Public AMT Relay Between Internet2 and Commodity Internet

William Zhang
Computer Systems Lab
Thomas Jefferson High School for Science and Technology
Alexandria, Virginia
Email: willzhang05@gmail.com

***Abstract*—This project focuses on the implementation of the Automatic Multicast Tunneling (AMT) protocol. The primary goal is to promote the widespread adoption of IP Multicast on the Internet through improving user-friendliness and access to multicast content. This will be accomplished by establishing the first publicly available AMT relay on the multicast-enabled backbone (MBONE) of the Internet, researching the current state of multicast content on Internet2 and the MBONE, as well as researching, documenting, and improving existing implementations of AMT gateway client software.**

***Keywords*—IP, multicast, networks, Internet2, MBONE, video, stream, livestream, AMT, relay, gateway, project, implementation**

I. Introduction

With the increasing popularity of high-resolution video streaming, virtual reality, and a growing audience for such content, it is becoming increasingly expensive to distribute this content over the limited bandwidth available to end-users, largely as a result of the way content is delivered over the Internet using the unicast delivery mechanism. The multicast delivery mechanism, by contrast, offers significant advantages in requiring far less bandwidth, but has not been widely adopted on the Internet.

The goal of this project is to transition the wider commodity Internet into multicast support. This paper describes the various methods through which this transition can be facilitated. AMT can be popularized with user-friendly AMT gateway implementations, widely available public AMT relays on the MBONE such as the one now deployed at TJ, and a larger quantity of desirable multicast content. With a large enough user base for AMT, network administrators will be motivated to enable multicast within their administrative domains and eventually join the multicast-enabled backbone of the Internet (MBONE).

II. Background

A. Advantages of Multicast

While unicast is the most widely deployed delivery mechanism on the Internet, unicast has significant disadvantages when scaling to a larger audience. Unicast involves a one-to-one association between the number of streams from the source server to the number of receivers. When delivering to a large audience, this one-to-one association results in much higher costs, due to the need for much higher bandwidth availability.

Large companies aiming to scale content delivery to large audiences currently utilize content delivery networks (CDNs), a network of globally-distributed content sources delivering to audiences based on their geographic location. A CDN does reduce the overall delivery cost to each user, but its widely distributed nature requires a high setup and maintenance cost. CDNs may be a feasible solution for large companies who are able to afford the high costs, however, it will be increasingly difficult for smaller companies or organizations without the same resources to compete with their own content in a bandwidth-constrained environment.

A viable alternative to both methods for video streaming is the multicast delivery mechanism, the significant advantages being its much lower bandwidth requirements. Theoretically, multicast has no additional cost per receiver after the first receiver. In addition, multicast removes the need for the source server to know the number of receivers, the identity of the receivers, or to maintain the same number of connections as receivers. Rather, the multicast source sends packets to a specified multicast group address.

Instead of each receiver having an individual address, interested receivers join the corresponding multicast group in order to receive those packets, using either Internet Group Management Protocol (IGMP) for IPv4 or Multicast Listener Discovery Protocol (MLD) for IPv6 [1]. When a receiver joins a multicast group, a multicast distribution tree is either constructed or extended to the receiver, most commonly using Protocol Independent Multicast Sparse Mode (PIM-SM). The advantage of this delivery mechanism is that, since the source only needs to send packets to a single address, the source requires far less bandwidth compared to that required to maintain a unicast stream for each receiver.

B. Problems with Multicast Adoption

The barrier to widespread adoption of multicast is analogous to that of the widespread adoption of Internet Protocol Version 6 (IPv6). In the same way as IPv6, multicast is an "all-or-nothing" feature that must be enabled on all layer 3 (network) "hops" between source and receiver in order to be effectively used. Each layer 3 router between the origin of the multicast stream and the client's network must maintain a multicast distribution tree and support a multicast routing protocol (typically PIM-SM) in order for the multicast traffic to reach the receiver.

