \in V, i \in N - e_v, j \in L_i, \tag{1e}$$

where B_{ijvr} is a binary variable that determines if bitrate r of video v has been chosen to be transmitted from node j to node i. The objective function 1a only considers the links between users and their egress switches to check the bitrates that will be delivered to

the end-devices and ignores the links between forwarding nodes as these do not add any value to the user experience.

Constraint 1b ensures that a user only receives one bitrate of a video stream and Constraint 1c ensures that the assigned rate meets the minimum guarantee criteria and does not exceed maximum allowed bitrate by CDN or the maximum bitrate stream-able by the user device. Constraint 1d limits the total multicast traffic flowing through a link to the maximum capacity allowed for the Danos service. For bi-directional links, the traffic flowing in both directions i.e. i to j and j to i is considered and the sum should not exceed the allowed capacity.

Finally, Constraint 1e establishes paths in the network by guaranteeing that a node only transmits the traffic that it receives. This is achieved by ensuring that if a bitrate of a video is transmitted on a link i.e. $B_{ijvr} = 1$, then at least one of the previous hop link of node i should be 1. This is checked for all the forwarding nodes in the network except the entry point. The model assumes that the entry point always has all the bitrates and videos available. If a CDN is using the ISP transrating service then the entry point will be the node attached to the Transrater module or else it will be the ingress node between ISP and CDN.

A secondary objective of our model is to minimize the resource utilization in the network. If more than one solutions exist to **Problem 1**, where users would receive the same optimal bitrates, then this objective will find the solution with the least redundant transmissions on network links. Depending on the planned infrastructure of the network and how an ISP wants to shape the traffic, the ISP may have different priorities set for each network link. β_{ij} represents the priority weight for a link between node i and node j. Note that if no such priorities exist then this constant will be set to 1 for all links.

Problem 2: Minimizing the network load.

$$min \sum_{v \in V} \sum_{r \in R_v} \sum_{i \in N} \sum_{j \in L_i} r \cdot \beta_{ij} \cdot B_{ijvr} \tag{2a}$$

subject to

$$\sum_{r \in R_v} \sum_{j \in L_m} B_{mjvr} = 1, \ \forall v \in V, m \in M_v \tag{2b}$$

$$\sum_{r=R_v[m_{v_{min}}]}^{R_v[m_{v_{max}}]} \sum_{j \in L_m} B_{mjvr} = 1, \ \forall v \in V, m \in M_v \tag{2c}$$

$$B_{ijvr} - \sum_{k \in L_j} B_{jkvr} \leq 0 \ \forall v \in V, i \in N - e_v, j \in L_i \tag{2d}$$

If solved without **Problem 1**, the objective function 2a would choose the minimum bitrate for each user. However, with 1a as primary objective, the link utilization will only be minimized when it does not impact user video quality defined by the system utility function. Constraints 2b, 2c and 2d are the same as Constraints 1b, 1c and 1e.

4.3 Real-Time Guided Optimization

Although our problem is an ILP with binary variables and is NP-complete [9], depending on the number of video streams, the number of users and the size of the network, it might take too long to solve the problem in real-time and fast enough to be implemented in our dynamic optimization framework. The envisioned use cases of multicast streaming involve large number of users and the time span of computation is a crucial factor to the overall efficiency and performance of the system.

To make our model scalable, we eliminate the dependence on the number of users by relying on the fact that there are only a limited number of distinct user classifications. The users are classified based on their ISP or CDN subscription and device specification. Each of this classification results in a minimum and a maximum bitrate that the user can be delivered. We combine these classes together at each egress node, to form a group and identify users that belong to a certain group.

For example, users attached to an egress node will form a group, if: they are allowed full-HD quality by the CDN with a maximum bitrate of $8Mbps$; have an ISP bandwidth capacity between $8Mbps$ and the next higher bitrate served by the CDN (say $20Mbps$ for 4K), and are using a device that can seamlessly support $8Mbps$ streaming. Similarly all the users at each egress switch are grouped together based on the available bitrates for the video and their ISP, CDN or device capabilities.

