Privacy-Preserving Multicast to Explicit Agnostic Destinations

Cuong Ngoc Tran*,Vitalian Danciu*

* Ludwig-Maximilians-Universität München
Oettingenstr. 67, 80538 München, Germany
Email: {cuongtran, danciu}@mnm-team.org

***Abstract*—Multicast protocols require either the participation of hosts in group management or partial address lists of the group members to be sent to end-points (hosts), thus creating a privacy issue. In our new protocol for 1:n multicast over the Internet, senders perform all group management while receivers do not require explicit support for the protocol. The protocol copes with varying degrees of support by routers in the network and avoids the disclosure of others' addresses to end-points. Performance evaluation shows a decrease of the total volume of traffic in the network of up to 1:5 as compared to unicast, suggesting suitability for applications, such as Internet Protocol Television (IP-TV), video conferences, online auctions and others.**

Keywords–Privacy-Preserving Multicast; Agnostic Destination.

I. INTRODUCTION

Applications replacing traditional broadcast services (IP-TV, IP-Radio), phone and video conferencing, and also technical services for software update or large-scale configuration may profit from an *n:m*, multicast, distribution scheme. Today, these applications still rely mostly on unicast transmission despite multicast having been available for a long time.

Typical multicast schemes are based on managed groups (e.g., [1], [2], [3]) where end-points may join the multicast group and the network forwards messages addressed to the group to all group members. Once set up, a multicast group is often symmetric in allowing any participant to address a message to all others. Unfortunately, it requires the network manager to effect configuration reflecting that a given application uses a different kind of network function, while the user is responsible for configuring the application to use multicast. The setup for services being provided across networks and thus across administrative domains always requires the co-operation of each participant domain's network managers.

A. Problem

As illustrated in Figure 1, by requiring an end-point to join and leave the multicast group that supports the desired application, the use of multicast

1) requires network management to authorize a service session and possibly setup (multicast routers),
2) requires the user to execute a network management action,
3) requires transfer of knowledge on group membership on the application level to a multicast group and
4) introduces state to the otherwise state-less (from the view of the end-point) IP communication.

A number of additional properties exacerbate the perceived drawbacks to multicast use:

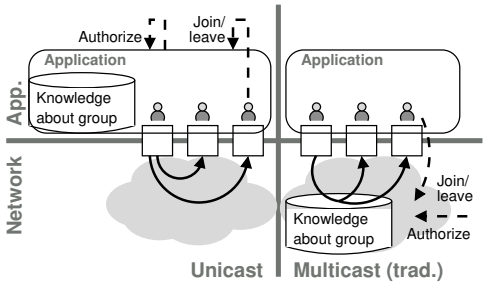


Figure 1. Knowledge and management actions in unicast and multicast.

5) If the network-level setup of multicast fails, there is no automatic fall-back to unicast: instead, the application simply fails as well.
6) All participants in a service must have multicast support.
7) Re-configuration of the application group requires re-configuration of the network.
8) Knowledge about the identities of the participants in a service session is present in the network, possibly in several administrative domains.

Therefore, applications seem to prefer unicast even with the expense of the higher transmission volume, or Application-Layer Multicast (ALM) (e.g., [4], [5]) in spite of it being application specific and requiring a network function within the application's code.

In essence, ALM reduces the *n:m* multicast pattern to the asymmetric case of *1:n* communication, where a single sender addresses a group of receivers. In this case, it is sufficient for the sender to hold knowledge about the group. Since the sender necessarily implements the application layer of the service being provided, group management may be transacted at the application level. Such communication is easily implemented over unicast transmissions. However, it requires the receiver-side configuration and does not profit from network support.

B. Contribution

We propose to combine the benefits of multicast to agnostic receivers with those of optional network support.

We introduce a protocol named Multicast to Explicit Agnostic Destinations (MEADcast) to allow sender-based multicast of IPv6 over the Internet. The novelty of MEADcast is that it protects receivers' anonymity and allows a gradual, pro-active and selective transition between multiple unicast and network-supported multicast. As the protocol favours conservative decisions, we present studies of the transmission cost in the network performed by simulating randomized as well as designed situations.

