MULTICAST-ASSISTED UNICAST DELIVERY

M. E. Nilsson, R. S. Turnbull, T. S. Stevens and S. Appleby

BT Research and Network Strategy, UK

ABSTRACT

There is growing use of the Internet to deliver television services by unicast. Delivering an individual stream for each viewer causes very popular content to drive large peaks in network traffic, which drives the need for network upgrades just to support the peaks.

We describe Multicast-Assisted Unicast Delivery (MAUD), the innovative approach we have developed to use multicast to assist with the unicast delivery of content, allowing content preparation and client applications to remain unchanged while reducing the large peaks in network traffic.

We also describe the trial we have performed with the CDN operator Qwilt where a Raspberry Pi was deployed in the home network of about 90 BT trialists, some of whom used the BT Sport app to request live BT Sport content, while others ran a robot-client on the Raspberry Pi.

Integration with the CDN Operator was straightforward, and analysis of performance statistics showed that the system behaved as expected. We plan further work towards ultimately deploying MAUD as a network-optimisation, cost reduction, technology.

INTRODUCTION

The delivery of audio-visual content is moving from traditional broadcast networks, including satellite, cable and terrestrial networks, to IP networks, where unicast delivery is often used to deliver a single copy of a content item to a viewer's device, even when many people are viewing the same content at the same time. This drives ever higher demand on broadband IP networks, and ever higher capital expenditure to increase network capacity. If IP multicast could be used to deliver content that is viewed by many viewers at the same time, then the demand on broadband IP networks could be reduced, which would lead to reduced or delayed capital expenditure on increased network capacity.

However, IP multicast is rarely used for any services other than delivery of a network operator's on-net television channels to their own set-top boxes because multicast does not lend itself to open use on the Internet. Hence to bring the benefits of multicast to unicast streaming, a class of techniques known as Multicast Adaptive Bit Rate (m-ABR) has been studied and standardised, but to date not widely deployed.

We describe some characteristics of current approaches that may have affected the rate of deployment, and then describe the novel approach to m-ABR called Multicast-Assisted Unicast Delivery (MAUD) that we have designed to enable more widespread use. We then report the trial we have performed with the CDN operator Qwilt to validate the MAUD design.

CHARACTERISTICS OF CURRENT APPROACHES TO M-ABR

In an m-ABR system, multicast is used along part of the content path, while the viewer's device continues to request and receive content by unicast, allowing the peak network traffic to be reduced, while avoiding the need for end devices to be upgraded to support multicast.

The DVB have defined a functional m-ABR architecture, DVB BlueBook A176 (1) "for delivering live and on-demand adaptive streaming media to a large audience in a highly efficient and scalable way by leveraging IP multicast for media object delivery".

The DVB m-ABR system specifies a means for using multicast to support the delivery of ABR content from the source to the client. The specification, which does not specify a single system design, enables a range of options for implementation to enable a single Content Service Provider (CSP) to deliver their own content to their own applications.

The application may obtain the presentation manifest URL, using, for example, a Service directory function, where the URL points to the Multicast rendezvous service, which maintains records of the status of Multicast gateways and multicast sessions. This allows the Multicast Gateway to be inserted into the client's content delivery path. Alternatively, the application must be modified to locate the Multicast Gateway using local service discovery.

The Multicast server is required to understand the presentation manifest in order to push the appropriate content over the respective multicast channels, and in one possible implementation, to be able to request the content components from the content source. In addition, the Multicast Gateway may be instructed to modify the presentation manifest, which in turn implies the need to have access to and understand the presentation manifest.

While this may all be practical for a CSP delivering their content to their own applications, the situation is more complex for a Network Service Provider who delivers content from a range of CSPs to their customers who use a range of applications, typically one per CSP.

This situation is worsened by the rights for valuable live content changing from one organisation to another frequently and unpredictably. It is vital for the Network Service Provider to be able to use m-ABR as soon as an organisation starts to deliver newly acquired content to avoid overloading the network. Recent examples of CSPs acquiring rights to live events or believed to be considering the acquisition of rights include: Apple providing Major League Baseball in the UK (2), Netflix considering offering live sports (3) and the comedian Chris Rock performing live (4), and Viaplay offering live sports including football (5)(6).

A Network Service Provider's Vision for Multicast ABR

The m-ABR solution must be commercially viable, meaning that there must be business incentives to support m-ABR for all the organisations involved in the delivery of the content. This includes not only the CSP, but the Network Service Provider and the CDN operator.