We redefine the optimization model to take the groups as an input instead of individual users. An additional weight factor w_{gj} is introduced in the objective function of **Problem 1** which is equal to the number of users in a group g attached to an egress switch j. The remaining constraints and **Problem 2** are applied in the same way but by replacing the set M of all the multicast users by set G which represents all the groups. The objective function is as follows:

$$max \sum_{v \in V} \sum_{g \in G} \sum_{r \in R_v} \sum_{j \in L_g} w_{gj} \cdot f(g, r) \cdot B_{gjvr}, \tag{3}$$

where each $g \in G$ is defined by the connecting egress switch, ISP and CDN user subscription level, and device specification. As the number of elements in G are limited and will not increase with the number of users, this modelling trick makes the optimization model very scalable and helps solving the problem in real-time.

4.4 Performance Analysis

Scalability and Computation Time: There are three main factors that can impact the scalability of our model: number of users, number of video streams and the size of the network. We implement the model in Gurobi solver [2] and compute the mathematically optimal solution for two ISP topologies from Topology Zoo database [26] including AT&T network (a MESH topology) with 25 forwarding nodes and KREONET (approximately a STAR topology) with 13 nodes. We set the capacity of all the network links at $30Mbps$. To determine the computation time of the model, we vary the number of users, that have characteristics as described in Section 5, and change the number of video streams, each served at average bitrates of $400kbps$, $1.5Mbps$ and $4Mbps$. Figure 7 shows the results averaged over 20 runs.

For the smaller STAR topology, the solution was computed within 100ms for all cases. As our model implements a scalable pre-grouping approach, the number of users did not increase the computation time significantly. Even for 1 million users in the larger MESH topology, the optimal solution for a single video stream was

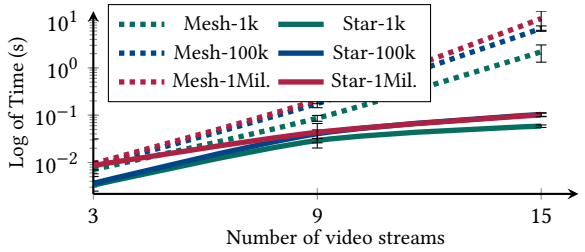


Figure 6: Time taken to compute optimal solution for Danos for different topologies, number of video streams and users.

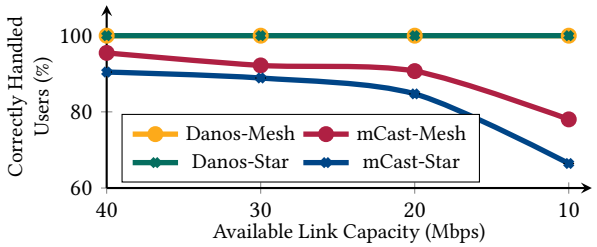


Figure 7: Percentage of correctly assigned users vs Available Link Capacity for 1 million users and 15 video streams

found within 2*ms*. The computation time increased with the number of video streams and for a fairly congested network the solution could be found in less than 1 second for up to 10 videos, which is a reasonable computation span for real-time systems.

Increasing the number of streams further, resulted in higher computation times, that might not be suitable to efficiently implement in a dynamic network. Based on the observed trends in the results, we propose two solutions for scenarios where an ISP intends to serve more than 10 video streams using multicast.

For very large networks, a **locally optimal solution** can be found by considering smaller slices of the network and solving the problem for each slice separately, which will result in lower computation times, as seen in Figure 7 for STAR topology. This is practical for large ISPs with national coverage where they manage local areas rather than finding a globally optimal solution for users located all over the country.

A second approach is to find a **near-optimal solution** by solving batches of videos in parallel, instead of all the videos simultaneously. The fraction of resources for each batch of videos will have to be determined prior to solving the parallelized problem and the nearness to the optimal solution will depend on the efficacy of the heuristics used for resource sharing. Proposing such a heuristic algorithm is out of the scope of this paper, however similar approaches exist for other network architectures, such as [6], where a multicast weighting function is proposed for multicast users in LTE networks.

Handling Network Congestion: We also look at the performance of our optimization model for Danos, by comparing it with mCast [13], a state-of-the-art approach that enables inter-domain network-layer multicast but does not consider network or user details. mCast has a broadly similar architectural view to Danos, thus easing the comparison. The metric for this analysis is the number of users that were assigned the best bitrate based on their specifications i.e. the highest rate which would result in no frame losses due to network congestion or device capabilities. Note that the metrics such as actual lost frames or average goodput, as presented in next Section, cannot be measured for this analysis as there is no aspect of time.

