An XCAST Multicast Implementation for the OverSim Simulator

Mario Kolberg
University of Stirling
Stirling, United Kingdom

John Buford
Avaya Labs Research
Basking Ridge, New Jersey, USA

Abstract

The development of hybrid multicast simulation models is required for analyzing proposed hybrid multicast architectures such as those from the IRTF Scalable Adaptive Multicastw Research Group. However most network layer simulators don't scale to the number of nodes needed for analyzing large overlays, and most overlay simulators don't have multicast routing models needed for analyzing hybrid approaches. In this work we have extended the OverSim simulator and INET framework which run on OMNET++ to include a multi-destination multicast routing protocol (XCAST). This paper describes our implementation experience.

1. Introduction

Multicast is important for communications applications such as small group video conferencing and IPTV [1]. In hybrid multicast schemes, native multicast islands are interconnected using tunnels [2] or overlays [3][4][5][6][7]. A hybrid protocol has been defined [9][10] which is based on integrating AMT (Automatic Multicast Tunnel) tunneling mechanism for native multicast (NM) [10] with overlay multicast (OM) protocol mechanisms.

The development of hybrid multicast simulation models is required for analyzing these types of hybrid multicast architectures. However most network layer simulators don't scale to the number of nodes needed for analyzing large overlays, and most overlay simulators don't have multicast routing models needed for analyzing hybrid approaches. The goal of this work is to create a simulation environment for hybrid multicast which includes native multicast and overlay multcast mechanisms. The specific additions needed for simulating the SAM architecture are AMT, XCAST, and IGMP.

In this work we have extended the OverSim simulator and INET framework which run on OMNET++ to include a multi-destination multicast routing protocol (XCAST). This paper focuses describes our implementation experience.

The specific results are:

– Addition of the XCAST protocol to the INET framework
– Integration of XCAST messaging into the overlay routing layer used in OverSim.
– Design considerations for adding AMT protocol support to INET

The next section gives a brief overview of OverSim, OMNET++, and the INET framework. Section 3 summarizes XCAST, presents the XCAST implementation in OverSim, and provides preliminary evaluation. Section 4 discusses AMT support in INET, and section 5 concludes the paper.

2. OverSim

OverSim [14] has been developed by the Institute of Telematics at the University of Karlsruhe. It is built on top of the OMNeT++ network simulator [15][16][17] and features a large number of implemented overlay protocols. The INET framework implements the common Internet protocols. Hence OverSim is part of a simulator suite. The other main members of this suite, OMNeT++, INET, and ReaSE are introduced below.

OverSim includes the overlay simulation with a number of P2P overlay algorithms implemented. Amongst others, OverSim contains models for Chord, Pastry, Koorde, Kademlia and Bamboo. Simulations with 100.000 nodes have been reported. With this OverSim reaches reasonable overlay sizes. A further advantage of OverSim is that it also models of overlay multicast approaches exists. For instance, Scribe has been implemented on the Pastry model. Furthermore, OverSim offers an Application Programming Interface to implement further overlay multicast approaches. On the downside, Omnet++ contains only a partial implementation of Host Group Multicasting (IGMP is not included) and there are no multidestination multicast implementations, such as XCAST, available.

2.1 OMNeT++

OMNeT++ is a very modular simulation environment. At the most basic level, OMNeT++ defines modules which communicate using message passing. Modules are defined using the NED definition language. Specifically the modules external interfaces, such as gates and parameters are defined in the NED file. The behavior of modules is then directly implemented in C++ (simple module), but modules can also be created using other modules (compound modules). Such aggregated modules can be used to define the behavior of complex network components, such as routers or a standard host. Modules are interconnected by incoming and outgoing gates that specify channels. Modules exchange messages via gates and channels. Channels can possess certain bandwidth and delay characteristics.

Furthermore, OMNeT includes a message generator which uses a basic message definition to automatically generate the corresponding C++ code which can be extended if required.

In terms of a standard OMNeT simulation, at least two components are required. Omnet.ini contains global parameters and simulation options, such as which module starts the simulation and which modules execute application behavior. NED files include the definition of the network to be simulated and the modules used. Typically, each module is defined in a separate NED file specifying its gates, parameters and submodules. The top-most-level NED file contains the simulated network referencing modules and specifying their interconnections.

OMNeT++ includes a graphical development and simulation environment which supports countless features for debugging, running simulations, and visualizing simulation results. The GUI supports visualizing networks, nodes and messages, and allows for message and variable inspection inside these OMNeT++ components. For larger networks, and more complex simulations, OMNeT also features a command line simulation environment which allows to dedicate more computing resources to the simulation.