To make the m-ABR solution resilient to content rights changing hands and avoid the need to continuously support additional CSPs and their applications, the solution needs to be isolated as much as possible from this unpredictability.

The solution should be independent of the ABR streaming format, other than the basic assumption of the use of HTTP request/response pairs. By avoiding the need to parse or modify manifests, the system becomes resilient to changes in the details of the ABR streaming formats that are used.

Integration with the CSP should be minimal. The CSP should not be required to modify their content preparation workflow or their client application, nor should they be required to integrate directly with the m-ABR solution. In many cases the Network Service Provider may not be able to influence the formats used by global CSPs.

The CSP should continue to benefit from the features provided by the CDN, such as logging, analytics, and restricting access through authentication or geo-restrictions. Client and session-specific information should be preserved: the content requests to the CDN and the responses delivered to the clients must be very similar to those when m-ABR is not used.

The m-ABR solution must be dynamic. It should be possible to enable the use of multicast flexibly, with the decision informed by both content popularity and detailed network knowledge. The use of multicast for a specific client must also be dynamic, considering factors including the stability of the client's ABR bit rate selection and the likelihood of causing quality degradation when switching from unicast to multicast delivery.

MULTICAST-ASSISTED UNICAST DELIVERY (MAUD)

BT has developed a prototype m-ABR system called Multicast-Assisted Unicast Delivery (MAUD) to verify the feasibility of the vision. The name is derived from the principle of using multicast to assist with the unicast delivery of content. Content preparation and client applications are unchanged, with the latter being unaware of the use of multicast, which could simply be considered as a means to optimise network performance.

In this section we provide an overview of the MAUD system which has many similarities with the DVB system, but also some critical differences to support the vision described above. The MAUD system architecture is shown in Figure 1. The reference points are given the same labels as the reference points in the DVB mABR reference architecture defined in DVB BlueBook A176 (1). An additional two reference points, O_{in}' and Q, are included.