Multicast is not a new technology; in fact, it has existed

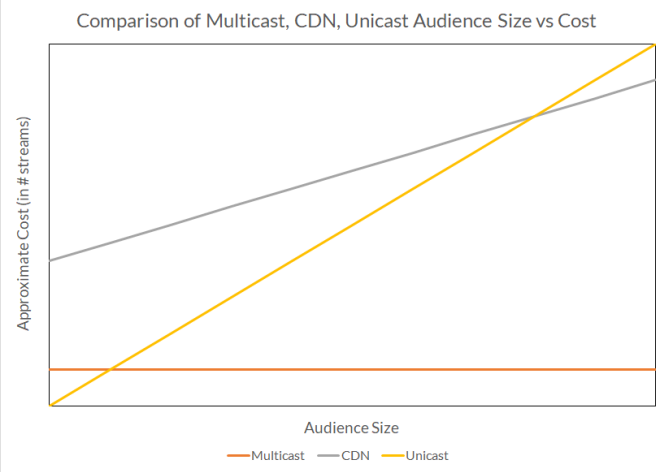

Figure 1. Conceptual graph comparing the cost of different methods of delivering content as the audience size scales. The cost of the unicast alternatives are linear, the cost of multicast is only the initial setup cost.

since 1988 [2]. Due to the "all-or-nothing" problem, however, it has not been widely deployed on the Internet and only certain networks are multicast-enabled. The multicast-enabled backbone of the Internet is commonly referred to as the "MBONE." The largest portion of the MBONE consists of Internet2, a high-bandwidth (100 Gb/s) network that interconnects 305 U.S. universities and other educational and governmental institutions [3].

On Internet2, some institutions, such as the National Science Foundation, provide live streams of various content, including presentations and conferences. Any interested receiver with multicast connectivity to Internet2 is able to receive those streams natively by joining the corresponding multicast groups. Receivers who do not have direct multicast connectivity to Internet2 are unable to receive multicast streams from those institutions, due to the need for all "hops" from the source to the receiver be multicast-enabled. In essence, receivers who do not have direct multicast connectivity to the source network (Internet2) cannot access the multicast content natively.

C. Solution

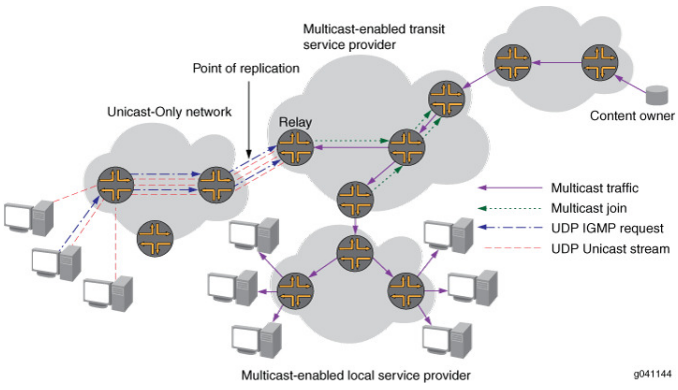

Figure 2. A diagram of how AMT can be used to allow hosts on a unicast-only network to receive multicast content. Source: Juniper Networks

In order to help circumvent this "all-or-nothing" problem and facilitate a transition to native multicast, Automatic Multicast Tunneling (AMT) was created (Fig. 2). AMT, defined in RFC 7450, allows receivers who do not have native multicast connectivity to the source to receive traffic from the multicast source [4]. While AMT does have a cost per receiver, in practice, it is lower than that of a CDN utilizing unicast, largely due to AMT's use of existing devices. Compared to a CDN, AMT is a step towards multicast adoption, rather than continuing to perpetuate the unicast delivery mechanism.

There are two main components to the AMT protocol, the AMT gateway and the AMT relay. Similar to a client-server model, the gateway requests multicast traffic from the relay sitting on the edge of a multicast-enabled network. The relay then encapsulates the desired multicast traffic as a unicast stream to the gateway. AMT functions similarly to Generic Routing Encapsulation (GRE) in that both support encapsulation of network protocols between a point-to-point link. However, AMT is specific to encapsulating multicast traffic within a unicast tunnel and requires far less manual configuration and management of the tunnel on both ends; AMT instead performs much of the configuration automatically, as the name implies [5]. This flexibility also allows for the possibility of direct use by end-hosts.

