Dynamic Manycasting Hierarchies

Oleg Kolesnikov  Erfan Ali
{ok,erfanali}@cc.gatech.edu

Georgia Institute of Technology
Atlanta, GA

Abstract

This paper addresses the need for intermediate handling of multicast data while it traverses the net from a multicast source to its receivers. The proposed concept of *dynamic manycasting hierarchies* is a fusion of both multicast and anycast methodologies. The concept is designed to be transparent and does not require any modifications to be made to the existing multicast and anycast delivery protocols.

By adding a distributed intermediate entity that keeps track of receivers and dynamically selects the best way to deliver multicast data we not only achieve faster data delivery and higher level of security, but also provide a framework for future protocol extensions and added functionality without making any global changes to the multicastric protocols themselves.

In particular, manycasting hierarchies address the problem of having anycast addresses as members of multicast groups, shifting the load from a multicast source. Since there is a number of servers that replicate the data from a source there are also less failures caused by congestion and routing problems. Another important factor is that the transmission of data to end-clients is not limited to multicast *per se*, and can be done based on an end-client’s request using various protocols, including unicast, broadcast, and reliable multicast.

Finally, by using interim source hopping, where various parts of a multicast message are being received by a client from different sources, a potentially high level of security can be achieved. Since the manycasting hierarchy that performs an extra security function on multicast data knows about its end-clients, it is much easier to provide verification and authentication and it is no longer necessary to utilize complex reliable multicast disciplines that would otherwise be used by an end-client to talk to a multicast source.

1 Design goals

One of our major design goals is to introduce a framework that provides for intermediate handling of multicast data and multifunctional packet preprocessing by utilizing a dynamic hierarchy of nodes. In addition, the designed protocol must effectively fit into the distributed
nature of the Internet and minimize the number of changes needed to be made to the existing multicast and anycast delivery protocols.

The key factor is to allow multicasting to be easily extensible and at the same time maintain a sufficient level of consistency and control. By balancing between a rigid scheme where once implemented a protocol is evolving very slowly and users are not able to effectively port the implementation to their needs, and various modular application-layer schemes [8], where users can easily define their own functionalities and extend the system voluntarily, it is possible to address the needs of a wider spectrum of Internet applications and users.

At the same time, it is important to avoid as many potential problems as possible at the design stage. To achieve this, we analyzed typical problems that were brought up during the evolvement of several fundamental Internet Protocols, such as IP Multicasting, Protocol Independent Multicast (PIM) [5], Reliable Multicast Transport Protocol (RMTP) [6], and Domain Name System (DNS). In our design document, we will emphasize the following issues:

1. **Scalability.** The ability of multicasting to build dynamic hierarchies of hosts all over the net is not only one of its strongest points but also a possible bottleneck. Therefore, it is necessary to effectively scale hierarchies and exchange information without flooding the network.

2. **End-to-end paradigm.** Should we only export a basic set of functions to a higher layer or should we make our solution more comprehensive? What layer of OSI model will multicasting be operating on?

3. **Extensibility.** By making multicasting extensible we address some of the questions that were brought up by end-to-end paradigm. However, to what extent should multicasting be allowed to be extended? How to effectively control the extensions, provide authentication, and delegate permissions? The latter is directly related to the next issue.

4. **Security.** It is essential to make multicasting secure to a certain degree. However, in a distributed environment with a large number of hosts, encryption may cause a significant drop in performance. Therefore, it is necessary to keep a balance between the supported level of security and application performance.

5. **Transparency.** If necessary, end-clients must be able to bypass multicasting and use multicasting or anycasting directly.

6. **Underlying architecture.** The issue here is overdependency on the existing set of lower-layer protocols. A certain degree of generalization must be present within a protocol, allowing it to be less dependent of specific features of underlying protocols [5]. To illustrate, many higher level OSI protocols were designed to benefit from the existing set of APIs provided by lower layers. Unfortunately, some of them became overdependent on specific features provided by lower layers. In some cases this was not only related to certain functions of a higher level protocol but also to the whole concept.
In particular, some early versions of multicast routing protocols that were using reverse path forwarding technique relied on certain OSI layer-2 optional features that were sometimes disabled by system administrators, which caused these protocols to fail. Protocol Independent Multicast [5] was designed to address this problem.