Even with no congestion in the network, users may experience frame losses if they request a bitrate that exceeds the bandwidth allowed by their ISPs or is higher than what their devices can support. While our optimization model and Danos is aware of such details and handles them to ensure no losses at user-end, other approaches do not. For a fair comparison of mCast with Danos, we consider that the users of the mCast service are smart and never request a bitrate higher than their device capabilities and also consider any cross-traffic they are receiving to ensure not exceeding their allowed ISP bandwidth. With no drops at user-end, network congestion in the ISP network remains a cause of frame losses for users.

Figure 6 shows average results of 20 runs for 1 million users requesting one of 15 available video streams each served at three different bitrates. We decreased the capacity available at all the links in the network from 40*Mbps* to 10*Mbps* and measured the percentage of users that were assigned a bitrate correctly in both STAR and MESH topology. At 40*Mbps* there was little congestion in the network and mCast served most users correctly (95% for MESH and 90% for STAR topology).

As the link capacities started decreasing and congestion occurred, mCast failed to respond properly and the percentage of correctly handled users kept on decreasing. The users in STAR topology, with no redundant paths suffered more than users in MESH topology. Because of its awareness of device specifications, ISP bandwidth limitations as well as any congestion in the network, Danos was able to handle all such scenarios and avoided any frame losses for users by assigning the best bitrate that could be delivered to them. Hence, in both MESH and STAR topology, Danos assigned 100% users with their best bitrates.

In addition to showing the gains of Danos over mCast, these results present another key message. An efficient and well-designed network-layer multicast approach can serve potentially unlimited number of users (1 million in this analysis) with very low link capacities in the network. As Figure 6 shows, when network links had only 10*Mbps* available for Danos, it still managed to find a solution where none of the million users suffered any frame losses.

5 EXPERIMENTAL EVALUATION

We implemented a prototype system of Danos for demonstration and evaluation. In this section, we present the details of the prototype and evaluate the performance of Danos in real-world scenarios.

5.1 Prototype Implementation

Figure 8 shows our prototype implementation. We consider SDN-based domains for both CDN and ISP networks. We use Mininet Emulator v2.3.0 [1] to emulate network topologies and client devices. For the CDN domain, we use a simple tree-based topology which is the most common approach for data-centers [3]. For the ISP domain, we use two ISP topologies from Topology Zoo database [26] as mentioned in Section 4.4.

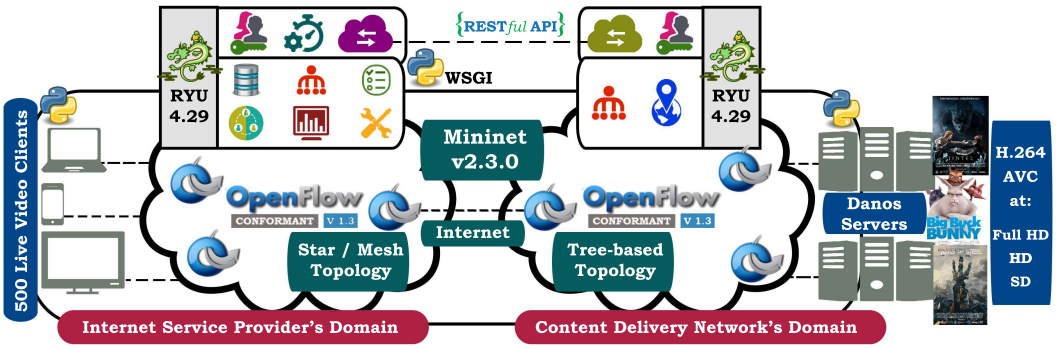


Figure 8: Prototype implementation of Danos over an emulated testbed.

One ISP ingress switch provides a gateway to the CDN, creating one entry point for all the video streams. We set the capacity of this link to 100Mbps, allowing enough bandwidth for all the videos and bitrates to be delivered to the ISP network. We consider the scenario where the CDN does not opt to use ISP transrating services and hence we do not implement the Transrater module. If used, it would reduce the traffic flow from the CDN to the ISP.

We use Ryu controller [8] to implement all the remaining modules of Danos (Figure 1) in the control and application plane of both domains and choose OpenFlow v1.3 protocol for the SDN southbound interface. We use OpenvSwitch v2.10 switches in both domains with default queue sizes for each link, which we found sufficient with $\alpha = 0.8$ (Equation 1a), for smooth switching as discussed in Section 3. The Danos ISP Agent runs as a web server built over the python Web Server Gateway Interface (WSGI) and the Danos CDN Agent communicates with ISP using a RESTful API defined the by the ISP Agent.

