

On the Efficiency of Explicit Multicast Routing Protocols

Mozafar Bag-Mohammadi, Nasser Yazdani, Siavash Samadian-Barzoki
Router Lab., University of Tehran, Tehran, Iran
{mozafarb, s.samadian}@ece.ut.ac.ir, yazdani@ut.ac.ir

Abstract

Recently, many explicit multicast schemes have been proposed to serve a very large number of low to fairly moderate size multicast groups in a scalable manner. In this paper, we discuss the design trade-offs of these protocols both through simulation experiments and real-trees analysis. Our metrics for comparison are the header size and processing overheads. We also introduce a modification to *ERM* (Explicit Route Multicast), called *Bcast* (Branch cast), which removes some inefficiencies of *ERM* and reduces its size overhead. In *Bcast*, sender host generates the tree description code(s) and inserts it into header of multicast data packets. The code contains the IP addresses of all receivers and branching points of the tree. *Bcast* has a proactive bypassing mechanism which helps it to adjust the code size in response to inconvenient distribution of the receivers.

1. Introduction

Traditional IP multicast routing protocols [14] [21] require group-related state maintenance in the on-tree routers, commonly known as Multicast Forwarding Table (MFT), in order to forward multicast packet properly. The state maintenance idea contradicts stateless philosophy of the IP routing which declares that network devices (i.e. routers) might keep minimum state for routing purposes. There are many small to moderate size multi-party applications such as video and audio conferencing, IP telephony and network games, which can not be well serviced by the current model in the large scales. Explicit multicast protocols are specifically designed to serve this class of applications in a scalable manner.

The basic idea in explicit multicast is to provide sender with receivers' information such as their IP addresses or their paths towards the sender. Then, the sender generates a proper header and attaches it to every multicast data packet. The header is investigated

by intermediate routers to forward multicast data packets. In explicit multicast, group size is inherently limited to a very small number in order to restrict processing complexities and header size overhead. *Xcast* [4] [5], its variation *Xcast+* [18], *Linkcast* [1] and *ERM* (Explicit Route Multicast) [3] are examples of explicit multicast protocols. In *Xcast* and *Xcast+*, a list of the destination IP addresses is sent with each multicast packet. In *ERM*, a tree code is sent with each multicast data packet. The encoded tree contains the IP addresses of receivers and branching points of the tree. *Linkcast* encodes the tree using the ID of on-tree links.

In this paper, we first propose *Bcast* (Branch Cast) as a modification to *ERM*. Then, we compare explicit multicast protocols through detailed discussions and simulation analysis. We also use the real-tree statistics collected in *mwalk* project [10] to evaluate efficiency of the protocols under a more realistic condition. Our findings imply that the basic approach, i.e. *Xcast*, suffers from the processing overhead mainly due to the existence of superfluous unicast lookups in the packet forwarding plane. *Bcast* and *ERM* reduce the processing overhead of *Xcast* and solve the deployment problem of it as well. But, they increase header size overhead. *Linkcast* shows many promising features for supporting fairly large multicast groups (less than 50 receivers). Although *Linkcast* minimizes the processing overhead, it decreases the availability of the offered multicast service. Similar to other multicast routing protocols which use unicast destination address for multicast data packets *HBH* [13] *REUNITE* [19], *Bcast* and *ERM* can be deployed gradually. *Bcast* sender could adjust the size of tree code in response to undesirable distribution of the receivers. This capability is useful when size of multicast group increases during session lifetime.

In section two, we briefly describe current proposals for explicit multicast. The key components of *Bcast* are discussed in section three. We compare explicit multicast protocols in section four. Section five describes the simulation results and real tree analysis. Finally, section six concludes the paper.

2. Explicit multicast

Xcast [4][5] was originally designed to overcome scalability and deployment problems of the traditional multicast routing protocols like PIM-SM[14] and DVMRP[21] in supporting very large number of low to moderate size multicast groups. In *Xcast*, multicast sender inserts a list of receiver IP addresses in a special header and sends it with all data packets. Each router along the way partitions the destinations into different sets based on their next hops. Then, for each set, it forwards a packet with an appropriate *Xcast* header, which contains all destinations in that set.

Xcast+ [18] introduced a simple modification to *Xcast* combining it with the well-known IGMP (Internet Group Management Protocol) [8] protocol. In *Xcast*, two or more receivers in the same LAN are treated as separate receivers. However, they are represented by a LAN Designated Router (DR) in *Xcast+*. This is achieved by adding IGMP (Source, Group) join at receivers' side and sending join requests toward sender, and by encoding the address of DRs instead of the receivers' addresses in the *Xcast+* header. Therefore, *Xcast+* is more efficient than *Xcast* when some receivers reside in the same LAN.