2.2 INET Framework

INET is an open source communications networks simulation package for OMNeT++. INET is used to simulate networks based on the IP protocol stack, specifically whose which use the MAC, IP and transport layers of the stack. INET provides implementations of protocols in these layers . as such INET, includes implementations for UDP, TCP, IP, IPv6, SCTP, Ethernet, PPP, IEEE 802.11, MPLS, and OSPF. Underlying is a simulation of IP queuing in nodes. INET supports static routing using network configurators, alternatively routing protocol implementations can also be used. Besides the communication protocols, INET defines the nodes which communicate using these protocols.

In the INET framework, protocols are represented as modules (NED file for interfaces plus behavior implementation). The NED language is then further used to combine these protocol modules to form network components, such as hosts, routers and switches. However, there are also special modules which do not implement any protocol. Common examples are RoutingTable, and FlatNetworkConfigurator. The former stores data (an IP routing table, whereas the latter assigns IP addresses to network nodes and sets up routing.

Protocol headers and messages are defined in msg files which are automatically translated into C++ code.

If a higher layer protocol wants to send a packet over a lower layer protocol, the higher layer protocol sends the message object (representing the packet) to the lower layer protocol. The lower layer protocol will encapsulate this message object and send it on. At the receiving side the reverse process will occur. The lower layer protocol will receive the encapsulated message object, remove the lower layer information and then pass the message object to the higher layer.

Associated information with a packet is transmitted as control info. Control info can be some connection identifier used by an application layer protocol when sending data over a transport protocol. In turn, when a transport protocol sends data over IP, an IP address is required. Control info is extra information which is attached to the message object (packets). Control info contains information for the next protocol layer and is not actually sent over the network to other components – just used for communicating down the stack.

2.3 ReaSE

ReaSE (Realistic Simulation Environments) [22] is a standalone tool which generates networks and simulation environments for the OMNeT++ simulator. It uses the INET framework as a base. ReaSE can generate network topologies, including different administrative domains (AS level – autonomous systems) and router level, as well as traffic patterns and attack traffic. For the AS level topologies, ReaSE uses the positive feedback preference model, which generates random topologies with power-law distributed node degree. At router level the topologies generated contain few meshed core nodes with a low node degree which forward aggregated traffic of a high number of gateway nodes with a high node degree. Finally edge nodes have a node degree of 1 and connect host systems to the network. This means that the link bandwidth increases from edge to core, whereas the connectivity decreases.

ReaSE works together well with OMNeT++ and INET. In fact ReaSE includes and extension to the INET framework to support hierarchical addressing and routing as well as self similar background traffic. ReaSE further includes a graphical user interface which allows for easy configuration of the topology and traffic parameters.

ReaSE generates NED files which are used as inputs for OMNeT++ simulations. The nodes in the NED files are represented by either the module Router or the module StandardHost depending on the topology. Other types of nodes, such as gateways and core routers are represented by assigning different bandwidths to the links, that is core-core and core-gateway channels. The actual size of the topology, i.e. the number of routers and the link bandwidth can be configured before topology generation.

3. XCAST and Multi-destination Routing

Multi-destination routing is not a new concept, in fact it was devised during the early stage of multicast design [19]. However, as it is not capable of supporting large multicast groups it did not attract large attention initially. Recently it has been recognized as a technology which offers scalability with respect to numbers of multicast groups. Multi-destination routing hence has its strength in supporting a large number of small groups.

The basic approach of multi-destination routing is quite simple: rather than sending a number of unicast messages to multiple destinations, a single message is sent which includes all the destination addresses. Multi-destination enabled routers make a routing decision on each of these addresses, and if addresses result in different routing decision, the message is duplicated and the corresponding subset of addresses is attached to the messages. Once a message only contains a single address, normal unicast behavior is resumed.

In Figure 1 this concept is illustrated: Node A is sending a message to Nodes B, C, D and E using multi-destination routing, hence the message originating at node A includes 4 addresses. At the first router, the routing decision for all 4 destination is identical, hence the message stays as one including all 4 destination. At the next router, a split occurs and hence the message is duplicated. One message is now sent towards nodes B and C (including two addresses) and a second message is sent towards nodes D and E (also including two addresses). These messages stay intact until the final router before the nodes where a final split occurs into 4 unicast messages. The number on the links indicates the number of addresses in the message (and the number of unicast messages which would be required).