The optimization framework runs periodically every 5 seconds and solves our guided optimization model (Section 4.3) using Gurobi solver [2]. We use proportional fairness as the system utility, defined as sum-log of bitrates assigned to users i.e. $f(m,r) = log(r_m)$ in Equation 1a, where r_m is the bitrate assigned to user m.

We encode three raw videos "Big Buck Bunny" (bbb), "Tears of Steel" (tos) and "Sintel", using H.264 AVC at three different resolutions, $1920x1080$, $1280x720$ and $480x270$ with a GOP size of eight and a frame rate of $25fps$, yielding approximate bitrates of $4Mbps$, $1.5Mbps$ and $400kbps$ respectively. We add three Danos streaming servers to the CDN, each streaming one of the video at the three mentioned bitrates.

We implement our Danos servers and video clients in python2.7. The clients send requests to join a video stream at a specific bitrate and rather than live decoding, save a log of the recieved content for post-processing. This allows us to scale the number of clients that can run simultaneously in our emulated testbed. We run one live video client on each user created in the Mininet topology and test up to 500 clients.

The Danos streaming servers run three threads: a **listening thread** to listen for content requests and establish a connection with clients; a **control thread** that uses an API to communicate with the Danos CDN Agent and takes instructions or provides the requested client information and; a **data thread** that streams a video at different bitrates and serves each client at their maximum allowed or requested bitrate.

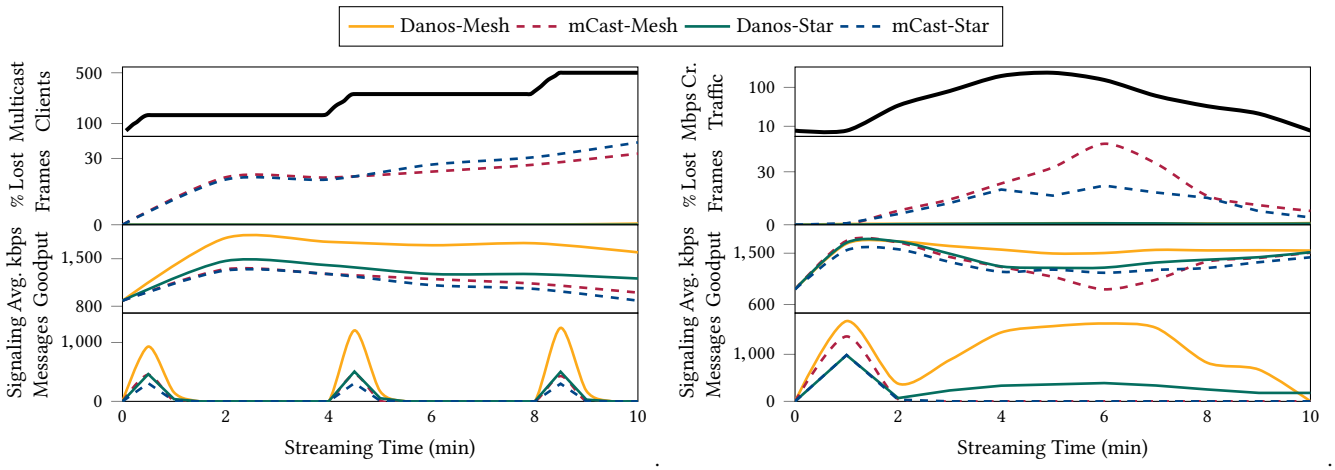
We evaluate Danos performance in two key scenarios: flash crowd and cross traffic and compare it with mCast [13], a state-of-the-art approach that has a broadly similar architectural view to Danos. Our performance metrics include: **Lost frames** defined as the number of video frames that were partially or completely dropped in the network; **Average goodput** which is the rate at which useful data arrived at client devices. Useful data includes the number of bytes in GOPs received completely by a client. **Probability Mass Function** (PMF) of the bitrates assigned to users. The PMF serves as an indicator of overall system performance by providing an idea of how many users can receive a certain bitrate in given conditions. Regarding the overhead of Danos, we record the number of **Signaling messages** shared between the SDN controller and the OpenFlow switches in the ISP network.