 ISBN: 978-1-61208-655-2

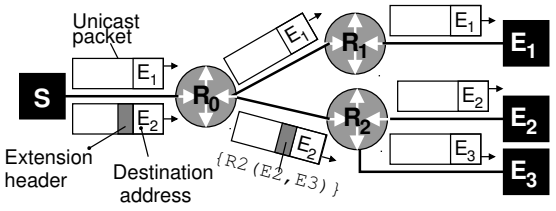


Figure 2. Multicast to agnostic receivers.

C. Technical overview

MEADcast implements a sender-centric multicast in that all knowledge about the receiver group, the network topology and the availability of MEADcast-capable routers (or the so-called MEADcast router in this paper) is gathered at and decided upon by the sender. Given an initial list of receivers, the sender commences to send data in unicast to each sender while simultaneously probing the network for the presence of MEADcast-capable routers and hence for the option to consolidate some of the unicast streams into multicast. Multicast packet headers reflect the MEADcast router responsible for translating the multicast packets into (multiple) unicast packets. Receiving end-points always receive true unicast packets either directly from the sender or generated by a MEADcast router, based on a multicast packet. Only unicast addresses are used in the protocol.

Multicast packets begin with a standard IPv6 header addressed to one of the multicast receivers on a path, followed by a Hop-by-Hop Routing Header with Router Alert. The addresses of all multicast receivers on a path as well as the MEADcast router responsible for translation are encoded into a multicast header. It is typically followed by a UDP header. The addressing pattern is similar to the one in Internet email, where one recipient is addressed directly (To:) while all recipients are included in the carbon copy (CC:) list. The protocol is designed to minimize packet duplication, and recipient list re-writing in transit routers is eliminated.

Figure 2 shows an example where a sender S transmits to three receiving end-points E_i with the aid of three MEADcast-capable routers R_j. Note that the sender transmits unicast directly to E_1, as it is the only end-point on its subtree. It transmits one multicast packet to E_2 and E_3, to be transformed into unicast by router R_2.

None of the end-points can discern the identity of the others, thus preserving privacy, or if the data has been multicasted.

D. Synopsis

Our work is inspired by Xcast [6], which is discussed along with other related work in Section II before expounding the technical properties of MEADcast in Section III. The study of the protocol's behaviour and performance, presented in Section IV, indicates that the reduction in total volume may well be worth the introduction of the mechanism. We provide a discussion of the protocol's overhead, security and application scenarios in Section V. Section VI summarizes our ideas and findings and points out further directions of research.

II. Related work

The idea of multicast was introduced decades ago and has drawn research efforts broadly. A variety of solutions have been proposed and a selection is presented here.

"Standard" multicast [1], [2], [3], [7] specifies the transmission of an IP datagram to a "host group", a set of zero or more hosts identified by a single IP destination address. It requires the network support (multicast capable router) and the receiving end-points to proactively join the "host group". The routers and end-points use the Internet Group Management Protocol (for IPv4) or Multicast Listener Discovery (for IPv6) to maintain the multicast group. The deployment of IP multicast in the Internet is yet far behind expectations due to a number of issues [8]. Amongst those are the management complexity put on end-point and the requirement of overall router upgrade, which constitute the motivation for our works.

ALM implements multicasting functionality at the application layer instead of at the network layer by using the unicasting capability of the network. In contrast to the slow deployment of IP multicast, ALM gains practical success thanks to the ease of deployment. A survey of ALM over the period 1995-2005 was given in [9]. The common approach of ALM is that the multicast participants establish an overlay topology of unicast links to serve as an overlay network on top of which multicast trees can be constructed. The drawback of ALM is the privacy of receiving end-point is not ensured, which means the identity of one end-point might be known by the other; furthermore, the data delivery of ALM depends on the end-point capability, which could not guarantee the stability and reliability. These problems are learnt in designing MEADcast.

Xcast [6] is a multicast scheme with explicit encoding of the list of destinations in the data packets, instead of using a multicast group address. Xcast supports a very large number of small multicast sessions, which makes up complementary scaling property to IP multicast, since the latter has a scalability issue for a very large number of distinct multicast groups. Xcast sends data via optimal route without traffic redundancy when Xcast-aware routers exist; otherwise, receiving end-point has to do ALM and data is sent in a daisy-chain form. The privacy of receiving end-points in the latter case is violated. Xcast limits the number of participants in a multicast session to 64, making it unsuitable for many applications. The idea of Xcast is inherited in MEADcast development while its shortcomings are remedied.