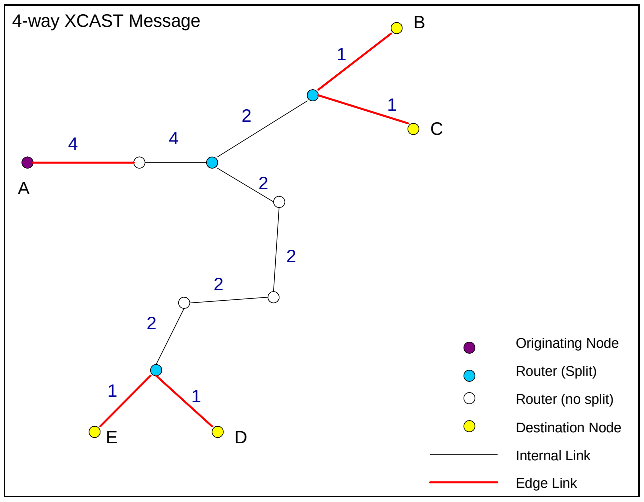

Figure 1: Sending an XCAST message to four destinations.

Another advantage of multi-destination routing is that it does not require state in the routers. The group information is part of the message. The routers check the routing path for each address, very much the same as for unicast messages. Hence in terms of routing decisions, multi-destination routing and unicast exhibit similar performance. However, clearly multi-destination routing cuts down the number of identical messages sent over a link.

XCAST (explicit multicast) is an IP protocol implementing multi-destination routing [20]. He and Ammar have analyzed the performance of XCAST [21].

3.1 The XCAST Implementation in Omnet++

An XCAST implementation in OMNeT++ impacts heavily on the INET framework. XCAST principally works at the IP layer. However, as XCAST is a protocol used by the sender nodes and network nodes, OMNeT++ applications, transport layer protocols (UDP), and the IP layer, as well as the corresponding nodes need to be adapted.

The implementation changes the INET framework in that a number of classes have been derived and augmented with XCAST specific behavior. Hence only behavior and code which had to be adapted actually changed. This allows for a realatively straightforward integration of XCAST in OMNeT++/INET simulations.

For an XCAST message to be sent, the application needs to specify multiple destination addresses for this message. Taking the UDP Application which is part of the INET framework as an example, the destination addresses are specified in the omnet.ini as a parameter. The default behavior is that one of the specified addresses is selected at random and the message is sent to this destination. For XCAST we have developed a new application which changes this behavior in that the message is sent to all destination addresses using an XCAST message. The destinations are part of the UDP control info data structure. Hence the UDP control info was amended to be able to specify multiple destination addresses. Furthermore, the protocol implementation of UDP had to be altered to cope with multiple destination addresses. This included both, the NED definition and the C++ behavior implementation. Clearly, having multiple destination addresses in the message also affect the UDP datagram.

UDP then passes the UDP datagram down to the IP layer, and with it the IP control info. Again the IP control info data structure and also the IP datagram were amended to include multiple destination addresses. The IP implementation also has to make the routing decision on all destination addresses for a message. To achieve this, the routing function was amended to look at all destination addresses and decide if the packet needs to be duplicated. This is the case if the routing decision for addresses differs, that is messages to different addresses are routed down a different path. The routing function will create a duplicate message if required, and attach the corresponding destination addresses. The IP packets are then sent. No changes were required at any lower layer protocols.

As the message traverses the network, the message will be duplicated and fewer addresses will be attached until the message reverts back to a unicast IP/UDP message.

In order to support this protocol behavior, key nodes in the network such as Router and StandardHost also had to be adapted. New versions of these components which support XCAST versions of the IP and UDP protocols were created.

For testing purposes, a numbe of sample networks were generated using the ReaSE tool. Clearly, ReaSE generates networks using the standard modules, rather than the XCAST versions. However, amending generated NED files is straightforward: changing the #include files to the appropriate XCAST versions, and a simple find-replace of all routers and StandardHost to the XCAST versions (RouterXCast, StandardHostXCast). No changes are required to any parameters to these modules in the NED file.

In the omnet.ini file also only very minor changes are required. The udpAppType for a module supporting XCAST needs to be changed to UDPAppXCast, and the destination addresses can then be specified in the destAddresses parameter.

3.2 Sample Output from the Simulation

Simulations using the XCAST modules need to be able to report on the performance gain achieved using XCAST. So far only a basic reporting of simulation performance values has been implemented. Whenever a message is sent at the IP layer, a vector output is generated reporting the number of destination addresses within this message. Using this the reduction in duplicate messages sent can be calculated.

Using the data-analyzer built-in with the OMNeT++ IDE, the values can be inspected and plotted. A sample result is shown in Figure 2 below.

In future versions further evaluation metrics will be included, such as bandwidth savings.