5.2 Flash Crowd Scenario

The nature of live video streaming, and in particular, rising popularity of user-generated live content has resulted in frequent occurrence of flash crowd events with large numbers of user arrivals in a short period of time. To determine how Danos would react to such events, we assume that the ISP assigns a slice of $12Mbps$ capacity on network *access links* for Danos sessions. Videos sessions last for 10 minutes during which three flash-crowd events occur by adding around 150 clients within a 30-second window at the beginning, middle and towards the end of the stream. We run this experiment five times by uniformly varying user locations and device specifications and show the average results of the five trials in Figure 9a.

Users request the highest bitrate that they can support, depending on various device, ISP or CDN constraints. As the number of users increase, with three videos each at three bitrates, this can create up to nine concurrent streams in the access network which would exceed the available link capacities ($12Mbps$). mCast is unaware of the network state and resulted in congesting the network, causing up to 37% video frame losses for users, and affecting the average goodput of the system.

As Danos ran our optimization framework that considered network constraints, it reacted to flash-crowd events efficiently by balancing the load across redundant paths in the MESH topology and eliminating frame losses. This helped Danos to also stream better video quality to its users, in comparison to mCast, as illustrated in Figure 10a. In cases of no alternate paths, Danos reduced the



(a) Flash Crowd Scenario **(b) Cross-traffic Scenario**

Figure 9: Results of experiments for three video streams each served at three different bitrates.

bitrates of the affected clients to handle congestion. This is evident in the STAR topology where no redundant paths are available and Danos reduced the bitrates (Figure 10a), and hence the average goodput of the clients (Figure 9a), to avoid frame losses.

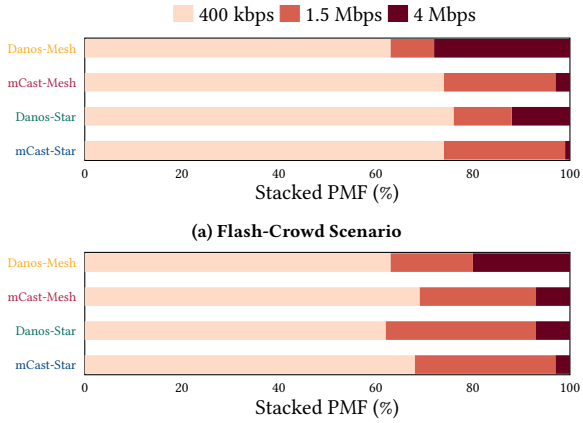
The cost of active network agents comes in the form of signaling overhead messages. When Danos re-configures network nodes, OpenFlow messages are sent and we can see more messages than mCast, which only re-configures when a client joins or leaves a stream. However, as shown in results (Figure 9a), Danos generated around 1000 messages over a 1-minute duration for flash crowd events. This is considered low for SDN switches which can usually handle thousands of flow modification messages per second [17].

5.3 Cross-Traffic Scenario

In this experiment, we consider no link slicing and use 30*Mbps access links* shared among 500 Danos users and up to 300 unicast users in the ISP network. All the Danos users request one of the three videos mentioned above, within the first minute of the streaming duration. We consider cross-clients that arrive based on a normal distribution, connect randomly to switching nodes of the ISP, uniformly choose 1, 2 or 4 Mbps as download rates and stay active for 1-minute each. The cross-traffic generated in the network is shown in Figure 9b and follows a Gaussian distribution which is a commonly adopted model for Internet traffic [21]. The results shown are averaged over 5 runs.

With no or low cross-traffic towards the beginning and end of the streaming duration, both Danos and mCast assigned users with their highest supportable bitrates and induced no frame losses in MESH or STAR topology. As the cross-traffic started increasing and the network links got congested, mCast failed to react and tried to serve multicast users with the same bitrates, resulting in up to 45% frame losses at the peak cross-traffic.

Danos, similar to flash crowd events, responded by rerouting traffic where possible, especially in MESH topology, and reducing user bitrates otherwise. This also resulted in higher goodput rates for Danos where it was capable of serving 20% users with highest and 17% users with medium bitrate for the MESH topology. As the cross-traffic started decreasing, the percentage of lost frames



(a) Flash-Crowd Scenario

(b) Cross-Traffic Scenario

Figure 10: PMF of average user-bitrates.

decreased in mCast and the goodput increased. Danos, being aware of the network state, also reacted by further increasing bitrates for users while maintaining no frame losses.