2 The Manycasting Model

The proposed architecture consists of a number of manycasting constellations that may serve as a multifunctional proxy between multicast source(s) and end-clients. Each manycast constellation consists of a number of members that have different ranks and are responsible for performing different functions within the constellation. Members within a manycast constellation form a dynamic hierarchy, that can be changed over time based on various factors, including current network statistics, number of multicast sources, average distance to end-clients, and number of end-clients.

At every point in time, fully functional constellation must have at least three logical components Registrar, Probe, and Delivery Agent, as shown in Figure 1. Members that implement one or more of the logical components mentioned above and form a static part of a constellation are called core members or quasars.

The rest of the members of the constellation that can join and leave the hierarchy dynamically are called neophytes. It is recommended that a constellation be diversified and have members in various independent network entities, such as autonomous systems, ISPs; this would allow to potentially increase the speed of the delivery as well as to distribute the load posed on the net.

The dynamic part of a constellation that consists of neophytes uses statistics to determine ranks of all members. The rank of a particular member can be a function of many parameters, such as members' reachability or number of failures over time, cumulative coefficient characterizing members' connectivity, network coverage area, average load, and other criteria.

At every point in time, fully functional constellation must have at least three logical components Registrar, Probe, and Delivery Agent, as shown in Figure 1. Members that implement one or more of the logical components mentioned above and form a static part of a constellation are called core members or quasars.

The rest of the members of the constellation that can join and leave the hierarchy dynamically are called neophytes. It is recommended that a constellation be diversified and have members in various independent network entities, such as autonomous systems, ISPs; this would allow to potentially increase the speed of the delivery as well as to distribute the load posed on the net.

The dynamic part of a constellation that consists of neophytes uses statistics to determine ranks of all members. The rank of a particular member can be a function of many parameters, such as members' reachability or number of failures over time, cumulative coefficient characterizing members' connectivity, network coverage area, average load, and other criteria.

The higher the rank of a member, the higher the probability that the member will be utilized by the constellation as an active participant. Eventually, highly ranked members may be added to the core of the constellation and be delegated more decisive functions and responsibility as members of the core. In some cases becoming a core of a constellation may also mean being added to a multicast group that is used by constellation members to exchange messages.

In some cases, it may be worth adding an additional Statistical analysis component
that will generate and keep track of statistics to decouple the load caused by gathering and analyzing intra-constellation statistics from other time-critical functions such as probing and data delivery. Accordingly, the other components will access the statistical data provided by this component, and use the obtained information to take actions and change the infrastructure of a constellation, if necessary.

3 Delivery of Data using Manycasting Hierarchies

3.1 Algorithm

The process of data delivery using Manycasting hierarchies is shown in Figure 2. The basic idea is that a manycasting constellation (MC) acts as a proxy between end-clients and multicast sources. MC waits for registration requests from clients, then it subscribes to multicast groups that were requested and starts receiving data the appropriate multicast sources.

While receiving data, MC members interact with each other and perform various measurements of the network connections, determining a set of best paths for delivering the data to end-clients. Then, based on the registration information, multicast data is delivered to the appropriate end-clients using the delivery modes specified during registration.

It is possible for an end-client (EC) to register multiple IP unicast addresses or an anycast address to be used to receive multicast data from MC. In this case, MC will provide best-effort delivery service and determine the best unicast address or anycast member to send multicast data to.

3.2 End-client registration

When an end-client wants to receive data for a particular multicast group, it uses a publicly known anycast address to contact the closest constellation registrar. The registrar is selected automatically using anycasting.

The registration message sent by EC should contain the following information:

- **IP address** of the multicast group that EC wants to receive information from

- **Delivery mode**:
  - *Network protocol*: multicast or unicast.
  - *Transport protocol*: reliable (tcp, valid only for unicast transmission) or unreliable (udp).

