QS-XCAST: A QoS Aware XCAST Implementation

Odira Elisha Abade
Graduate School of Engineering, Nagoya University, Japan
Furo-cho, Chikusa-ku, 464-8603,
abade@ucl.nuee.nagoya-u.ac.jp

Katsuhiko Kaji
Graduate School of Engineering, Nagoya University, Japan
Furo-cho, Chikusa-ku, 464-8603
kaji@nuee.nagoya-u.ac.jp

Nobuo Kawaguchi
Graduate School of Engineering, Nagoya University, Japan
Furo-cho, Chikusa-ku, 464-8603
kawaguti@nagoya-u.jp

ABSTRACT
Explicit multiunicast (XCAST) is a complementary protocol that can solve the scalability problems of IP multicast by providing a stateless design that encapsulates multicast information in an IP packet header. There is a need to test new protocols like XCAST with network simulators that offer models which can scale to a large number of nodes. However, existing simulators like OMNeT++ do not have IP header encapsulation and routing models required for analyzing XCAST, especially its implementation on IPv6 protocol stack, called XCAST6. Further, XCAST is designed for applications such as videoconferencing and IPTV systems that are loss and delay sensitive hence they need some form of QoS provisioning. However, most research in XCAST is only focus on its performance and implementation, ignoring the important facet of QoS assurance. In this paper we describe the implementation of XCAST6 on the IPv6 protocol stack of INET Framework for use with OMNeT++. We also extend the QoS classes in INET Framework to allow for simulation of XCAST6 integrated with Differentiated Services architecture for QoS provisioning. This paper therefore also serves to open up research for XCAST QoS provisioning.

Categories and Subject Descriptors
C.2.2 [**Computer-Communication Networks**]: Network Protocols.
C.2.5 [**Local and Wide Area Networks**]: Internet.

General Terms
Performance, Design, Experimentation

Keywords
XCAST, QoS, DiffServ, OMNeT++, INET Framework, IPv6, Simulation.

1. INTRODUCTION
The Internet has witnessed rapid growth in the past decades which has resulted in merging of both data and real-time multimedia traffic. However, availability of adequate bandwidth at low cost continues to be one of the challenges faced on the Internet.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
OMNeT++ 2012, March 19-23, Desenzano del Garda, Italy
Copyright © 2012 ICST 978-1-936968-47-3
DOI 10.4108/icst.simutools.2012.247761

ensuring that for group communication, a user sends only one packet which then gets replicated only at definite points in the network when necessary. Deploying IP Multicast in routers has however met several challenges among them; scalability issues, congestion control, complex tree construction algorithms and complications in providing Quality of Service for Multicast applications especially when using Differentiated Services architecture [1].

XCAST [5] utilizes IP header encapsulation to provide a stateless method for sending data to multiple recipients. However, XCAST's header encapsulation and routing model are currently not implemented in most network simulators. Its QoS has also received only a very little attention. In [8], Siregar et al only suggest that per-packet dynamic routing for unicast can be used in XCAST but they do not show how it can be done.. In [9], they proposed a new modified IPv6 extension header they call "*IPv6 QoS header,*" which contains a QoS value. The QoS value is to be calculated by routers based on the number of users underneath a router, total users requesting the video streams and the available priority levels which also depend on some probability assignment. The prioritization and how the probability values are calculated and allocated are not explained. Since XCAST header is already complex for existing routers, we propose to leave it as specified in [5] but use DiffServ for QoS provisioning. However, integrating DiffServ QoS with XCAST is currently unexplored. In order to open up the QoS research in XCAST, this paper explores implementation of XCAST on the IPv6 stack of the INET Framework for OMNeT++ and extends the INET's QoS classes to realize integration of DiffServ with XCAST.

In the next section we describe XCAST protocol on IPv6 (XCAST6). Section three describes our implementation of XCAST6 in INET Framework and introduces a sample XCAST6 application for OMNeT++. We then present a simple model for testing the QoS aware XCAST in section four and discuss the results.