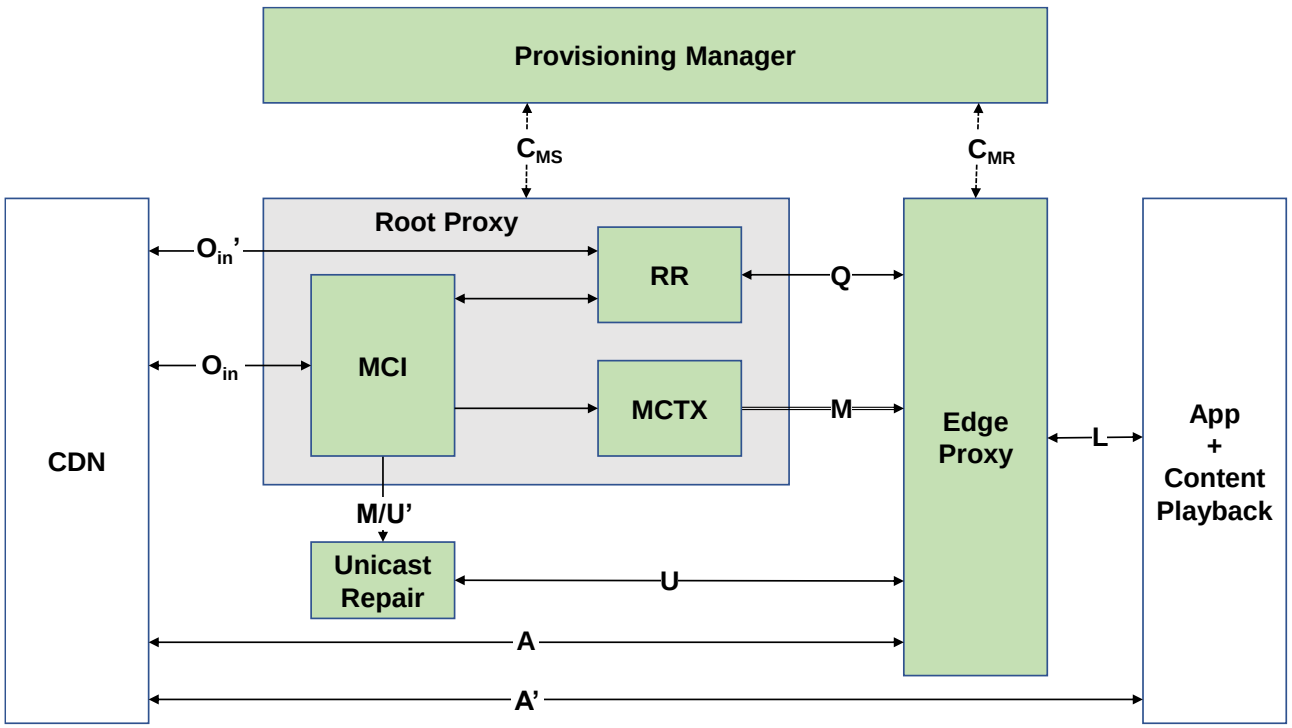

Figure 1 – MAUD System Architecture

Overview

The Root Proxy is located at the root of the multicast tree. One function of the Root Proxy is like the Multicast server in the DVB system: to obtain content, format it for multicast and then transmit it by multicast. This is shown in Figure 1 where the Multicast Ingestion (MCI) function obtains content from the CDN, and the Multicast Transmitter (MCTX) function transmits the content payload by multicast. But the Root Proxy has additional functions, as described below, performed by the Request Router (RR) function using the additional reference points, O_{in}' and Q.

The Edge Proxy performs similar functionality to the Multicast Gateway in the DVB system. It is hosted in the home gateway at the end of the multicast tree and performs functions including multicast to unicast conversion. The Edge Proxy communicates with the client application at reference point L using HTTPS.

The Provisioning Manager performs a similar role to the DVB Provisioning Manager: it receives reports from Root Proxies and Edge Proxies, makes decisions using the information in these reports, and disseminates multicast and session configuration information back to the Proxies.

However, the MAUD system differs from the DVB m-ABR system in several important aspects. To avoid modifying the client application, the Edge Proxy is inserted into the content path either by the CSP, who would provide a suitable entry in their Content Management System or CDN selector, or by the CDN operator, who would re-direct requests to the Edge Proxy. This avoids the need to modify the client application since there is no requirement for the client application to discover the presence of the Edge Proxy and avoids the need to deploy a Multicast rendezvous service.

In the DVB m-ABR system, when the Multicast Gateway receives a segment request, it serves it from its cache if it is stored, otherwise fetches it from the CDN. In the former case, the CDN and the CSP are unaware that the client made the request, although the Multicast Gateway may occasionally send high level summary reports to the Provisioning Manager.

In the MAUD system, when the Edge Proxy has joined a multicast channel and received a request for a segment stored in its cache, rather than replying immediately to the client with the cached content as in the DVB system, it changes the GET request to a HEAD request and forwards it to the CDN. It combines the response from the CDN with the cached payload received by multicast to form a complete response which is returned to the client.

This approach of delivering the payload by multicast and the header by unicast and combining them in the Edge Proxy helps to achieve several aspects of the vision.

It gives the CDN operator almost the same level of visibility and control of client requests as with normal unicast streaming. This enables the CDN operator to perform normal logging, analytics, and reporting services on behalf of their customers. Further, the CDN operator can control access to content. For example, if token-based access is being used then the client application would need to include the appropriate token in its request. If it does not, then the CDN would return a status code to indicate this, and the Edge Proxy would not release the content to the client application. This contrasts with the DVB m-ABR system, where the CDN operator would have no visibility or control of client requests.

Additionally, by passing request and response headers back and forward by unicast, the MAUD system ensures that any header attributes, such as cookies or other tokens, will be identical to those of unicast streaming.

In the MAUD system, when the Edge Proxy has joined a multicast channel and received a request for a segment that is not stored in its cache, it forwards the GET request to the Root Proxy at reference point Q. As described below, the Root Proxy may instruct the Edge Proxy to get the segment from the CDN, or may return the just the header, because the payload has been or will be delivered by multicast to the Edge Proxy.

By passing client requests through to the CDN to obtain content, the MAUD system is largely independent of the streaming protocol used. There is no content ingestion function in the Root Proxy which needs to understand the protocol to obtain content from the CDN or CSP.