III. Protocol design

MEADcast is implemented by senders and routers. We describe the functions relevant for sender and router elements and message types and procedures necessary for the realization of these functions. A simple multicast scenario described in full illustrates the behaviour of the protocol.

The information needed to describe the protocol is the sender S, the set of end-points E_i that it transmits to, the set of MEADcast routers R_j in the network and their distance d to the sender in hops between MEADcast routers. Association is indicated by superscript, i.e., a router responsible for a subset E_k of the end-points is R^k and an end-point served by a router R_j is E^j.

A. Functions

In MEADcast, we need to distinguish two groups of functions for the sender and the router.

 ISBN: 978-1-61208-655-2

Sender functions include transmission of *unicast* and *multicast* messages, *initiation of discovery* of the path to an end-point and *discovery response evaluation*.

Router functions include normal *forwarding*, *decomposition* of multicast packets to unicast packets and multicast packets and *reaction to discovery* requests from a sender.

1) Discovery-related functions: Both the sender and the MEADcast router are involved in the discovery process. The goal of discovery is for the sender to determine the sequence of routers $(R_1^i, R_2^i, \cdots)$ on the path to each end-point E_i. Discovery requests and responses can be written as `req(E, d)` and `resp(E, d, R)`, respectively.

To initiate discovery, the sender addresses a MEADcast discovery request `req(E, 0)` to an end-point E. When receiving the request, every router R on the path to E increments d and forwards the discovery request `req(E, d+1)` to the next hop; at the same time, R sends a discovery response `resp(E, d+1, R)` to S. Thus, the first router R_1 on the path to E will send $(E, 1, R_1)$, the second $(E, 2, R_2)$ and so on.

S can compile the sequence $\{(E_i, d_1, R_1^i, d_2, R_2^i, \cdots), \cdots\}$ and can compute the groups of end-points to be handled by a given router with a specific distance $(R_j, d_j, E_1^j, E_2^j, \cdots)$.

2) Decomposition: Decomposition, which is specific to MEADcast router, means the transformation of a multicast packet addressed to a set of target end-points into multiple unicast and multicast packets with the same payload.

The target addresses $R_j, E_1^j, E_2^j, \cdots, R_k, E_1^k, E_2^k, \cdots$ within a multicast message are structured to denote that a router R_i is responsible for end-points E^i. During decomposition a router will send unicast packets to each of the end-points it is responsible for and send multicast packets to the routers responsible for the remaining target end-points.

When a multicast packet is created, the targets already served either by unicast or by other multicast messages are removed from the list of targets of the packet being created. The removal process can be implemented efficiently by marking removal in a bitmap and thus eliminating the need to compose a new list of targets.

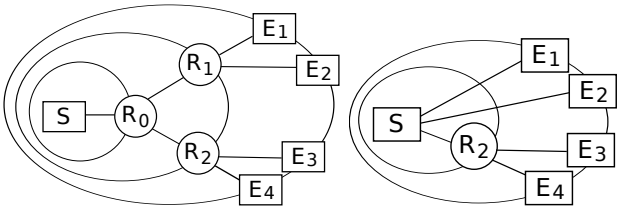
B. Sender behaviour

The sender behaviour involves two phases: *MEADcast discovery* and *MEADcast data sending*.
The sender sends *MEADcast discovery request* `req(`E_i`,0)` to all receiving end-points and updates the network topology in the form of $(R_j, d_j, E_1^j, E_2^j, \cdots)$ whenever it receives a *MEADcast discovery response*. In the mean time, the sender also transmits data to these end-points "unicastly".
MEADcast data sending phase starts when the discovery phase is complete (e.g., after a pre-defined timeout). Based on its network topology view, the sender constructs and transmits *MEADcast data messages* containing the target addresses $(R_j, E_1^j, E_2^j, \cdots, R_k, E_1^k, E_2^k, \cdots)$ for those end-points that can be served by MEADcast routers and stops unicast data to them. In the current MEADcast design, the sender does not put the MEADcast router and its end-point in the address list if it is responsible for only one end-point since it may be a waste of the header space. That end-point is served by unicast. This is the case for E_1 in Figure 2.