2. XCAST6 PROTOCOL
XCAST6 refers to the implementation of explicit muliunicast(XCAST) on IPv6[4,5]. In XCAST6, the sender explicitly specifies the destination addresses of all the receivers as a list of unicast addresses embedded in the IPv6 packet header. Succinctly, the sender embeds a list of destinations in the routing extension header of the IPv6 packet then sends the packet to a router. Along the transmission path, each router examines the IPv6 packet header and determines the next-hop for each destination in the specified list. The router then groups together the destinations with the same next-hop and forwards a packet with an appropriate XCAST6 header to each of the identified next hops. The process is repeated until all the destinations are reached.

The XCAST6 header also comprises of a bitmap with bits corresponding to each destination, which the routers use to determine which of the embedded destinations the packet needs to be delivered and to which ones a copy of the packet has already been delivered[5].

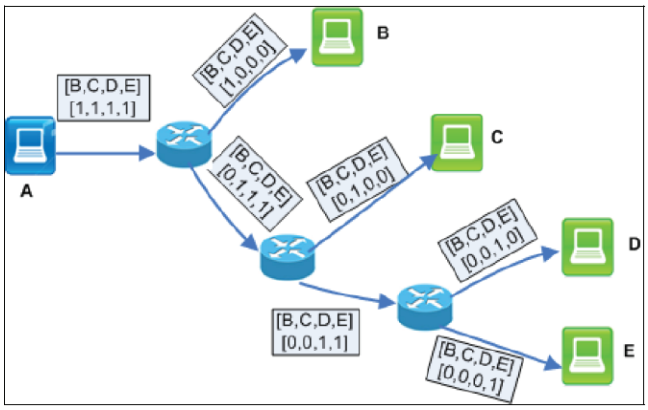

Figure 1. XCAST6 Overview

Each of the branching routers updates this bitmap for each copy of XCAST6 packet during replication. In figure 1, the sender (A) sends an XCAST6 packet to B,C,D and E. The destination addresses B,C,D,E have corresponding bitmaps which if set to 1 means the packet is to be delivered to a corresponding destination and if reset, means otherwise. On each XCAST6 router, if need be, the XCAST6 packet is duplicated, bitmaps updated and delivered upward according to the destination's next hop with respect to the current branching router.

3. ADDING XCAST6 TO IPV6 STACK OF THE INET FRAMEWORK

XCAST routing model is currently not supported by most of the existing network simulators, especially its special approach of encapsulating a list of destinations and a bitmap in the IP header. In [3], Kolberg et al have proposed an implementation of XCAST for Oversim and OMNeT++ which in the current state, is only on the IPv4 protocol stack of INET Framework. Additionally, their implementation is an XCAST model whereby an XCAST packet gets duplicated at branching nodes and fewer addresses are attached to each copy of the packet until each copy reverts back to a unicast IP/UDP message as the packet traverses the network. However in [4,5] it is specified that this complex header reconstruction at every branch can be reduced to a simple operation of updating a bitmap in the XCAST6 header and leaving the destinations intact in all copies of the XCAST6 packets. Furthermore, the use of bitmaps instead of the header reconstruction has the advantage that it allows for the possibility of using XCAST with IPsec since only the bitmap is modified and not the entire IPv6 header.

Our target was the implementation of XCAST simulation on the IPv6 stack as detailed in [7] in addition to real world XCAST6 [6]. We further seek to open up QoS research in XCAST networks, especially on integration of DiffServ architecture into XCAST networks. This work also implements the XCAST6 concept where the bitmaps are used instead of the complex header reconstruction process [4, 5]. Further we improve on the QoS Classification of INET Framework to integrate XCAST with DiffServ QoS. Our approach utilizes much of the functionalities in the INET Framework's network layer modules with significant changes done only in the network layer modules. Since the key modules such as *Router6* and *StandardHost6* inherit from most of the network layer modules like *IPv6* and *IPv6ControlInfo*, our implementation does not require any adaptation whatsoever to the existing *StandardHost6* and *Router6* modules.