The Provisioning Manager

The Provisioning Manager receives reports from Root Proxies and Edge Proxies for one or more “routes”, makes decisions using these reports, and reports back to the Proxies. A route is associated with a single ABR representation of a content stream where multicast may be used to assist delivery. The Provisioning Manager publishes the following for each route:

- Active flag indicates whether multicast is currently being used for the route
- Multicast address indicates the address used for multicast delivery
- A regular expression used by an Edge Proxy to match a client segment request with the ABR level and content stream of the route
- Address of the Root Proxy an Edge Proxy should communicate with
- Address of the unicast repair service to use to obtain data lost in multicast delivery

The Provisioning Manager receives reports from Edge Proxies indicating, for each route, the number of segment requests from clients that have matched the route’s regular expression. This enables the Provisioning Manager to determine the relative popularity of the content streams associated with each route. It considers this information and determines for which routes multicast should be used. In the case of limited multicast resources, this allows dynamic allocation of multicast resources between the routes. For example, a multicast channel could be used for a sports channel during a live event when the audience is large and used for another channel when the audience for the sports channel is lower.

The Edge Proxy

Each Edge Proxy periodically obtains route information from the Provisioning Manager.

When an Edge Proxy receives a segment request from a client, it applies the regular expression of each route to the URL requested by the client to determine whether the URL “matches the route”. The Edge Proxy periodically sends reports to the Provisioning Manager indicating the number of requests it has received with a URL that matches each route.

If the URL does not match any route, or if the URL does match a route but the active flag is false, the Edge Proxy forwards the GET request to the CDN, and responds to the client with the response from the CDN.

If the URL matches a route and the active flag is true, but the Edge Proxy has not joined the route’s multicast channel, the Edge Proxy makes the decision of whether to join the channel.

If the URL does match a route, the active flag is true, and the Edge Proxy has joined the multicast channel, it searches its cache for a segment that matches the requested URL. If there is a match, the Edge Proxy changes the GET request to a HEAD request and sends it to the CDN at reference point A. It then combines the HEAD response with the segment stored in its cache to create a full response which it sends to the client.

But if there is no match in the cache for the requested URL, the Edge Proxy sends the GET request at reference point Q to the Root Proxy, which, as described below, may instruct the Edge Proxy to get the segment from the CDN, or, forwards the request to the CDN, and returns just the header of the response from the CDN by unicast.

An Edge Proxy that had joined a multicast channel may later decide to leave the channel.

The Root Proxy

When a Request Router in the Root Proxy receives a GET request from an Edge Proxy, and the requested segment has not been multicast, the Root Proxy either instructs the Edge Proxy to get the segment from the CDN, or the Multicast Ingestion function forwards the request to the CDN, and the Request Router delivers the header of the response to the Edge Proxy by unicast and the Multicast Transmitter delivers the payload of the response by multicast where the payload includes an identifier to associate the payload with the requested URL. The Request Router is horizontally scaled according to the demand from Edge Proxies.

When a Request Router receives a GET request from an Edge Proxy after the requested segment has been multicast, it changes the request to a HEAD request, forwards it to the CDN at reference point O_{in}', and returns the response to the Edge Proxy. This happens when the requested segment is not in the Edge Proxy cache, for reasons including:

1. The Edge Proxy received the request from the client before it had received any or sufficient data for the segment to be able to associate it with the request
2. The Edge Proxy may have recently joined the multicast channel, joining after the requested segment had been transmitted on the multicast channel
3. The client may have requested the segment sufficiently behind the live edge that the segment, although having been received by multicast and stored in the cache, is no longer stored in the Edge Proxy cache

The first case may occur most frequently, especially when many clients request segments associated with the route very close to the live edge. Many such requests could be made in the time it takes for the Root Proxy to receive a request from an Edge Proxy, to obtain the segment from the CDN, to decide to deliver it by multicast, to format it for multicast delivery and for it to be delivered over the multicast channel.

And in this first case, by the time that the Edge Proxy has received the response from the Root Proxy containing just a header, it is likely that enough of the segment body has been received by multicast for the Edge Proxy to identify it as corresponding to the header, and to form a complete response which is then returned to the client.

Otherwise, including in the second and third cases, the Edge Proxy does not have a segment body to combine with the header received from the Root Proxy, and so sends the original GET request to the CDN, receives a complete response and forwards it to the client.

Message flows to include the Edge Proxy in the content delivery path

The following describes one method that could be used in the MAUD system to insert an Edge Proxy into the content delivery path.

The user runs a CSP application, identifies content to watch and clicks play, causing a request to be made to the CSP's Content Management System. Normally this would return a list of one or more CDNs that the application could use to obtain the presentation manifest

and the segments. In the MAUD system the response includes a URI that identifies a MAUD host as the location from which to obtain the presentation manifest.