- **IP address(es) of interface(s)** that EC wants to use to receive data. If there
are multiple IP addresses they are considered interchangeable and logically equivalent. The selection of a particular IP address for delivery will be based on current network load. This can be especially useful for multi-homed hosts that have multiple links to the Internet.

- **Port number** used to receive data.
- **Optional data**: Secure transmission request, anycasting mode registration et al.

The process of registration consists of at least two messages – registration request and registration reply. Both messages are sent using udp transport protocol, port \textbf{1533}. If EC does not receive a reply from CR during the timeout interval specified in the registration message, EC retransmits two more times before giving up and trying to connect another CR.

After a successful registration EC must be immediately available to receive data using the appropriate data delivery mode specified in the registration request. For example, if a reliable transmission on port 50000 was requested, EC must create a socket, bind it to the tcp port 50000 and be ready to receive data.

To have a better idea of what a anycast registration request/reply messages look like, an excerpt from a anycast request messages is included as follows:

**Static part**

Bits 0-3: **Version.** This document defines version 1 of the Manycasting protocol.

Bits 4-7: **Type of Request.** There are currently nine request types defined by the protocol:

- 0 = MA_REG, Registration request.
- 1 = MA_RSEC, Secure Registration request.
- 2 = MA_ACK, Registration acknowledgement.
- 3 = MA_SACK, Secure Registration acknowledgement.
- 4 = MA_NACK, Registration denial.
- 5 = MA_BOUNCE, Forwarding of client's request.
- 6 = MA_EXT, Extension of service request.
- 7 = MA_DEREG, Deregistration request.
- 8 = MA_USEGEXT, Use a generic extension.
- 9 = MA_ADDEXT, Add a localized extension.
- 10 = MA_USELEXT, Use a localized extension.

Bits 8-9: **Network protocol.**

- 0 = unicast
- 1 = multicast

Bits 10-11: **Transport protocol**

- 0 = tcp
- 1 = udp

Bits 12-31: **Timeout (msec).** Defines maximum time interval EC will wait before resending its registration request.

Bits 31-62: **MD5 Checksum.** MD5 checksum of the registration message with Checksum field set to zero.

**Dynamic part**

The rest of the registration message consists of variable number of quadruples that can be described as follows:

\( (\text{Object type}, \text{Object instance length}, \text{Number of instances}, \text{Object instance value(s)}) \).

**Object Type** (1 byte):

- 0 = Requested Multicast group address(es).
- 1 = Destination IPv4 address(es).
- 2 = Destination IPv6 address(es).
- 3 = Destination Anycast address(es).
- 4 = Destination port number.
- 5 = Reference to a common extension.
- 6 = Extension code.
7 = Error code.
8-15 = Reserved for Security info.
16-20 = Optional, implementation dependent info.

**Instance length** (2 bytes):
Length of each object in bytes. Must be consistent with a list of object lengths corresponding to all generic object types that is kept by each member of registrars. For example, for IPv4 address, instance length equals to 4.

**Number of instances** (1 byte).
Number of objects of type “Object type” that follow.

**Object instance(s) data** ("Instance length" multiplied by "Number of Instances" bytes).

Binary object instance data that complies with “Object type”, “Object instance length”, and “Number of instances”.

CR uses soft state mechanism to keep registration information. Therefore, if EC wants to deregister it can either send a registration message that is exactly the same as the initial registration message but has a MA_DEREG flag set or simply do nothing and let the registration information expire.

During the registration, MR can respond with a MA_BOUNCE message. This means that the manycasting constellation recommends the end-client that sent a registration request to either use another constellation or to subscribe to a multicast source directly. This response can be caused by various reasons, such as MC experiencing internal problems, too many clients and other reasons. The explanation for both MA_BOUNCE and MA_NACK messages sent by a registrar is encoded as an object of type “Error code” containing the respective error code.