3.1 Network Layer

We extensively extended a number of classes to implement XCAST6 functionality while some other classes were only slightly altered by adding a few lines of code for XCAST6 specific functionality. In [4,5], XCAST6 destinations and the bitmap are contained in the IPv6 routing extension headers. We modified the *IPv6ExtensionHeader.msg* file to include these XCAST6 specific attributes in the INET Framework's IPv6 routing extension header. This in effect adds these two data containers into the generated *IPv6ExtensionHeader* class. To facilitate inter-protocol layer transmission of XCAST6 controls, we added the XCAST6 traffic class, destination addresses list and the bitmap in the *IPv6ControlInfo* module. Traffic class of XCAST6 header is *"010111"*, (*"23"* in decimal). However, this work aims at providing XCAST6 QoS using DiffServ in which the DSCP-PHBs are stored in IPv6 header's traffic class. So for this implementation, we identified XCAST6 packets using the *flow label field* of the IPv6 header. Therefore any IPv6 traffic in the simulation with a flow label of 23 (*"010111"*) is recognized as XCAST6 traffic and this information is shared between protocol layers using the *IPv6ControlInfo* module. The *IPv6Datagram* module was also amended to be able to handle the new XCAST6-capable *extension header* modules.

In INET Framework, routing decisions are made in the IP layer. To allow for XCAST6-aware routing, we added a method called *routeXcast6Packet()* in the *IPv6 class* and also amended the message handling method of the *IPv6 class* to be able to identify XCAST6 messages and handle them by invoking our new XCAST6 routing method.

INET Framework's default *IPv6FlatNetworkConfigurator* assumes that all nodes are in the same subnetwork. To work with numerous subnets, we used a *NETCONF-style* XML file detailing routing tables and IP addresses of the nodes within the model network. We therefore modified the *RoutingTable6* class such that at stage3 of the initialization process it invokes our newly defined method called *parseXMLConfigFileForStaticRoutes()*. In this method we parse the XML routing file and use the route information to either invoke the *addDefaultRoute()* or *addStaticRoute()* methods of the *RoutingTable6* class.

3.1.1 Differentiated Services (DiffServ) QoS

The INET Framework has the basic DiffServ infrastructure in place. We defined a new class for QoS classification called *XCASTQoSClassifier* which extends IQoSClassifier and implements fourteen Per-Hop-Behaviors(PHB) defined in the DiffServ architecture[1,2]. This class is used together with the *DropTailQoSQueue* module to provide XCAST QoS classification. The classifier implements an array of DropTailQueue instances. The most demanding DSCPs are scheduled in the top DropTailQueue instances in the array to be processed at a higher priority than those in the rear end of the array. We also implemented DSCP processing in XCAST from the list of embedded DSCPs such that the effective DSCP-PHB of the XCAST packet is that of the most demanding DSCP from the embedded list.

3.2 Transport Layer

We modified the *UDPControlInfo* module to be able to handle multiple destinations as required by XCAST6. We therefore set the destination address of the *UDPControlInfo* to *ALL_XCAST_NODES* (*"ff0e::114"*) [4,5] while the list of Unicast destinations and the bitmap are contained in our newly defined containers in this module. There are no further modifications and derivations needed in the implementation of UDP protocol since we use *UDPControlInfo* and *IPv6ControlInfo* to exchange the XCAST6 information between protocol layers.

3.3 Application Layer

We developed a simple UDP application which can be used to test XCAST. In the *omnetpp.ini* file of our sample model, if the 'xcast6' parameter is set to 1, XCAST6 is enabled otherwise the application sends unicast packets.

3.3.1 QoS Aware XCAST sample applications

We specify parameters with the list of IP addresses and the corresponding DSCP classes to be used by the XCAST6 receivers from the *omnetpp.ini* file. Instead of randomly selecting only one destination to send to as is the case with the INET's *UDPBasicApp*, our application sends a prioritized data to all the specified receivers at once.