The following occurs when there is a MAUD Edge Proxy available to the client. The client makes a DNS query for the MAUD host to the local DNS server, running for example on the home gateway device. This DNS server maps this to the address of the Edge Proxy, which may also be running on the home gateway device. The client then requests the presentation manifest from the Edge Proxy, which makes a DNS query to a WAN (not local) DNS server. This is configured to map the MAUD host to a CDN IP address. The Edge Proxy then requests the presentation manifest from the CDN and returns it to the client.

The following occurs when there is not a MAUD Edge Proxy available to the client. The local DNS server cannot resolve the query for the MAUD host because there is no Edge Proxy available in the home gateway device. The client then makes a DNS query for the MAUD host to a WAN (not local) DNS server. This responds in the same way as above, mapping the MAUD host to a CDN IP address, but this time returning this to the client, which then requests the presentation manifest from the CDN.

Summary of the technical differences between the DVB mABR and MAUD systems

As stated above, the reference points common to both the DVB mABR system and the MAUD system, as shown in Figure 1, are used in essentially the same way in each system. The only significant difference is that in the MAUD system, reference point A, in addition to being used as in the DVB system to obtain segments that have not been received by multicast and to repair segments received by multicast, it is also used for HEAD requests for segments that have been received by multicast and cached at the Edge Proxy.

The MAUD system includes two additional two reference points O_{in}' and Q. The usage of these reference points has been described above and is summarised as follows.

Reference point Q is used for GET requests from the Edge Proxy to the Root Proxy when the requested segment is not present in the cache at the Edge Proxy, and for the response to the Edge Proxy which may be a HEAD response or a redirect to the CDN.

Reference point O_{in}' is used for HEAD requests from the Root Proxy to the CDN and the corresponding responses, following the Root Proxy receiving a GET request from an Edge Proxy at reference point Q.

MAUD SYSTEM TRIAL

We now describe a proof-of-concept trial that we carried out during the summer of 2022 with the CDN operator Qwilt and our software implementation of the MAUD System.

Aims of the Trial

The aims of the trial included the following:

- Assess the complexity of CDN integration – is it as straightforward as expected?
- Confirm request routing to insert the Edge Proxy into content path works as expected
- Assess whether use of multicast causes any undesirable client behaviour
- Assess resources required to run the Edge Proxy – could it run on a home gateway?
- Assess client application compatibility – which players operate as normal when multicast is inserted in the delivery path, and what prevents other players doing so?
- Assess security aspects – to ensure that MAUD can be deployed in a secure manner

Trial Architecture

We implemented the “Provisioning Manager” as seven micro-services hosted on the BT Research Platform to instruct individual Raspberry Pis to perform specific experiments, receive performance reports, and monitor the overall system.

The Root Proxy was hosted securely in BT premises in London. A dedicated national multicast channel was provisioned at 12MBit/s and used to deliver MAUD content. The Root Proxy was configured to deliver a segment by multicast when a request is received for a segment that has not previously been multicast and the multicast channel is available.

The Edge Proxy was implemented on a Raspberry Pi and deployed in the home network of about 90 BT trialists, with the home network being connected to the Internet as normal through a BT (or other) home gateway device. Some trialists used the unmodified BT Sport app to request live BT Sport content, while others ran our robot-client (“Bot”) on the Raspberry Pi to emulate users and client applications requesting the same content.

The content management system signalled if multicast is available in a query parameter of the response giving the presentation manifest URL to the BT Sport app, while the Bot was simply given this URL with the query parameter. The CDN redirected requests with this query parameter to the Edge Proxy running on the Raspberry Pi in the trialist’s home. The Edge Proxy joined the appropriate channel on the BT National Multicast network and satisfied the player’s segment requests using multicast-delivered data whenever possible.

The Raspberry Pi ran three components: the Edge Proxy, the Client Controller and the Bot. These could be updated remotely using ansible, git and bash. The Bot periodically polls the Experimental Script Server and receives a JSON file specifying an experiment. The Client Controller consequently instructs an HLS player that gets manifest files, makes ABR decisions and requests segments by unicast. The Edge Proxy receives requests from the HLS player, receives content, either by multicast or unicast, and satisfies the requests by unicast. It joins and leaves the multicast channel as specified in the experiment. It records and reports the results.

Example Experiment

We now provide an illustrative example of an experiment run in this trial.