### 3.3 Multicast subscription optimization

When a multicasting constellation (MC) accepts a non-secure request from an end-client (EC), MC has to subscribe to the multicast group that was requested. However, in most of the cases it is extremely inefficient to subscribe all members of a constellation to the multicast groups requested by ECs. Therefore, after EC has made a request, a constellation registrar (CR) determines two members of the constellation that are the closest\(^1\) to the appropriate EC and makes them responsible for delivering data to that EC.

A sample network configuration explaining multicast subscription is presented in Figure 3. This figure reveals that there are two members of a constellation responsible for each end-client. It is important to note, however, that the problem of selection an optimal number of members of MC that are responsible for a particular client is non-trivial. In some cases it is more efficient to have more than two MC members per client. For example, when EC is an anycasting group that is spread over the network.

For secure and/or reliable multicast delivery requests it is recommended that the number of allocated MC members be more than two to both increase reliability and, if the host hopping technique described below is used, improve security.

---

\(^1\)The question of how “close” a member of MC is to an EC involves measuring different types of metric, such as load, latency, packet loss and others. In particular, Host Distance Estimation Service [2] can be used.
4 Security

Manycasting has a potential of resolving many critical security issues that take place with the existing versions of multicast protocol. Presently, there has already been a significant amount of work done in the area of secure multicast communications [7]. Therefore, instead of replacing the work that has already been done, we will make use of it as one of manycasting extensions. Consequently, as new kinds of security threats are discovered, additional extensions will be added to address these new security threats and protect end-clients.

One of the security extensions that we are going to talk about is dynamic host hopping. The idea is illustrated in Figure 4. For simplicity, we will assume that there is just one end-client that requested secure delivery using host hopping. Thus, when MC receives a packet from a multicast source that the end-client registered for in its request, the packet is divided into several parts which are then sent to the end-client by different members of MC.

As shown on the figure, all four parts of a message are sent by four different constellation members Q1-Q4. However, even if there are only two alternating members sending various parts of multicast data, it still potentially increases the level of security.

Thus, if there is a sniffer between MC and the end-client that wants to intercept data, there is a high probability that it will only get certain parts of the original message which will prevent it from being able to reconstruct the original message. The efficiency of this technique will depend on network configuration and a point of
Figure 4: Dynamic host hopping
attachment of a sniffer. For example, when a sniffer is one hop away from EC and that EC only has one way of connecting to the internet, the sniffer will potentially be able to get all the parts of the original message.

Another security extension that is natively supported by manycasting constellations is Distributed Denial of Service Attacks (DDoS) or major network congestion problems protection. For instance, if there is a point of congestion between an end-client and certain members of a multicast constellation that serves that end-client, the data will be sent by the other members of the constellation, thus bypassing the point of congestion and increasing the chance of successive packet delivery.

Finally, since the MC keeps list of all its end-clients, it is possible to verify the identity of each client before starting sending data to that client. This includes various possible types of verification, such as digital certificates, Kerberos et al.

5 Extensibility

Manycasting is designed to be an extensible protocol. This means that when an end-client wants to add an extension the protocol itself will assist in this process. There are two basic types of manycasting extensions: generic and localized.

Generic extensions are in most of the cases independent fragments of code that can perform various functions, including multicast data preprocessing, interaction with end-clients, storing and retrieving various kinds of state-based information, and other functions provided by manycasting environment API (MEAPI\textsuperscript{2}). The major difference of generic extensions from localized ones is that the former can be requested by any of the end-clients. Therefore, generic extensions are static and thoroughly tested pieces of code.

Localized extensions, in contrast, can be added and removed dynamically, which makes them more vulnerable to various security attacks. Therefore, these extensions can generally be applied only to hosts that sent them or to the hosts that were successfully authenticated and explicitly requested to use a particular localized extension. Before any end-host can send an extension, the identity of that host must be verified to avoid security problems. Manycasting registrar's that add extensions also use soft state mechanism to make old extensions expire.

The part of manycasting architecture that provides support for extensions is a generic extension itself. This is why it is possible to accommodate various kinds of environments, including the one of application layer multicast [8]. By having an environment capable of executing application layer multicast code manycasting becomes an integrated environment for running various kinds of multicast extensions.