It is obvious that if there is no MEADcast router responsible for any receiving end-points, the data sending phase of MEADcast operates exactly as unicast.

The discovery phase is carried out periodically so that the sender can maintain an updated view of the topology.

C. Protocol mechanics by example



(a) Scenario 1 (b) Scenario 2

Figure 3. Network topology from sender viewpoint.

Figure 3 describes the network where the proposed scheme is effective, consisting of five endpoints S, E_1, E_2, E_3, E_4 and three routers R_0, R_1, R_2. Their connections are shown in Figure 3(a) (without the rings). Two different scenarios are described, the first one with all routers being MEADcast capable, the second one with only R_2 being a MEADcast router. The communications between the sender S and the recipients E_1, E_2, E_3 and E_4 via the network in the first scenario are as follows:

1) S transmits data to E_1, E_2, E_3, E_4 "unicastly".
2) S sends four different *MEADcast discovery requests* `req(`E_i`,0), i ∈ {1, 2, 3, 4}`.
3) for unicast message, R_0, R_1, R_2 simply forward it to the intended receiver.
4) R_0 receives `req(`E_1`,0)`, it reacts to the presence of the Hop-by-Hop header and analyses the content of the MEADcast header. R_0 sends a *MEADcast discovery response* `resp(`E_1`, 1, `R_0`)` to S. It also sends `req(`E_1`,1)` to E_1. The same procedure is carried out for `req(`E_2`,0), req(`E_3`,0), req(`E_4`,0)`.
5) R_1 receives `req(`E_1`,1)`, it sends `resp(`E_1`, 2, `R_1`)` to S. It also sends `req(`E_1`,2)` to E_1. The same procedure is carried out when R_1 receives `req(`E_2`,1)`.
6) R_2 receives `req(`E_3`,1)` and `req(`E_4`,1)`, it sends `resp(`E_3`, 2, `R_2`)` and `resp(`E_4`, 2, `R_2`)` to S. It also sends `req(`E_3`,2)` to E_3 and `req(`E_4`,2)` to E_4.
7) E_1, E_2, E_3, E_4 receive the unicast messages normally. For the *MEADcast discovery request*, they do not understand and simply drop it.
8) S receives *MEADcast discovery responses* and updates its network topology view as $(R_0, 1, E_1, E_2, E_3, E_4), (R_1, 2, E_1, E_2), (R_2, 2, E_3, E_4)$. The topology view of sender can be illustrated by the rings in Figure 3(a), where the sender is at the center, R_0 has distance one and lies on the first ring, R_1 and R_2 are on second ring with a distance of two and all receiving end-points are always at the outermost ring.
9) S stops transmitting data via unicast and starts MEADcast data sending phase. S transmits a *MEADcast data message* with E_1 as the destination IP address and $\{R_1, E_1, E_2, R_2, E_3, E_4\}$ in the MEADcast header address list. A position field showing the position of MEADcast router in the address list and another status field

 ISBN: 978-1-61208-655-2

marking whether a MEADcast router has received the *MEADcast data message* are also included in the message.

10) R_0 receives the *MEADcast data message*, sees that:
    - it does not have to deliver message to any receivers since its address is not in the MEADcast address list.
    - based on the position field and status field, there are two other MEADcast routers that need to receive *MEADcast data message*. R_0 duplicates the original *MEADcast data message*, the status field of the first one is modified, indicating that R_2 has received a *MEADcast data message*. R_0 sends this message to R_1. R_0 changes the destination IP address of the second message to E_3, modifies the status field to indicate that R_1 has received a *MEADcast data message* and sends it to R_2.
11) R_1 receives a *MEADcast data message*, sees that it is responsible for E_1 and E_2. R_1 constructs two unicast messages with the data from the *MEADcast data message* and transmits each to E_1 and E_2. There is no MEADcast router that needs to receives this *MEADcast data message*.
12) R_2 receives a *MEADcast data message*, sees that it is responsible for E_3 and E_4. R_2 constructs two unicast messages with the data from the *MEADcast data message* and transmits each to E_3 and E_4. There is no MEADcast router that needs to receives this *MEADcast data message*.