III. Project

A. Model for Widespread Multicast Adoption

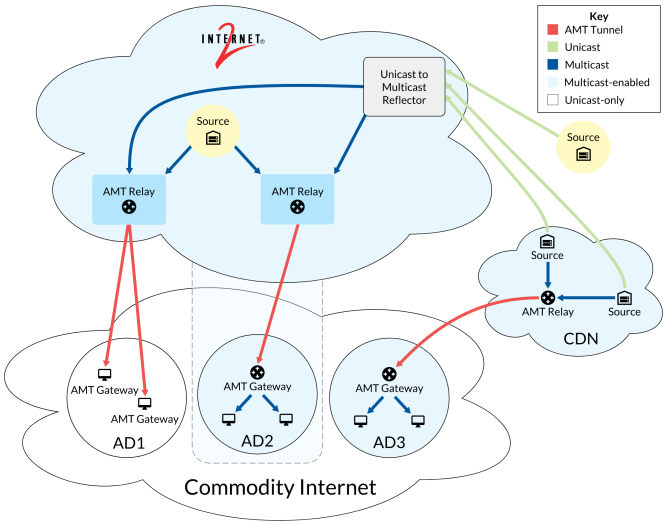

Figure 3. Diagram illustrating a model for multicast adoption.

The primary objective of this project is to drive the adoption of multicast-enabled networks through a multifaceted process, illustrated in Fig. 3. Primarily, AMT relays must first be established to relay traffic from the MBONE to the wider outside "commodity" Internet. AMT can then be deployed in two different models, depending on whether or not the administrative domain (AD) in question, such as a typical Internet service provider (ISP) network, is multicast-enabled. These two models, specified in RFC 8313 sections 3.3 and 3.4, describe the use of AMT tunnels to bridge two multicast-enabled domains and the use of AMT tunnels to bridge end-user devices (end-hosts) to a multicast-enabled domain, respectively [6].

In Fig. 3, The connection between the AMT relay and end hosts in AD1 is the model described in RFC 8313, section 3.4. An AMT relay with connectivity to the MBONE/Internet2 builds AMT tunnels to end-hosts running AMT gateway software in a unicast-only administrative domain, such as an Internet provider's network.

The connection between the AMT relay and the router running an AMT gateway in AD2 is the model described in RFC 8313, section 3.3. Here, rather than building AMT tunnels directly to end-hosts, since the AD is already multicast-enabled, the AMT tunnel is built to an AMT gateway running on the AD border router.

The model described in section 3.3 is also being explored by Akamai, a large CDN that also provides live video streaming services. Akamai's CDNs are multicast-enabled, but to reach end users with its multicast content, Akamai hopes to partner with ISPs to deploy the section 3.3 model and deliver the content over AMT. AD3 in Fig. 3 illustrates this model in use by a CDN like Akamai.

The end goal of the section 3.3 and 3.4 models for AMT is to encourage more ADs to not only become multicast-enabled, but peer directly to the MBONE, thereby slowly building out the multicast-enabled portion of the Internet. This is illustrated by the extended boundary of the MBONE to encompass AD2 in Fig. 3. To justify this, however, there must be demonstrated demand from end-users. The easiest way to encourage this demand is through the creation of more multicast content and easier, user-friendly methods to receive such content.

The creation of more multicast content could be possible through the use of one or more unicast-to-multicast reflectors deployed on the edge of the MBONE. Such a reflector does not currently exist, but in theory should be possible. A unicast-to-multicast reflector would receive unicast streams from the outside commodity Internet and reflect the content to multicast groups within the MBONE, thereby creating multicast content from unicast content. With the combination of this reflector, a public AMT relay, and a user-friendly AMT gateway implementation, multicast adoption could be driven through the availability of content and the resulting demand for this content.

B. Existing Multicast Content on Internet2

Before the step of creating more multicast content or improving user-friendliness, it is important to have an idea of multicast content that already exists. As a result, a survey was done of content currently available on Internet2.

Indiana University provides the Internet2 Looking Glass, a website that allows the execution of basic commands on Internet2 routers [7]. To find multicast content, the 'show multicast route detail' command was run to show information on the available multicast groups on that router. Rather than manually running the command on each router and saving the output, the POST request URL endpoint used by the website was reverse-engineered, and a Python script was used to automatically get the multicast group information.