3.4 Source Code URL

The XCAST6-capable INET Framework files are here: https://sourceforge.net/projects/xcastonomnet/

4. SIMULATIONS AND RESULTS

We model an IP Television (IPTV) service provider network with service plans classified for bandwidth pricing that map onto *EF*, *AF41*, *AF31*, *AF21*, *AF11* and *BE* DSCP-PHB classes [1][2]. The model network comprises of 29 (13 core and 16 edge) routers. Each edge router is connected to at least 5 hosts. One host is the source that sends data to all the remaining hosts. XCAST messages of 1450 bytes payload size are scheduled at 50ms and a background non-XCAST, TCP traffic scheduled at an exponentially distributed frequency is also run in the network. The MAC TX rate for all router interfaces was set to 100Mbps. Receivers are assigned various DSCP classes as explained above then throughput and average delay per hop are measured at the receivers. We then compare XCAST and UNICAST based on these two metrics.

4.1 Throughput and delay analysis

As shown in figure2 and figure 3, XCAST performs better than Unicast in both throughput and average per-hop delay measurements except for the *EF* class. EF Unicast performs better than XCAST in *EF* class because Unicast delivers to one host at a time but XCAST has to process the header for all embedded decisions in each case even though the classes are of the same priority. Generally, for each of the two metrics, Unicast throughput degrades faster with decrease in priority of the DSCP-PHB classes. The average per-hop delay for Unicast packet also increases as the DSCP-PHB priority reduces. However for XCAST, all DSCP classes other than EF register higher throughput compared to their corresponding values in Unicast. This is because XCAST sends data to all receivers in one packet hence the lower priority DSCP classes get near-optimal treatment since the DSCP of the packet is set to the most demanding DSCP from the list of DSCPs of the receivers.