Even with the dynamic configuration of the network, the signaling overhead was reasonable for Danos as shown in the Figure 9b. Furthermore, due to its stable response mechanism, Danos did not inflict excessive bitrate switching on clients and the total number of switches for all 500 of its clients was at most 100 over the complete streaming duration of 10 minutes.

6 RELATED WORK

Existing multicast approaches for live video streaming can be classified into three main categories.

IP multicast-based intra-domain streaming: As IP multicast faces practical deployment, management and security issues across domains [16], its implementation is limited to services within a certain network. For example, ISP-oriented IPTV services [5] or cable networks (DOCSIS [10]) can pre-provision users and statically configure their infrastructures to deliver live content using multicast. Even in its limited scope, IP multicast is incapable of providing

dynamic services such as adaptive bitrate streaming due to its rigid and complex design and lack of efficient congestion handling [7].

Overlay multicast streaming: To enable a high control granularity, peer-to-peer (P2P) streaming services implement multicast at the application layer but are oblivious to the underlying network state [12]. Prior research reported that P2P live streaming suffers from unstable video quality and playback lags due to peer churn and limited uplink bandwidth of end-users [15]. Recent efforts, such as AngelCast [22], augment overlay multicast with CDN clouds to reduce the latency and improve the streaming quality, but the inability to handle congestion in the underlying infrastructure repels users with dynamic network conditions and need for complex client-side algorithms makes it difficult to attract users watching live streams on low-end devices.

Multicast in programmable networks: The advent of SDN and network function virtualization (NFV) has revitalized the demand of multicast at the network layer. LiveJack [25] presents a network service that allows CDN servers to leverage ISP cloud resources and extend multicast towards the edge. SDM [20] proposes a framework that enables ISPs to support resource efficient peer-to-peer streaming. mCast [13] proposes an architecture that enables network-layer multicast through ISP-CDN collaboration using a communication framework to create dynamic multicast trees and provide CDNs will full control over their clients. These solutions reduce the complexity of deploying and managing multicast, making inter-domain streaming feasible. However, they do not present a mechanism for enabling adaptive bitrate streaming for multicast sessions and do not address some key challenges that arise in live streaming services, such as heterogeneous capabilities of user devices or network congestion due to cross-traffic.

In addition to solving the aforementioned problems, we also presented a novel optimization model for live multicast streaming services. Multicast optimization models for other network architectures, e.g. RTOP [14] for cellular networks, can complement our work by optimizing delivery in the edge network, while our model optimizes the core and wired-access networks. Similar work, such as NOVA [23] and SABR [4], exist for streaming over DASH with network-assisted content distribution. However, these models consider unicast-based delivery mechanisms which do not apply to multicast-based streaming scenarios, as decisions made for multicast users are inter-dependent and choosing a certain bitrate for a user can have a direct impact on the cumulative utility.