The communications between the sender S and the recipients E_1, E_2, E_3 and E_4 via the network in the second scenario (only R_2 is a MEADcast router) have the same first three steps as in the first scenario. The further steps are as follows:

1) R_0 receives `req(`E_1`,0)`, it reacts to the presence of the Hop-by-Hop header and analyses the content of the MEADcast header, which it does not understand. It forwards the message further to the E_1 direction. R_0 does not drop the message since the option type identifier of MEADcast header is 00 [10]. The same procedure is performed for `req(`E_2`,0)`, `req(`E_3`,0)`, `req(`E_4`,0)`.
2) Similarly, R_1 receives `req(`E_1`,0)` and `req(`E_2`,0)`, it sends these messages to E_1 and E_2.
3) R_2 receives `req(`E_3`,0)`, `req(`E_4`,0)`. It sends `resp(`E_3`, 1, `R_2`)` and `resp(`E_4`, 1, `R_2`)` to S. It also sends `req(`E_3`,1)` to E_3 and `req(`E_4`,1)` to E_4.
4) E_1, E_2, E_3, E_4 receive the unicast messages normally. For the *MEADcast discovery request*, they do not understand and simply drop it.
5) S receives *MEADcast discovery responses*, updates its network topology view as $(R_2, 1, E_3, E_4)$. Its network topology view is illustrated in Figure 3(b). There is no MEADcast router on the paths to E_1, E_2, only R_2 is on the paths to E_3, E_4 and it lies on the first ring of distance one. All receiving end-points are on the outermost ring.
6) S starts MEADcast data sending phase. Based on its topology view, S sees that:
    - there is no MEADcast router on the paths to E_1, E_2. S transmits data to these receivers "unicastly".
    - R_2 is on the paths to E_3, E_4. S transmits a *MEADcast data message* with E_3 as the destination IP address and $\{R_2, E_3, E_4\}$ in the MEADcast header IP address list. A position field and a status field as described in the first scenario are also included in the message.

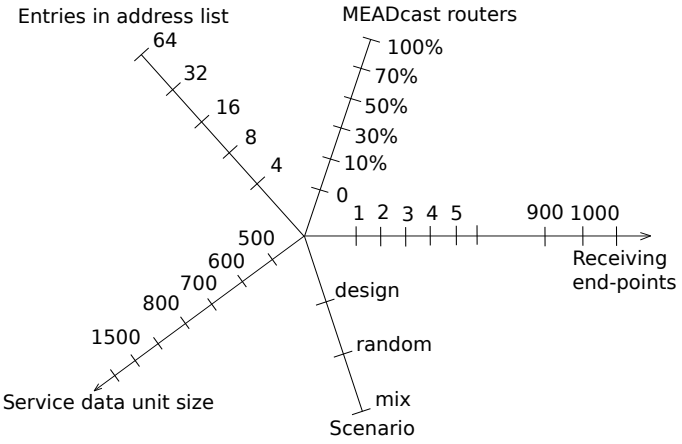


Figure 4. Parameters for experiments.

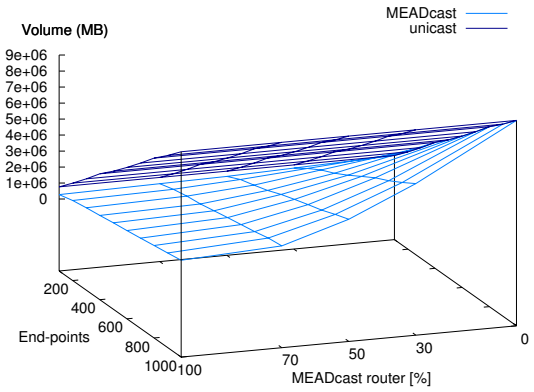


Figure 5. Total data volume of unicast and MEADcast.

7) for unicast message, R_0, R_1 simply forward it to the intended receiver.
8) R_0 receives the *MEADcast data message*, which it does not understand. It forwards the message further to the E_3 direction. R_0 does not drop the message since the option type identifier of MEADcast header is 00 [10].
9) R_2 receives a *MEADcast data message*, sees that it is responsible for E_3 and E_4. R_2 constructs two unicast messages with the data from the *MEADcast data message* and transmits each to E_3 and E_4.