Figure 2 shows an example of the JSON files that the Bots fetched periodically. The “config” element specifies that the Bot should check every 60s for an update to this file, and that the HLS client should request segments about one segment behind the live edge. The “experiment” element specifies the experiment. The start time is the sum of “start_time” and a delay selected using a uniform probability distribution between zero and “start_jitter”.

```
{
  "config":{
    "poll_interval":     60,
    "seg_offset":        1 },
  "experiment":{
    "start_time":        "2022-06-09 16:50",
    "description":       "Baseline test",
    "start_jitter":      0,
    "route":{
      "CohortID":        202 },
    "master_url":        "https://t1-live-euwe3.us.
      qw.streams.test.sport.bt.com/bts1-fp/
      bts1-fp.isml/bts1-fp.m3u8? maud=true",
    "events":[
      {"t":0, "id":"38,42,172", "event":"start"},
      {"t":30, "id":"38", "event":"join"},
      {"t":60, "id":"42", "event":"join"},
      {"t":240, "id":"38,42", "event":"leave"},
      {"t":250, "id":"38,42,172", "event":"stop"}]}
}
```

Figure 2 – Example experimental JSON script

The multicast channel is specified by “CohortID”, and the m3u8 URL by “master_url”. The “events” element specifies the actions to be taken during the experiment. Each has a start time, relative to when the HLS client started and a list of Raspberry Pi IDs that the action applies to. In this example, at time t=0, the IDs 38, 42 and 172 are instructed to start. ID 38 is instructed to join the multicast channel at time t=30s, and ID 42 at time t=60s. Both are instructed to leave the multicast channel time t=240s. Ten seconds later all three are instructed to finish the experiment. The other Raspberry Pis in the trial received the same JSON file and recognised that there were no actions to take during the experiment.

Figure 3 shows a graphical representation of the output from the experiment. The charts are colour coded as follows. The green vertical line shows when an Edge Proxy joined the multicast channel, and the red vertical line shows when it left. The coloured horizontal lines show the delivery of segments by unicast from the Edge Proxy to the HLS client, with the line going from the start of delivery to the end of delivery, so that fast delivery shows as a dot. The different colours show the different ABR levels, but in this experiment all segments were delivered at the highest ABR bit rate of 6.5MBit/s and are shown in red.