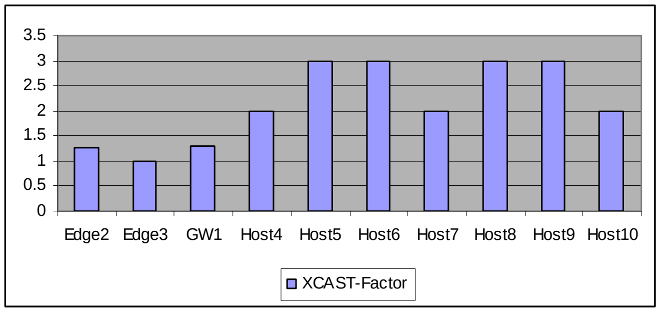

Figure 2: Plot of simulation output using the XCAST module.

4. Adding AMT to INET Framework

In order to evaluate the hybrid multicast framework described in [9][10], INET must also implement AMT [11] and IGMP (Internet Group Management Protocol). Here we briefly summarize the requirements and changes needed.

AMT components are either a gateway or a router. The AMT GW can be implemented in either a host or a router. There are six message types in the protocol. The messaging is shown in Figure 3. The first two messages represent the advertisement and discovery exchange by which a gateway discovers a router. The next three messages are the handshake by which the GW and router set up a connection. Thereafter, multicast data can be sent encapsulated in AMT IP Multicast Data messages.

The AMT discovery mechanism requires anycast addressing support. Anycast addressing is not currently supported in INET. Further, the routing layer does not have BGP routing mechanism. All AMT messages are UDP packets.

We propose three AMT modules are needed: AMT-GW, AMT-Router, and AMTApp. The AMT-GW module implements both the six message types to the AMT-Router. It also acts as an IGMP proxy on the local network. The AMT-App module is needed for endpoint multicast apps running on the same host as the GW to participate in an AMT connection. The AMT-Router module supports the ATM router side of the messaging to the GW, and connects to other multicast-enabled routers.

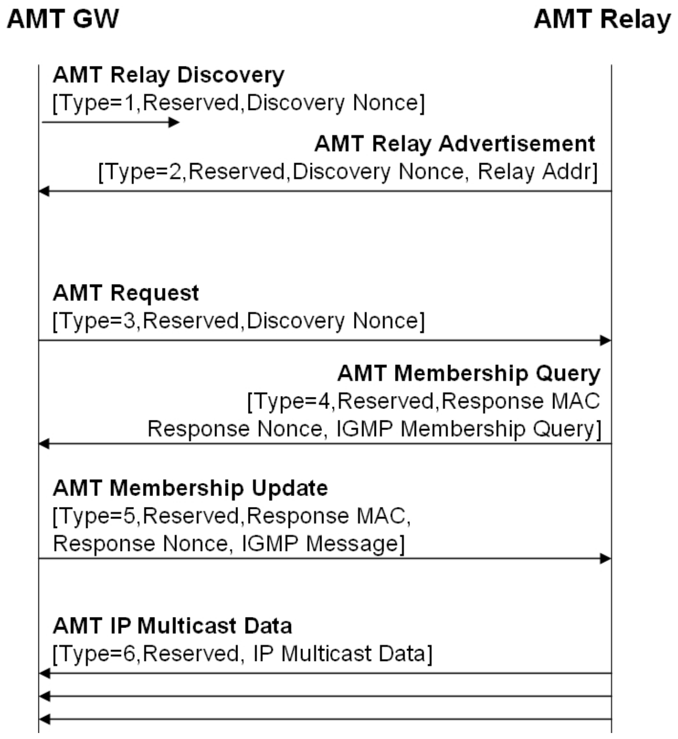
AMT GW
AMT Relay
AMT Relay Discovery
[Type=1,Reserved,Discovery Nonce]
AMT Relay Advertisement
[Type=2,Reserved,Discovery Nonce, Relay Addr]
AMT Request
[Type=3,Reserved,Discovery Nonce]
AMT Membership Query
[Type=4,Reserved,Response MAC
Response Nonce, IGMP Membership Query]
AMT Membership Update
[Type=5,Reserved,Response MAC,
Response Nonce, IGMP Message]
AMT IP Multicast Data
[Type=6,Reserved, IP Multicast Data]

Figure 3: Example AMT message sequence

5. Conclusions and Further Work

This paper presents preliminary work towards simulating and evaluating a new type of hybrid multicast protocols. As a first step we have implemented the XCAST protocol in the OverSim/Omnet++/INET simulator environment. Due to a simple interface, this new component can easily be integrated with other existing simulations. As next steps we plan to extend the performance metric reporting facility and to integrate this the XCAST component with OverSim and Application Layer Multicast approaches. In addition, AMT support should be added as described in section 4.