Not all multicast streams in the output are necessarily used for video, however. In order to filter out any non-video sources, a threshold of 5 packets/second (pps) was used to identify possible active multicast video streams. In addition, without any interested receivers, the state of the multicast group will timeout and the multicast distribution tree will disappear. To help catch as many available multicast streams as possible, the Python script was run as a Linux cron job over the course of two months, saving daily logs of the active multicast video streams. At the end of the two months, another script was written to organize the collection of logs and label each entry with WHOIS information based on the source IP addresses, giving context to each video stream.

In total, the logs yielded 119 unique multicast sources and 40 unique multicast groups. The three sources with the highest packet rate are shown in Fig. 4.

Whois	Source	Group	pps
MASONET	129.174.131.51	233.44.15.9	1167
REDIRIS	130.206.3.133	224.4.0.10	990
EUMETSAT	193.17.9.3	232.223.222.2	3762

Figure 4. Most active multicast sources on Internet2

The first entry in Fig. 4 is NSF TV. This is the National Science Foundation's educational and scientific programming multicast video stream, delivered through George Mason University. Since the source consistently and reliably outputs video and the multicast group port (50001) was known, it was chosen for testing AMT.

The other two sources in Fig. 4, REDIRIS and EUMETSAT, are a Spanish national research network and the European Organisation for the Exploitation of Meteorological Satellites, respectively. Their high packet rates suggest video and audio content, but this could not be verified, as the multicast group ports for both were unknown.

C. Public AMT Relay Deployment

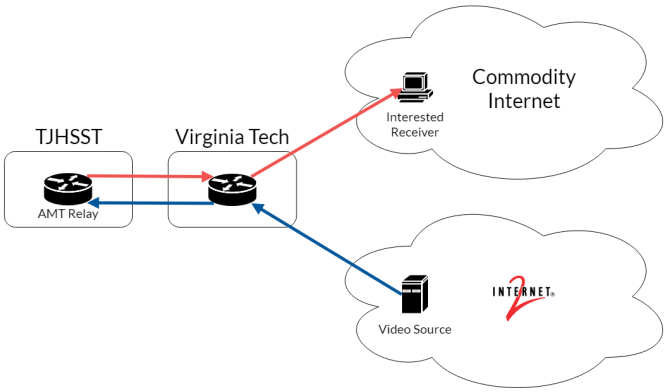

Figure 5. Diagram of the TJ public AMT relay setup.

Virginia Polytechnic Institute and State University (Virginia Tech), a member institution of Internet2, is peered with Thomas Jefferson High School for Science and Technology (TJHSST or TJ), allowing for high-speed (10Gb/s) Internet2 connectivity at TJ [8]. This Internet2 connectivity is limited, however, to the TJ border network.

To initially test native multicast connectivity, a Linux workstation was placed on the border network and VLC media

player was run to test multicast connectivity. VLC, a free and open source media player and framework, has built-in support for receiving native multicast video streams. The primary test video stream used was the George Mason University's live stream at 129.174.131.51 with a multicast group address of 233.44.15.9.

Next, a Juniper MX80 router was deployed on the TJ border network to serve as an AMT relay. In the same way as the Linux workstation, the MX80 is able to receive native multicast streams via TJ's Internet2 connectivity. This AMT relay has been made available to the outside Internet. While there have previously existed publicly available AMT relays, these relays were connected to "walled-garden" multicast networks. The AMT relay deployed at TJ is believed to be the first publicly available AMT relay on the MBONE, relaying multicast traffic from Internet2, the majority of the MBONE, to the commodity internet.

D. Survey of Existing AMT Gateway Implementations

One of the primary limiting factors to the widespread use of AMT is the current lack of user-friendly AMT gateway implementations. AMT relays, conversely, are already well-supported. Cisco and Juniper routers already natively support the AMT protocol on IOS and Junos, respectively, and there already exists documentation for setting up an AMT relay on both router operating systems [5] [9].

Most of the AMT gateway implementations that already exist are difficult to configure and run. Currently, the most user-friendly implementation of an AMT gateway was created at the University of Texas at Dallas (UT Dallas). It is a modified version of VLC 0.9.0 for Windows with added support for AMT [10]. While functional, this gateway implementation is limited to Windows and is based on a decade-old version of VLC. In addition, it does not perfectly implement the AMT protocol; in testing, it was found to be using a non-cryptographically secure 32-bit nonce, a one-use number passed between the AMT gateway and relay to verify a transmission, opening the VLC implementation to possible replay, or man-in-the-middle, attacks.