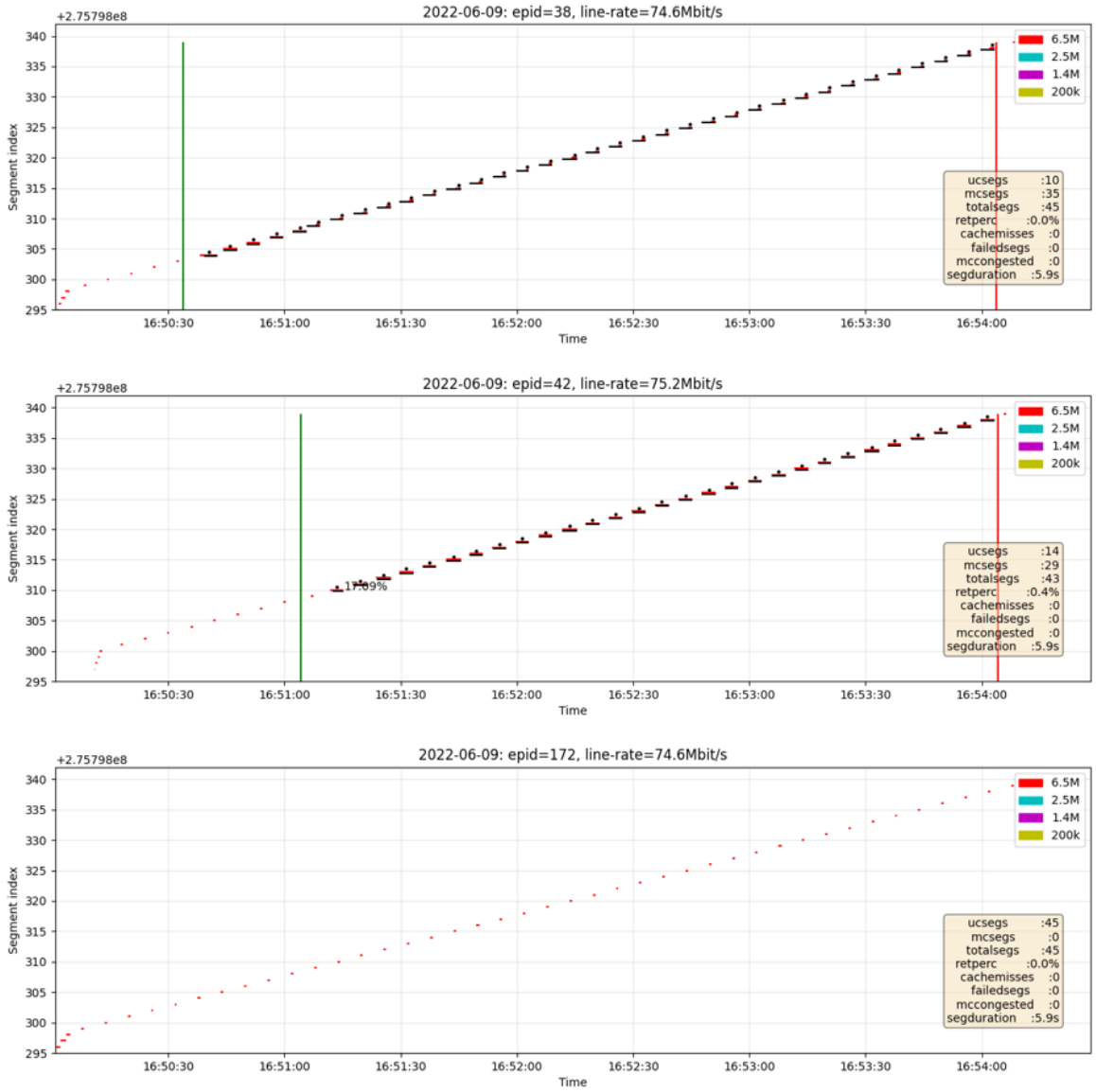

Figure 3 – Example graphical representation of experimental output

The black horizontal lines show multicast delivery from the Root Proxy to the Edge Proxy. The black dot above them indicates that the segment had been received by the Edge Proxy by multicast, stored in its cache and delivered as unicast from the cache to the HLS client. When a number is also shown, this indicates the percentage of the segment that was fetched from the unicast repair service. The vertical index on the charts shows the segment index.

The top chart of Figure 3 shows Raspberry Pi 38 starts at 16:50:00, fetches three segments quickly by unicast, and fetches further segments at the segment interval of 6s. Unicast delivery to the Edge Proxy is fast due to the line speed of 70MBit/s. It joins the multicast channel soon after 16:50:30, immediately after the delivery of segment ‘303 has completed.

As this is the first Edge Proxy to join the multicast channel, when the Root Proxy receives a request for segment ‘304 and for the next segments from this Edge Proxy, it fetches the segment from the CDN and transmits it on the multicast channel, because the segment has not previously been transmitted by multicast and the multicast channel is available. The latter is true because the Edge Proxy makes requests at 6s intervals and the 12.0MBit/s multicast channel can deliver segments encoded at 6.5MBit/s in about 3.25s. The red dots become red lines during this time, with the same length as the black multicast delivery lines, as the time taken to respond to a segment request from the HLS client is mostly determined by the time for delivery over the multicast channel.

The middle chart of Figure 3 shows Raspberry Pi 42 starts after 16:50:00. The reason for the delay is not known, but not unexpected for a real experiment running on hardware. This also fetches three segments quickly by unicast, and further segments every 6s and joins the multicast shortly after 16:51:00, after receiving all of segment ‘308 by unicast.

This HLS client, presumably due to it starting at a slightly different time to the other HLS client, requests each segment about 2.4s before the other HLS client. Hence when Edge Proxy 42 joins the multicast, the first content received by multicast is too late – it is the end of segment ‘308, which this Edge Proxy has already got by unicast after a request from its HLS client. By coincidence, the Root Proxy receives the request from Edge Proxy 42 for segment ‘309 only 80ms after multicast delivery of segment ‘308 has completed. But as the multicast channel is not in use (just), and this segment has not been delivered by multicast, the Root Proxy fetches segment ‘309 from the CDN and delivers it on the multicast channel. This back-to-back use of the multicast channel can be seen on the top chart of Figure 3.

Due to a delay in the process of joining the multicast channel, Edge Proxy 42 does not receive segment ‘309 by multicast and hence must fetch it by unicast, and only starts to receive multicast data part way through segment ‘310, and must fetch the missing part, 17% at the start of the segment, from the unicast repair service.

From this time until the Edge Proxies leave the multicast channel at 16:54:00, Edge Proxy 42 sends the first request for each segment, causing the Root Proxy to deliver it by multicast. The HLS client associated with Edge Proxy 42 observes a segment delivery time equal to the delivery time on the multicast channel, about 3.25s. Whereas the HLS client associated with Edge Proxy 38, which makes requests about 2.4s later, by which time nearly 75% of the segment has been received and cached, observes segment delivery times of about 1s.