Examples of generic manycasting extensions:

- Anycast support
- Application layer multicast
- Secure multicast
- Host hopping

\textsuperscript{2}To be defined
• http tunneling
• Statistical analysis

Examples of localized mancasting extensions:
• Packet filtering based on certain criteria. For example, certain type of stock in a data received from stock exchange.
• Storing certain packets and dynamically determining current derivative to detect rate of change.

6 Intra-constellation messaging overview

In this section we will cover basic principles of functioning of a mancasting constellation as well as methods that are used by its members to interact. Members of a constellation can send several generic types of messages: Solicitation, update, or directive.

All these types can be delivered using either broadcast or unicast types of network transport. Broadcast is implemented using a multicasting group to which every member of a constellation is subscribed. In some cases, however, it is not efficient to use multicasting for intra-constellation messaging, especially when there is a small number of members. In such cases unicasting will be used exclusively.

A solicitation is a request for various types of information. The response to this type of message is based on its destination address specified in the destination address field of a mancasting packet. If the address is equals to "*" and the request is sent using broadcast, every member of a constellation will respond to the request if there is any data to be sent as a response.

If the address is anything other than an asterisk, it is interpreted as an IP address of a specific mancasting member and the response will only be sent by the mancasting member with the specified IP address if the member has any data to send. Responses are sent to the originator of messages using TCP protocol with a destination port set to 1534.

For example, a registrar may send solicitations whenever it is necessary to obtain statistical information from other members and select the one that fits a new end-client the best. Also, solicitations are sent to request various types of data, such as client’s statistics from other registrars, cached multicast data for a particular group or confirmation from additional members to participate in a host-hopping mechanism.

An update consists of additional information that is sent by a member to synchronize with other members, distribute important event descriptions, make an announcement of a new member, send rank updates and other kinds of information.

A directive can only be sent by core members and constitutes their decision of delegating certain responsibilities to other members of a mancasting constellation. For example, when a registrar assigns two members responsible for an end-client, the registrar sends a solicitation first to determine which members are the closest to the end-client and then sends directives to those members to cause them to subscribe to the appropriate multicast
group and start delivering data to the end-client.

All messages that are sent within a constellation are signed by their senders using their private keys to ensure authenticity.

7 Summary and Future work

Manycasting can be characterized as a protocol that combines various efforts to extend multicasting with a single, easily extensible, and transparent framework. This includes reliable multicast, secure multicast, anycasting, and all kinds of generic and client-defined extensions to add additional features.

Manycasting hierarchies or constellations are designed to be transparent and, therefore, end-clients are always able to revert to regular multicast without disrupting their normal functioning. One of the advantages of manycasting is that it does not require any modifications to the existing multicast protocols — it merely extends them.

The protocol makes use of a large number of Internet users that can all potentially contribute to packet delivery by becoming constellation members. For example, clients all over the Internet that want to participate may simply run a screen saver that registers with the closest constellation and participates in its work when a user is not active.

Since one of the basic principles of manycasting is sending data to clients based on network conditions, statistical data, and various extensions, the global consequences of using manycasting have a potential of making the Internet more balanced, secure, and reliable when it comes to multicasting.

Due to the size constraints and the fact that we came up with a relatively new concept, we conceived the purpose of this paper as giving an idea of basic concepts of manycasting protocol, its components, and functioning. Even though we have given some details related to message formats and features that are provided by manycasting constellations, we deliberately avoided getting into too much detail to be consistent with our goal.

We plan to consider each fundamental component of the manycasting protocol in a separate document with a sufficient number of details to implement the component. As a part of our future work, we also intend to focus on both refinement of the existing techniques used by manycasting and addition of what was mentioned but not extensively covered in this paper.

In particular, we will research the process of adding new members to a constellation, calculating ranks for existing members, statistical analysis of trends of latency observed with a dynamic set of end-clients, and the use of various transport protocols for distribution of data among members of a constellation.
References


XII