A team at Upipe, an organization focused on building a flexible dataflow framework library, also created an AMT gateway implementation, but this implementation was unique in that it was an in-browser demo [11]. While a cross-platform AMT gateway implementation using the web browser would be ideal, the Upipe demo utilizes Native Client (NaCl). NaCl allows C or C++ code to be compiled and run natively within the browser, making it simple to have cross-platform support [12]. However, as of May 2017, Google has deprecated the NaCl project in favor of WebAssembly [13].

The possibility for an equivalent implementation in WebAssembly is currently infeasible due to WebAssembly's lack of support for UDP. A possible method to implement AMT in a browser is with the Web Real-Time Communication (WebRTC) project, which provides a in-browser JavaScript API for audio and video streaming.

Another AMT gateway implementation was written in Java by Greg Bumgardner at Cisco Systems as a part of the Java software development kit (SDK) for Multicast Services, known as js4ms [14]. Unfortunately, due to a lack of documentation and the requirement of self-signed certificates to run the AMT gateway, the code could not be tested.

The most promising AMT gateway implementation is based on the original reference implementation. The original reference implementation was done by Tom Pusateri at Juniper Networks [15]. This implementation only worked on the FreeBSD operating system and lacked support for IGMPv3 and forwarding IGMP messages from more than one downstream interface [16]. This code, amtgwd, was then improved at UT Dallas; it was ported to Linux, support was added for IGMPv3 and IGMP message forwarding, and amtgwd was updated to be compliant with the latest draft of the AMT protocol at the time [17]. Further improvements were later made at Concordia University [18].

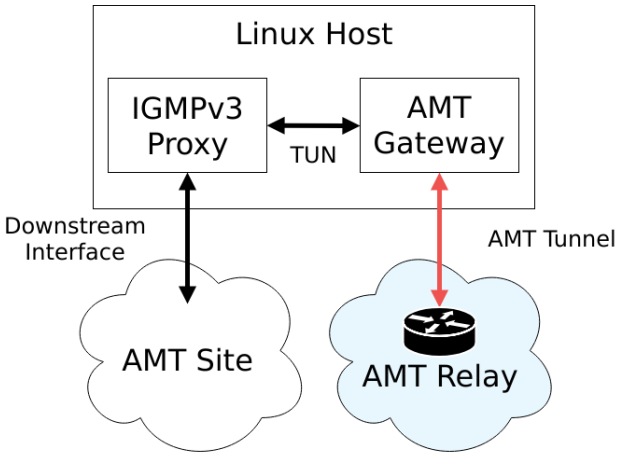

Figure 6. A diagram illustrating the IGMPv3 proxy and AMT gateway interaction.

The improved UT Dallas implementation, as shown in Fig. 6, was designed to be flexible to allow for different deployment scenarios. Rather than a single, standalone implementation, amtgwd requires an IGMPv3 or MLD proxy for either IPv4 or IPv6, respectively. The IGMPv3/MLD proxy has a flexible configuration and allows for not only the Linux host to receive AMT traffic on its own downstream interfaces, but act as a forwarder for AMT traffic to other devices connected to it on a local-area network (LAN).

E. AMT Gateway Test Setup with amtgwd

The source for amtgwd and its bundled IGMPv3 proxy implementation, gproxy, was downloaded from the UT Dallas project website and tested. While amtgwd compiled and ran, unfortunately, gproxy could not be compiled to work on Linux x86-64, due to its out-of-date state. After some searching, Jake Holland of Akamai Technologies' fork of the amtgwd implementation with various improvements and updates was used instead [19]. Some minor modifications were needed in order to compile amtgwd on Linux x86-64. Once compiled, a suitable IGMPv3/MLD proxy was needed to forward the AMT traffic to the downstream host interfaces, so mcproxy was chosen for being actively developed and having been previously tested to work with Holland's amtgwd fork [20].