IV. EVALUATION

We have performed experiments within the parameter space illustrated in Figure 4. We simulate MEADcast for 100 routers both on random network topologies with a diameter of 16 (generated by GT-ITM [11]) and on "designed", realistic topologies, using ns-2 [12]. Table I shows the total volume transmitted on all links in the network when the sender transmits a stream of 800 MB of data into the network with 500 and 1000 receiving end-points.

The number of receiving end-points and the number of MEADcast routers in the different scenarios are varied in the experiments. The impact of the service data unit size and the number of entries in the address list are discussed in Section V.

The volume gap of two approaches is plotted in Figure 5. The number of receiving end-points ranges in 100, 200...

TABLE I. TOTAL TRAFFIC VOLUME IN THE WHOLE NETWORK [MB]. * INDICATES DESIGNED TOPOLOGY.

Topology		*Unicast*	*MEADcast without discovery*	*Discovery (one time)*	*Traffic reduction [%]*
End-points	*MEADcast routers [%]*				
500	0	4,136,544	4,136,544	0.409	$(-0.x)$
	10(*)	4,136,544	1,279,936	0.813	69.1
	30	4,136,544	2,513,296	0.897	39.2
	50	4,136,544	1,698,336	1.216	58.9
	70	4,136,544	1,069,276	1.389	74.2
	100	4,136,544	902,056	2.835	78.2
1000	0	8,341,776	8,341,776	0.826	$(-0.x)$
	10(*)	8,341,776	2,525,904	1.640	69.7
	30	8,341,776	4,978,384	1.802	40.3
	50	8,341,776	3,137,972	2.462	62.4
	70	8,341,776	1,866,456	2.819	77.6
	100	8,341,776	1,552,304	5.727	81.4

to 1000.

If there is no MEADcast router, sender sends mainly unicast messages and periodically sends discovery messages which occupy only a little traffic volume over the whole network. This discovery overhead is indicated by the value "$-0.x$" in Table I and depends on how many times the discovery is performed. Hence, the traffic volume of MEADcast protocol when there is no MEADcast router is approximately that of unicast, provided that sender has large traffic to send. The gap increases when the number of receiving end-points and the percentage of MEADcast routers grow. The extreme case of 1000 end-points and 100% MEADcast routers shows the difference of 81.4% in total traffic volume.

The total traffic volume reduction is considerable in the presence of sufficient MEADcast routers, as shown by the designed cases. The link stress (the number of identical packets sent by a protocol over each underlying link in the network) [13] at the sender is reduced to an even higher degree.

V. DISCUSSION

The concepts of MEADcast require a higher degree of interaction between the network layer and its upper layers (transport and application), that merits discussion. While our simulation results indicate significant performance gains for a wide range of parameters, MEADcast scenarios may be limited by properties of the protocol or the applications using it, and routers may experience a higher control plane load. After discussing these points, we conclude with remarks on fault and security issues.

A. Relation to upper layers

Decomposition of MEADcast data packets may yield packets with different destination addresses and thus invalidate checksums in upper layer headers that include network addresses in the checksum (e.g., UDP for IPv6). For the new packet to be valid at the destination, MEADcast routers must re-compute these checksums for every new unicast packet and every new MEADcast packet with a different destination address. This issue is due to the re-use of network addresses in transport layer protocols, and problematic not only because of the increased load on routers' control plane but also because of the requirement to handle protocols other than IP.

Transport layer port numbers will differ at end-point sockets and have to be included in the MEADcast header along with the IP address of each end-point, thus creating an additional binding to the transport layer.

Network service primitives do not support addressing multiple recipients. Therefore, applications and higher protocols on the sender side must be modified to make use of MEADcast. A solution idea would be to use "regular" IGMP/IGMP6-based multicast on the first hop, thus allowing applications and higher protocols to employ multicast addressing as usual, then use a proxy function to translate between regular multicast and MEADcast before transmitting. While Path MTU discovery [14] is a standard function of the Internet, the application requirements on payload size are not readily available to allow the computation of optimum header size. We envision an interface to the network layer allowing the application to issue *hints* with respect to its intended use of the network.