Explicit Route Multicast (*ERM*) [3] is another multicast scheme, which utilizes packet header in order to forward multicast data packets. In *ERM*, the tree is encoded in terms of its final destination and branching routers and included in each multicast data packet. *ERM* uses the DR that directly attached to the receivers as final destinations. The information to build the delivery tree is acquired by collecting *trace* messages from the receivers. In addition, the destination IP address field in IP header is replaced with address of the next branching router, enabling *ERM* to use unicast forwarding between branching routers.

Sender Initiated Multicast (SIM) [20] and Simple Explicit Multicast (SEM) [6] are *Xcast*-like schemes which use the receivers' list to construct multicast distribution tree. Although their tree construction processes use explicit multicast techniques, they maintain MFT in the branching routers of the tree. SIM has two forwarding modes: list mode and preset mode. In the list mode, the SIM sender always attaches the receivers' list to the packet. In the preset mode, which is the prevalent SIM mode, the SIM sender periodically attaches the receivers' list to the packet. Both SEM and SIM suffer from scalability issues due to the group-based state maintenance in branching routers.

Basically, *GXcast* (Generalized *Xcast*) [7] is intended to avoid packet fragmentation in *Xcast*. To cope with large set of receivers, *GXcast* regroups them into smaller sets and sends an *Xcast* packet to each set.

Clearly, their approach is applicable to other explicit multicast protocols as well with minor modifications.

3. *Bcast*

The main difference between *Bcast* and *ERM* is the location of tree encoding authority. *ERM* employs nearest *ERM*-capable router to encode the delivery tree and send multicast data packet down the tree. This node, which is called source *ERM* router, is responsible to send dummy heartbeat packets in absence of the multicast traffic. However, failure of a source *ERM* router stalls entire multicast session until selection of a new source *ERM* router, which is done after detection of failure by of all receivers. In *Bcast*, the sender host generates the tree code(s). This simple design choice cures session stall problem of *ERM*.

Bcast tree encoding scheme is more elaborated than *ERM*, which reduces the header size overhead. *ERM* encoding method assigns an ID (1 byte) to each receiver and Branching Point (BP), which determines the location of its parent in the tree code. In contrast, *Bcast* assigns a number to each BP, which resolves the position of the BP, number of its children and their positions. Also, *Bcast* avoid *ERM* packet encapsulation overhead by including the IP address of the multicast group in the *Bcast* header.

3.1 Tree encoding

Hereafter, we use the term "sub-tree" instead of "tree" since the sender might be directly attached to a branching router (as in figure 1). In this situation, it is beneficial to generate a separate code for each sub-tree rooted at the sender. We discuss *Bcast* encoding mechanism by means of an example. In figure 1, sender *S* is attached to two BPs namely *H1* and *H2*. Therefore, *S* generates two separate tree codes for each sub-tree. First, we discuss the generated code for *H2* sub-tree (figure 2a). For each BP, there is a number following it, which determines starting position of its children when added by the BP position. For example, value 2 after *H3* indicates that *H3* children start from position (1+2). Children of "sub-tree root" (*H2*) starts from position 1.

Each BP must calculate the starting position of children of the next BP in the code to determine its children. For example, after finding that children of the next BP (*H3*) start from position 3, *H2* could infer that it has two children namely *H3* and *H4*. *H3* and *H4* determine their children in the same way. The case for *H5* is slightly different. *H5* finds that there is no next BP in the code. Hence, its children are starting from position 7 toward the end of the code (DR4 and DR5).

In this example, *H1* sub-tree only consists of two receivers. Hence, *S* simply inserts them in the *Bcast* header as shown in fig. 2b.

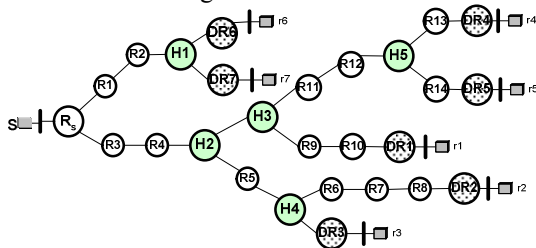


Figure 1. An example tree with seven receivers.

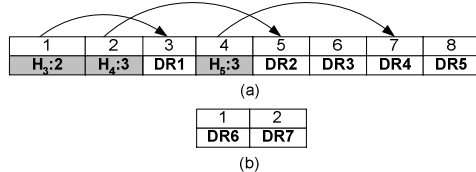


Figure 2. (a) The encoded sub-tree for *H2* (b) The encoded sub-tree for *H1*

There is some difficulty in representing numbers that are associated with each BP. These numbers must be represented in a way that one can easily distinguish them from unicast IP addresses around them. Since all IP addresses in *Bcast* header are unicast, they are beginning with patterns 0, 10 and 110 for address classes A, B and C respectively. Consequently, the required numbers in the code can be represented by 111*****. Since pattern 11100000 is meaningless, we can represent value between 1 and 32 by one byte. In rare cases, if there is a node with more than 32 branches, the coder (i.e. sender) can reduce the node branching factor by shifting sufficient number of its children to a nearby BP in the code.