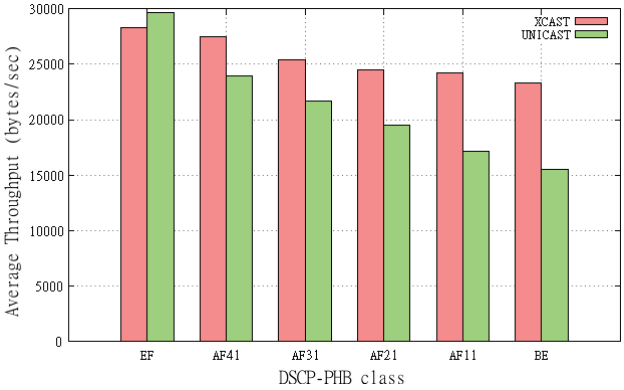

Figure 2. Comparative average throughput

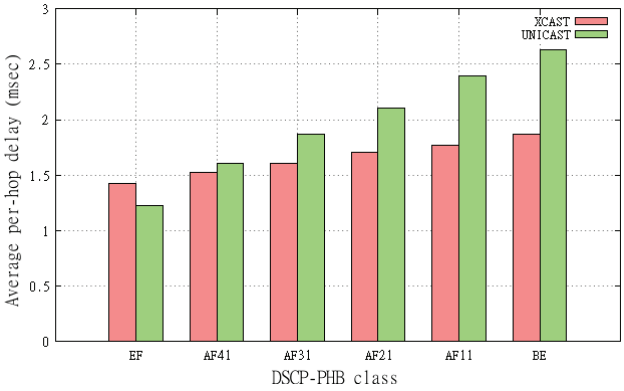

Figure 3. Comparative average per-hop delay

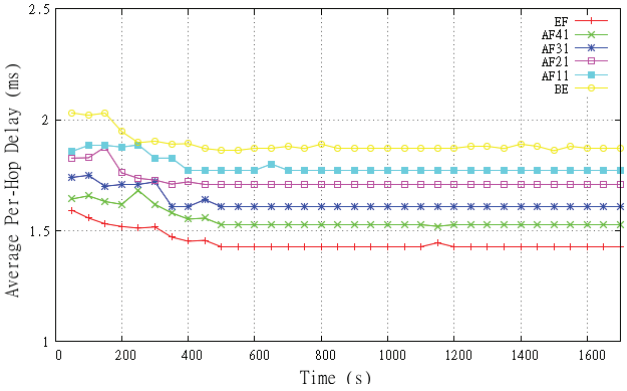

Figure 4. Average per-hop delay for XCAST

Figures 4 shows that delays of all other classes coalesce around that of *EF* traffic in XCAST compared to Unicast traffic in figure 5 where each class is more distinct. We also observed significant *IPv6 neighbour discovery* and *router advertisement* traffic at beginning of each run which we attribute to the spikes in the first 200 seconds especially for Unicast (figure 5).

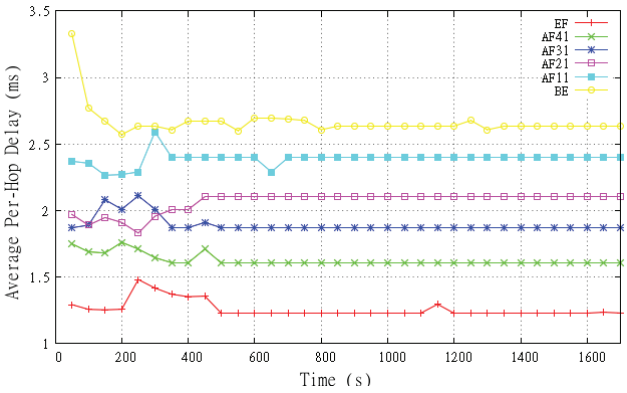

Figure 5. Average per-hop delay for Unicast

5. CONCLUSION AND FUTURE WORK

This paper presents an implementation of XCAST6 protocol into OMNeT++ and DiffServ QoS integration with XCAST. Our approach is simple and focuses on the key classes of the INET Framework that are shared by many modules hence ensuring that such modules are not re-customized to support XCAST. We find out that XCAST performs better than Unicast due to its lower delay and higher throughput for all DSCP classes. We hope that this work serves to open up research in QoS facet of XCAST. As next step, we shall use this base to perform comprehensive analysis on XCAST QoS to identify possible challenges in integrating DiffServ and XCAST.

6. REFERENCES

[1] S. Blake, D. Black, E. Davies, Z. Wang and W. Weiss. An Architecture for Differentiated Services, RFC2475, December 1998

[2] D. Grossman, New Terminology and Clarifications for Diffserv, RFC 3260, April 2002

[3] M. Kolberg and J. Buford. An Xcast multicast implementation for the oversim simulator. Proceedings of Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE, January 2010.

[4] Y. Imai, T. Kurosawa, and E. Muramoto. Xcast6 (version 2.0) protocol specification,. Internet Draft, draft-ug-xcast20-protocol-spec-00.txt, February 2008.

[5] R. Boivie, N. Feldman, Y. Imai, W. Livens, and D. Ooms. Explicit multicast (xcast) concepts and options. RFC 5058, November 2007.

[6] O. E. Abade, K. Kaji, and N. Kawaguchi, Design, Implementation and Evaluation of a Routing Engine for a multipoint communication protocol: XCAST6. International Journal of Computer Science and Network Security, VOL.11 No.5, May 2011

[7] O. E. Abade, K. Kaji, and N. Kawaguchi, Quantitative Simulation of XCAST6 Performance Using OMNeT++, Proceedings of AINTEC'11, Nov. 9-11, Bangkok, Thailand.

[8] L. Siregar, R. Budiarto, M.N. Omar, A.H. Rosli. Quality of Service Performance for Xcast in IPv6 Network. Proceedings of DFmA 2008, Oct. 21-22 2008.

[9] L. Siregar, R.F. Aji, Z.A. Hasibuan, R. Budiarto. Quality of Service For IPTV Using Xcast in IPv6 Network. Proceedings of NETAPPS 2010, 22-23 Sept. 2010.