We emphasize that these modifications are required for the sender only. The providers of asymmetric applications (IP-TV, Internet radio etc.) can be assumed to correctly gauge the cost and benefit of introducing modifications to consolidate the multitude of unicast flows they create presently.

B. Limitations

Inherent limitations of the approach include the maximum number of entries in the address table, the overhead introduced by the address table, the time required to establish multicast structures and load introduced in the control plane of routers.

MEADcast routers do not keep group information, thus rendering MEADcast processing stateless, while nevertheless complex in contrast to multiple flows, that may be handled by accelerators such as FPGAs.

MEADcast performs a gradual transition from a number of unicast packet flows to a (smaller) number of multicast flows as the availability of MEADcast-capable routers is discovered. Sessions that are shorter than the time for discovery not only forego the benefit of multicast but also carry the additional load for discovery; they are an application for unicast.

Given a path MTU value, the number of entries in the address table determines the remaining space for payload. If the service data units received from the upper layer is small, the sender may enlarge the number of entries, however, for large number of end-points even small payloads will require the sender to issue multiple multicast packets. Figure 6 shows the critical points where data volume is increased when the address table space of 32 entries is exhausted by one router multicasting to an increasing number of end-points. Conversely, a large address table leaves less space for payload and may lead to fragmentation, as illustrated in Figure 7.

C. Fault and security considerations

Packet loss naturally incurs a larger penalty for MEADcast than unicast, as more receivers are affected. In particular, the failure of a MEADcast path by changes in routing (by administrative action or by faults) will lead to continuous loss of packets until the periodic discovery mechanism informs the sender of the change in the network topology. A higher discovery frequency might lessen the consequences at the expense of increased control plane load in routers and an increase in the number of (albeit small) packets transmitted over a path.

 ISBN: 978-1-61208-655-2

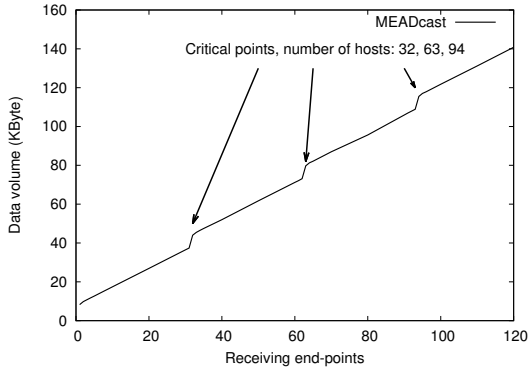


Figure 6. Total volume increased by exhausted address table.

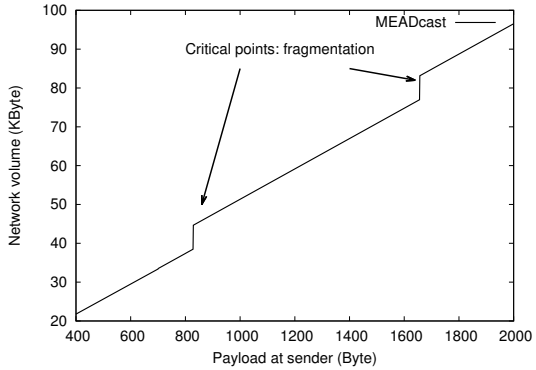


Figure 7. Fragmentation impact on total volume (1 router, 31 end-points).

Beyond the security issues noted for Xcast (see [6]), that also employs sender-based multicast, we note that the deprecation [15] of the Type 0 Routing Header in IPv6 to prevent amplification attacks suggests careful scrutiny of any mechanism, that causes Internet routers to transmit more packets than they receive. We presume our mechanism to be reasonably safe due to the following properties: *i)* the total volume of transmitted multicast data does not exceed the corresponding unicast volume for the same data, with the exception of the signalling required for fallback to unicast, *ii)* addresses are not modified by routers, i.e., data is transmitted via the same path in both unicast and multicast modes.