In order for the AMT gateway to forward to the downstream host interfaces, a tunnel device needs to be created on

the host and assigned an IP address. Tunnel device creation on Linux can typically be accomplished with either the 'tunctl' or 'ip tuntap' commands. In the following example, a tunnel device tun0 is created and assigned the address 10.0.0.2.

```
ip tuntap add tun0 mode tun
ip addr add 10.0.0.2/24 dev tun0
```

Running amtgwd is relatively straightforward; the only parameters required to be passed to amtgwd are the IP address of the AMT relay and the name of the tunnel device using the '-a' and '-c' flags, respectively. In the following example, amtgwd is binding to the tun0 tunnel device and connecting to 198.38.23.145, the public AMT relay deployed at TJ.

```
./amtgwd -a 198.38.23.145 -c tun0
```

On the other end of the tunnel device, the IGMPv3/MLD proxy must also be set up. In the test setup, only a single downstream interface, enp7s0, was used. The following lines were the contents of mcproxy.conf, the configuration file for mcproxy. As the test setup used IPv4, this configuration specifies IGMPv3. The configuration additionally defines a proxy instance 'A' with the tun0 tunnel device forwarding traffic to enp7s0, the downstream interface, as well as a routing table allowing traffic in both directions.

```
protocol IGMPv3;
pinstance A: tun0 ==> enp7s0;
table allways {
        (*|*)
};
```

After configuring mcproxy, it can then be started. During testing, mcproxy would sometimes refuse to bind to tun0. The stock mcproxy was replaced with Jake Holland's fork and ran without any issues.

In order for IGMP messages to reach the AMT gateway, a route must be added for the corresponding source address via the tun0 tunnel device, as shown below.

```
ip route add 129.174.131.51 dev tun0
```

After both amtgwd and mcproxy are running, and a route has been added for the specific multicast source address, a native IGMP join can be sent. This can be accomplished through multiple means; in testing, both the latest, stock version of VLC and the 'mreceive' command available through the set of multicast testing tools (mtools) from the Multimedia Networks Group at the University of Virginia were used [21].

In the test setup, multicast video could successfully be played in the latest version of VLC using the URI shown below.

```
udp://129.174.131.51@233.44.15.9:50001
```

The setup of amtgwd and mcproxy would be difficult for a typical end-user; it could use simplification or automation of the process. To help simplify deployment of an AMT gateway, Jake Holland created docker images that deploy an AMT gateway automatically [22]. However, while docker can help automate most of the setup tasks, the container requires various flags to run and set up properly and is not immediately straightforward to use. For instance, a route to the specified multicast source must still be added via the virtual docker0 interface, bridging the host to the docker containers' network stack.

IV. Future Work

Currently, the amtgwd and mcproxy setup has been tested only on Linux x86-64. In the future, to help drive wider adoption of AMT, an easy-to-use gateway implementation must be ported to or updated for other platforms. An ideal implementation would be cross-platform and potentially utilize in-browser video streaming functionality with AMT.

To encourage adoption of multicast, more multicast content should be made available via the use of a unicast-to-multicast reflector. This reflector would ingest unicast video streams from the outside commodity Internet and output a native multicast stream to the MBONE. Receivers outside of the MBONE could then use an AMT gateway to receive the multicast stream. Once enough demand exists, Internet service providers, for instance, can eventually be convinced to join the MBONE and support multicast.

Once enough multicast content exists, an online multicast content directory website should be created. Such a directory would allow users to browse listings for livestreamed multicast content based on their interests and watch the content over AMT either in-browser or a simple-to-use standalone application. Curators of multicast content would also be able to add to the listings. This central multicast content directory would facilitate access to multicast livestreams and subsequently help to popularize the use of multicast and prove its viability.

V. Conclusion

The ultimate goal of this project is to encourage the adoption of IP multicast on the wider commodity Internet. This paper has described a model for multicast adoption consisting of widely available public AMT relays on the MBONE such as the one now deployed at TJ, user-friendly AMT gateway implementations, and a larger quantity of desirable multicast content.

Even users in the commodity Internet uninterested in multicast content would benefit from the widespread adoption of the models described. If video content can be delivered using less bandwidth through the use of multicast, higher bandwidth availablity will improve for everyone else.

Acknowledgment

The author would like to thank Mr. Peter Morasca, TJ network engineer, for his invaluable support and knowledge of the TJ network. The author would also like to acknowledge the support of his research advisor, Dr. John Zacharias, Mr. Leonard Giuliano of Juniper Networks, and Mr. Jake Holland of Akamai Technologies. In addition, the author would like to acknowledge Juniper Networks' generous donation of the Juniper MX80 router used in this project to the TJ Computer Systems Laboratory and making this project possible.