7 CONCLUSION

In this paper, we presented Danos, an optimal live streaming service that is aware of device and network state and uses SDN and its global knowledge to enable resource-efficient network-layer multicast over the Internet. We designed an architecture and discussed various design issues and choices for ISPs and CDNs. We formulated an optimization model that can build optimal paths in real-time and ensure high video quality for users by minimizing frame losses or switches and maximizing the assigned bitrates. Our performance analysis showed that the model is extremely scalable and can solve the problem in order of milliseconds. We implemented a prototype for Danos and used it to compare and evaluate Danos against a state-of-the-art approach in real-world scenarios such as flash crowd and cross-traffic. Danos handled both events efficiently and improved average goodput by up to 70% while almost eliminating video frame losses for clients.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Mininet 2.3.0. 2018. http://mininet.org/
[2] Gurobi Solver 8.1. 2018. http://www.gurobi.com/
[3] M. Bari, R. Boutaba, R. Esteves, L. Granville, M. Podlesny, M. Rabbani, Q. Zhang, and M. Zhani. 20013. Data Center Network Virtualization: A Survey. *IEEE Communications Surveys & Tutorials* 15, 2 (20013), 909–928.
[4] D. Bhat, A. Rizk, M. Zink, and R. Steinmetz. 2017. SABR: Network-Assisted Content Distribution for QoE-Driven ABR Video Streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference (MMSys)*. ACM, 62–75.
[5] M. Cha, P. Rodriguez, J. Crowcroft, S. Moon, and X. Amatriain. 2008. Watching Television over an IP network. In *In Proceedings of the 8th ACM SIGCOMM conference on Internet measurement (IMC)*. ACM, 71–84.
[6] J. Chen, M. Chiang, J. Erman, G. Li, K. Ramakrishnan, and R. Sinha. 2015. Fair and optimal resource allocation for LTE multicast (eMBMS): Group partitioning and dynamics. In *IEEE Conference on Computer Communications (INFOCOM)*. IEEE.
[7] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen. 2000. Deployment issues for the IP multicast service and architecture. *IEEE Network* 14, 1 (2000), 78–88.
[8] Ryu SDN framework. 2017. https://osrg.github.io/ryu/
[9] M. Garey and D. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York.
[10] J. Godas, B. Field, A. Bernstein, S. Desai, T. Eckert, and H. Parandekar. 2005. *IP Multicast In Cable Networks (White Paper)*. Technical Report. Cisco Systems.
[11] E. Haleplidis, K. Pentikousis, S. Denazis, J. Salim, D. Meyer, and O. Koufopavlou. 2015. *Software-Defined Networking (SDN): Layers and Architecture Terminology*. https://www.rfc-editor.org/rfc/rfc7426.txt.
[12] M. Hosseini, D. Ahmed, S. Shirmohammadi, and N. Georganas. 2007. A Survey of Application-Layer Multicast Protocols. *IEEE Communications Surveys & Tutorials* 9, 3 (2007), 58–74.
[13] A. Khalid, A. Zahran, and C. Sreenan. 2017. mCast: An SDN-Based Resource-Efficient Live Video Streaming Architecture with ISP-CDN Collaboration. In *IEEE 42nd Conference on Local Computer Networks (LCN)*. IEEE, 95–103.
[14] A. Khalid, A. Zahran, and C. Sreenan. 2019. RTOP: Optimal User Grouping and SFN Clustering for Multiple eMBMS Video Sessions. In *IEEE INFOCOM*. IEEE.
[15] Y. Liu, Y. Guo, and C. Liang. 2008. A survey on peer-to-peer video streaming systems. *Peer-to-Peer Networking and Applications* 1, 1 (2008), 18–28.
[16] M. Moyer, J Rao, and P. Rohatgi. 1999. A survey of security issues in multicast communications. *IEEE Network* 13, 6 (1999), 12–23.
[17] NoviFlow. 2019. Retrieved Feb 07, 2019 from https://noviflow.com/products/noviswitch/
[18] K. Pires and G. Simon. 2015. YouTube live and Twitch: a tour of user-generated live streaming systems. In *Proceedings of the 6th ACM Multimedia Systems Conference (MMSys)*. ACM, 225–230.
[19] Cisco Public. 2018. *Cisco Global Cloud Index: Forecast and Methodology, 2016-2021 (White Paper)*. Technical Report.
[20] J. Ruckert, J. Blendin, and D. Hausheer. 2015. Software-Defined Multicast for Over-the-Top and Overlay-based Live Streaming in ISP Networks. *Journal of Network and Systems Management* 23, 2 (2015), 280–308.
[21] R. Schmidt, R. Sadre, and A. Pras. 2013. Gaussian traffic revisited. In *Proceedings of IFIP Networking*. 1–9.
[22] R. Sweha, V. Ishakian, and A. Bestavros. 2012. Angelcast: Cloudbased Peer-assisted Live Streaming using Optimized Multi-tree Construction. In *Proceedings of the 3rd Multimedia Systems Conference (MMSys)*. ACM, 191–202.
[23] R. Sweha, V. Ishakian, and A. Bestavros. 2014. NOVA: QoE-driven optimization of DASH-based video delivery in networks. In *IEEE Conference on Computer Communications (INFOCOM)*. IEEE.
[24] B. Wang, X. Zhang, G. Wang, H. Zheng, and B. Zhao. 2016. Anatomy of a Personalized Live Streaming System. In *Proceedings of the 2016 Internet Measurement Conference (IMC)*. ACM, 485–498.
[25] B. Yan, S. Shi, Y. Liu, W. Yuan, H. He, R. Jana, Y. Xu, and H. Chao. 2017. LiveJack: Integrating CDNs and Edge Clouds for Live Content Broadcasting. In *Proceedings of the 2017 ACM on Multimedia Conference (MM)*. ACM, 73–81.
[26] The Internet Topology Zoo. 2013. Retrieved Feb 03, 2019 from http://www.topology-zoo.org/