VI. Conclusion

The MEADcast protocol introduced in this paper creates a separation of concerns between the stateful sender, stateless routers and agnostic end-points. While avoiding the need of network-side group management and the transmission of address tables to end-points, multicast is automatically employed when possible, falling back on unicast, when not. MEADcast yields in our simulations a significant factor of reduction of the traffic volume of an application session compared to the same session in pure unicast, in network topologies with a sufficient number of supporting routers.

Our discussion indicates several open questions and avenues for development, including the study of the load increase in router control planes and the real-world evaluation of streaming applications based on a module implementation for the Linux kernel and the development of an interface for the management of multicast groups and parameters on the sender side. A different point of interest is the realisation of MEADcast with virtual network functions, to be used in and between Software Defined Networks (SDN).

Acknowledgment

The authors wish to thank the members of the Munich Network Management (www.mnm-team.org), directed by Prof. Dr. Dieter Kranzlmüller and the anonymous reviewers for valuable comments on previous versions of this paper.

References

[1] S. Deering, “Host extensions for IP multicasting,” RFC 1112 (INTERNET STANDARD), Internet Engineering Task Force, Aug. 1989, updated by RFC 2236, [retrieved: June, 2018]. [Online]. Available: http://www.ietf.org/rfc/rfc1112.txt

[2] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, “Internet Group Management Protocol, Version 3,” RFC 3376 (Proposed Standard), Internet Engineering Task Force, Oct. 2002, updated by RFC 4604, [retrieved: June, 2018]. [Online]. Available: http://www.ietf.org/rfc/rfc3376.txt

[3] H. Holbrook, B. Cain, and B. Haberman, “Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast,” RFC 4604 (Proposed Standard), Internet Engineering Task Force, Aug. 2006, [retrieved: June, 2018]. [Online]. Available: http://www.ietf.org/rfc/rfc4604.txt

[4] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, Scalable application layer multicast. ACM, 2002, vol. 32, no. 4.

[5] D. A. Tran, K. A. Hua, and T. T. Do, “A peer-to-peer architecture for media streaming,” IEEE journal on Selected Areas in Communications, vol. 22, no. 1, 2004, pp. 121–133.

[6] R. Boivie, N. Feldman, Y. Imai, W. Livens, and D. Ooms, “Explicit Multicast (Xcast) Concepts and Options,” RFC 5058 (Experimental), Internet Engineering Task Force, Nov. 2007, [retrieved: June, 2018]. [Online]. Available: http://www.ietf.org/rfc/rfc5058.txt

[7] W. Fenner, “Internet Group Management Protocol, Version 2,” RFC 2236 (Proposed Standard), Internet Engineering Task Force, Nov. 1997, updated by RFC 3376, [retrieved: June, 2018]. [Online]. Available: http://www.ietf.org/rfc/rfc2236.txt

[8] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, “Deployment issues for the ip multicast service and architecture,” IEEE network, vol. 14, no. 1, 2000, pp. 78–88.

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

[10] R. Hinden, “Internet protocol, version 6 (ipv6) specification,” no. 8200, Jul. 2017, [retrieved: June, 2018]. [Online]. Available: http://www.ietf.org/rfc/rfc8200.txt

[11] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, “How to model an internetwork,” in INFOCOM’96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE, vol. 2. IEEE, 1996, pp. 594–602.

[12] “The network simulator - ns-2,” [retrieved: June, 2018]. [Online]. Available: https://www.isi.edu/nsnam/ns/

[13] Y.-h. Chu, S. G. Rao, S. Seshan, and H. Zhang, “A case for end system multicast,” IEEE Journal on selected areas in communications, vol. 20, no. 8, 2002, pp. 1456–1471.

[14] J. McCann, S. Deering, J. Mogul, and R. Hinden, “Path mtu discovery for ip version 6,” no. 8201, Jul. 2017, [retrieved: June, 2018]. [Online]. Available: http://www.ietf.org/rfc/rfc8201.txt

[15] J. Abley, P. Savola, and G. Neville-Neil, “Deprecation of Type 0 Routing Headers in IPv6,” RFC 5095 (Proposed Standard), Internet Engineering Task Force, Dec. 2007, [retrieved: June, 2018]. [Online]. Available: http://www.ietf.org/rfc/rfc5095.txt

 ISBN: 978-1-61208-655-2