REFERENCES

[1] *Multicast Overview*, Juniper Networks, 2017. [Online]. Available: https://juniper.net/documentation/en_US/junos/topics/concept/multicast-ip-overview.html

[2] J. Crowcroft, "A brief history of multicast," University of Cambridge, 2010. [Online]. Available: https://slideshare.net/RockyS11/a-brief-history-of-multicast

[3] "About us," Internet2, 2018. [Online]. Available: https://internet2.edu/about-us

[4] G. Bumgardner, "Automatic Multicast Tunneling," Internet Engineering Task Force, Request for Comments 7450, February 2015. [Online]. Available: https://tools.ietf.org/html/rfc7450

[5] *Automatic Multicast Tunneling*, Cisco Systems. [Online]. Available: https://cisco.com/c/en/us/td/docs/ios-xml/ios/ipmulti_pim/configuration/xe-16/imc-pim-xe-16-book/imc-auto-mlt-tun.pdf

[6] R. Sayko, G. Shepherd, and R. Krishnan, "Use of multicast across inter-domain peering points," Internet Engineering Task Force, Request for Comments 8313, January 2018. [Online]. Available: https://tools.ietf.org/html/rfc8313

[7] Indiana University. [Online]. Available: https://routerproxy.grnoc.iu.edu/internet2

[8] "AS3140 The Thomas Jefferson High School for Science and Tech," Hurricane Electric Internet Services. [Online]. Available: https://bgp.he.net/AS3140

[9] *Example: Configuring Automatic IP Multicast Without Explicit Tunnels*, Juniper Networks, 2017. [Online]. Available: https://juniper.net/documentation/en_US/junos/topics/topic-map/mcast-amt.html

[10] K. Sarac, "AMT augmented VLC client implementation," University of Texas at Dallas, 2017. [Online]. Available: https://utdallas.edu/~ksarac/amt#development

[11] C. Massiot, "Automatic Multicast Tunneling & Upipe: a proof of concept," Upipe, 2015. [Online]. Available: https://upipe.org/blog/wp-content/uploads/2015/02/fosdem2015.pdf

[12] "Welcome to Native Client," Google Inc. [Online]. Available: https://developer.chrome.com/native-client

[13] B. Nelson, "Goodbye PNaCl, hello WebAssembly!" Google Inc., 2017. [Online]. Available: https://blog.chromium.org/2017/05/goodbye-pnacl-hello-webassembly.html

[14] G. Bumgardner, "Java SDK for multicast services," 2016. [Online]. Available: https://github.com/gbumgard/js4ms

[15] J. W. Atwood, T. Malla, and A. Ferdous, "Secure and accountable AMT," Concordia University and Internet Engineering Task Force, 2015. [Online]. Available: https://datatracker.ietf.org/meeting/92/materials/slides-92-mboned-1

[16] S. Karisiddappa and K. Sarac, "Automatic Multicast Tunneling open source development," University of Texas at Dallas and Internet Engineering Task Force, 2007. [Online]. Available: https://datatracker.ietf.org/meeting/69/materials/slides-69-mboned-5

[17] K. Sarac, "AMT," University of Texas at Dallas, 2017. [Online]. Available: https://utdallas.edu/~ksarac/amt

[18] R. Brash, "Auto Multicast w/out Explicit Tunnels," Concordia University, 2015. [Online]. Available: https://sourceforge.net/projects/automulticastnoexplicittunnels

[19] J. Holland, "amt," 2018. [Online]. Available: https://github.com/grumpyoldtroll/amt

[20] J. Holland and J. Brzozowski, "Multicast over SPRING @hackathon," Internet Engineering Task Force, 2017. [Online]. Available: https://datatracker.ietf.org/meeting/99/materials/slides-99-mboned-amt-spring-hackathon-update-00

[21] J. Nilsson, "mtools," 2015. [Online]. Available: https://github.com/troglobit/mtools

[22] J. Holland, *amtgw*, 2018. [Online]. Available: https://hub.docker.com/r/grumpyoldtroll/amtgw