Both HLS clients make one more request after their Edge Proxy has left the multicast channel as instructed, which each Edge Proxy satisfies by making a unicast request.

The bottom chart of Figure 3 shows Raspberry Pi 172 fetches three segments quickly by unicast and further segments every 6s until instructed to stop: it was never instructed to join the multicast channel. This Raspberry Pi is used as a control in the experiment, to help with

the identification of any unexpected behaviour, especially when the issue is not related to the MAUD system, but to some other component, such as the content source or the CDN.

CONCLUDING REMARKS

We have described the innovative approach to m-ABR, known as Multicast-Assisted Unicast Delivery (MAUD), that we have developed to enable the use of multicast delivery to reduce the peaks in network traffic caused by delivering identical individual streams for each viewer of the same live content.

MAUD integrates with the CDN rather than the CSP, keeping the CDN operator in the value chain and avoiding integration with a large and potentially rapidly changing set of CSPs.

We have described the trial we have performed with the CDN operator Qwilt and our implementation of the MAUD system. The trial was an overall success, showing that the implementation of the MAUD system behaved as expected.

The content management system needed just a small change to route requests for content from trialists towards the MAUD system. It is expected that in a real deployment the CSP would want some control over when MAUD would be used, and for which content services and for which clients it would be used.

Integration with the CDN Operator was also straightforward. The implementation of mapping a request for a presentation manifest URL containing the specific query parameter to a URL with a MAUD host was reported to us as taking a very small amount of time. The request routing to insert the Edge Proxy into the content path operated as expected for the bots running on Raspberry Pis, and for almost all the devices we tested.

In addition to running an HLS client on the Raspberry Pi running the Edge Proxy, we also tested with the unmodified BT Sport application on a range of consumer devices, with the viewer requesting and viewing content. The Raspberry Pi running the Edge Proxy was configured as a Wi-Fi access point to which the user had to connect the device, rather than their usual home Wi-Fi access point. Users were able to access and view live BT Sport content via an Edge Proxy receiving multicast on the following categories of devices: iOS, Android, X-Box, Fire TV, Android TV, Apple TV and Samsung TV.

We did not observe any undesirable behaviour by inserting multicast into the content delivery path, either on the bots or the clients. Although ABR clients do change the ABR level they request from time to time, we did not observe any situation in which this seemed to have been caused by using multicast.

To address the question of whether the Edge Proxy could be run on BT's commercial home gateway product, we ran an Edge Proxy binary on a next generation BT home gateway development system and confirmed that the memory and processing requirements for the Edge Proxy would be acceptable.

The architecture used this trial was designed to be as close as possible to a scaled deployment, with MAUD head-end components hosted in BT premises in London. The multicast traffic, which was injected into BT's national multicast network, was shown to satisfy BT's alarm and monitoring requirements. BT Security approved the design for use during the trial.

Considering the success of the trial and the ease of integration with the CDN Operator, we plan to develop the MAUD system further, run larger trials, and ultimately deploy MAUD as a network-optimisation, cost reduction, technology.

REFERENCES

1. DVB, January 2022, “Adaptive media streaming over IP multicast”, DVB Document A176 Rev.2 (Third edition), https://dvb.org/wp-content/uploads/2022/01/A176r2_Adaptive-Media-Streaming-over-IP-Multicast_Jan-2022.pdf

2. Apple press release, March 2022, “Apple and Major League Baseball to offer ‘Friday Night Baseball’”, https://www.apple.com/uk/newsroom/2022/03/apple-and-major-league-baseball-to-offer-friday-night-baseball/

3. Wall Street Journal, November 2022, “Netflix Explores Investing in Sports Leagues, Bidding on Streaming Rights”, https://www.wsj.com/articles/netflix-explores-investing-in-sports-leagues-bidding-on-streaming-rights-11667930861

4. Netflix press release, November 2022, “Chris Rock Will Make History as the First Artist to Perform Live on Netflix”, https://about.netflix.com/en/news/chris-rock-will-make-history-as-the-first-artist-to-perform-live-on-netflix

5. Viaplay news release, November 2022, “Viaplay is now live in the UK”, https://www.viaplaygroup.com/news/news-releases/viaplay-now-live-uk-2064665

6. Viaplay, accessed 3 April 2023, “Supported Devices”, Viaplay, https://viaplay.com/gb-en/devices