6. References

[1] J. Buford, M. Kolberg. Hybrid Overlay Multicast Simulation and Evaluation. IEEE CCNC 2009 (short paper). Jan. 2009.

[2] B. Zhang, S. Jamin, and L. Zhang. Universal IP multicast delivery. In Proc. of the Int'l Workshop on Networked Group Communication (NGC), Oct. 2002

[3] X. Jin, H.-S. Tang, S.-H. Chan and K.-L. Cheng, Deployment Issues in Scalable Island Multicast for Peer-to-Peer Streaming," IEEE Multimedia Magazine, vol. 16, issue 1, pp. 72-80, Jan.-Mar. 2009.

[4] X. Jin, K.-L. Cheng, and S.-H. Chan, Scalable Island Multicast for Peer-to-Peer Streaming, Hindawi Journal of Advances in Multimedia special issue on Multimedia Networking, vol. 2007, Article ID 78913, 2007.

[5] X. Jin, K.-L. Cheng, and S.-H. Chan, SIM: Scalable Island Multicast for Peer-to-Peer Media Streaming, in Proceedings IEEE International Conference on Multimedia Expo (ICME), pp. 913-916, Toronto, Canada, 9-12 July 2006.

[6] M. Waehlisch, T. Schmidt. Multicast routing in structured overlays and hybrid network. in: Handbook of Peer-to-Peer Networking. (eds. S. Shen, H. Yu, J. Buford, M. Akon). Springer-Verlag. Forthcoming.

[7] M. Wählisch, T. C. Schmidt, G. Wittenburg. A Generalized Group Communication Network Stack and its Application to Hybrid Multicast, In: Proceedings of the 28th IEEE INFOCOM. Student Workshop, IEEE Press, April 2009.

[8] J. Buford, S. Kadadi. SAM Problem Statement. Dec' 06. Internet Draft draft-irtf-sam-problem-statement-01.txt, work in progress.

[9] J. Buford. Hybrid Overlay Multicast Framework. IRTF SAM RG. draft-irtf-sam-hybrid-overlay-framework-02. March'08, Work in Progress.

[10] J. Buford. SAM Overlay Protocol. IRTF SAM RG. draft-irtf-sam-overlay-protocol-00.txt, Feb 2008. Work in progress.

[11] D. Thaler, M. Talwar, A. Aggarwal, L. Vicisano, T. Pusateri. Automatic IP Multicast Without Explicit Tunnels (AMT). Internet Draft draft-ietf-mboned-auto-multicast-09, Work in progress. June 2008.

[12] J. Buford, A. Brown, M. Kolberg. Exploiting Parallelism in the Design of Peer-to-Peer Overlays. J. Computer Communications. Special Issue on Foundations of Peer-to-Peer Computing Vol 31/3, Feb. 2008, pp 452-463.

[13] SSF-NET: http://www.ssfnet.org/homePage.html

[14] I. Baumgart, B. Heep, S. Krause. OverSim: A Flexible Overlay Network Simulation Framework, Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA, May 2007.

[15] A. Varga. Omnet++ community site. http://www.omnetpp.org

[16] A. Varga. The OMNeT++ Discrete Event Simulation System. Proceedings of the European Simulation Multiconference (6 June 2001), pp. 319-324.

[17] A. Varga, R. Hornig. An Overview of the OMNeT++ Simulation Environment. Proceedings of SIMUTools 2008: 1st International Conference on Simulation Tools and Techniques ICST, Marseille, France, Mar 2008.

[18] Y.-H. Chu, S. G. Rao, and H. Zhang. A Case for End System Multicast. In Proceedings of ACM SIGMETRICS, June 2000.

[19] L. Aguilar, Datagram Routing for Internet Multicasting, Sigcomm 84, March 1984.

[20] R. Boivie, N. Feldman , Y. Imai , W. Livens , D. Ooms, O. Paridaens, Explicit Multicast (Xcast) Basic Specification, draft-ooms-xcast-basic-spec-11.txt, Work in Progress. Jan. 2007.

[21] Q. He, M. Ammar. Dynamic Host-Group/Multi-Destination Routing for Multicast Sessions. J. of Telecommunication Systems, vol. 28, pp. 409-433, 2005.

[22] T. Gamer, M. Scharf. Realistic Simulation Environments for IP-based Networks. Proceedings of 1st International Workshop on OMNeT++ (Hosted by SIMUTools '08: 1st International Conference on Simulation Tools and Techniques), ICST, Marseille, France, Mar 2008.