3.2 Gathering tree information

The sender collects the tree information from receivers *Join* messages. Each new member sends a *Join* message with the Router Alert option to the sender. As stated in [16], routers that do not recognize this option shall ignore it and routers that recognize the option shall examine packets more closely to determine whether or not further processing is necessary. All *Bcast*-aware routers on the path will inspect the message and append their IP addresses to it. Other routers simply forward the message to its next-hop towards the sender. Using the Router Alert option, we guarantee the incremental deployment of *Bcast*. Since only *Bcast*-aware routers sign the *Join* message, a *Bcast*-unaware router cannot appear in the tree code. In the worst case, when all routers on the receiver's path do not support *Bcast* yet, the sender uses unicast delivery method for that receiver.

Having the path information from receivers to the source, the source can calculate reverse shortest path tree. Simulation result in [1] confirmed that using reverse path instead of direct path has a negligible impact on the tree cost in terms of the required unicast lookups and size of the tree code. The receivers must repeat their *Join* messages periodically in order to refresh their state in the sender enabling it to repair the multicast distribution tree against temporary route changes and BPs failures. If sender misses three consecutive *Join* messages from a receiver, it will remove the receiver from the tree code. The receiver can also immediately depart from the multicast session by sending *Leave* messages directly to the sender.

3.3 Data Distribution

In *Bcast*, a multicast data packet is always sent toward next BP in the tree (or receiver). A non-BP router forwards *Bcast* data packets same as unicast packets. When a BP receives a data packet, it decodes the tree code and finds the next BP(s) and/or receiver(s). Then it generates the required copies of the packet with proper destination IP addresses and forwards them.

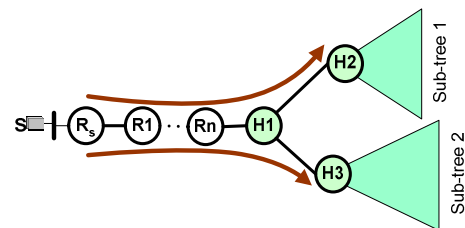


Figure 3. Bypassing of first BP in *Bcast*

3.4 Bypassing first branching router(s)

An increase in the number of receivers will definitely decrease the useful data space portion of the packets. In addition, some undesirable distributions of receivers could have a huge tree code. In these cases, the sender could ignore the first BP of the tree and sends data packets directly to the BP children breaking the original tree code into two or more sub-tree codes (figure 3). Consequently, the branching factor of the sender is virtually increased. This allows *Bcast* to support larger number of receivers and efficiently cope with detrimental (?) distribution of the receivers. Obviously, the overhead of the bypassing mechanism is the presence of duplicate packets on the link(s) between sender and the bypassed BP. One can define some criteria to deduce whether it is better to encode the complete tree or bypass the first BP. These criteria can entail size of the original tree code, size of the tree codes resulted from bypassing mechanism, source rate,

Maximum Transfer Unit (MTU), hop-count and bandwidth of the path between the sender and the first BP.

4. Comparison

4.1 Processing and header size overhead

To calculate the next-hop(s) of an *Xcast* data packet, each router has to perform as unicast table lookups as the number of the destination addresses that exists in the *Xcast* header. This processing overhead limits the number of supportable receivers to a very small number (order of 10 [17]). Obviously, *Xcast+* has the same processing problem as *Xcast*. Number of performed unicast lookups in *Xcast* and *Xcast+* is same as sending a separate unicast packet towards each receiver. *ERM* and *Bcast* insert more information in the header of data packets to reduce the number of required unicast table lookups. They pre-calculate locations of packet duplications in the network and add them to the constructed header. In these protocols, data packets are always sent to a next BP or a receiver. They perform as unicast lookups as the number of multicast tree branches minus one. *Linkcast* goes even further than them and strictly defines the tree by its link components. As a result, it does not require any type of table lookups for packet forwarding purposes. The only processing overhead of *Linkcast* is the header decoding, which is common between all explicit multicast protocols.

An obvious trade-off for these reductions in processing overhead is a noticeable increase in the header size of *ERM*, *Bcast* and *Linkcast*. Our results indicate that *Linkcast* header size is normally less than *Bcast* and *ERM* thanks to Chuang-Sirbu scaling law [12]. *ERM* and *Bcast* can handle header size rise by virtually increasing the branching factor of the coding entity using first BP bypassing mechanism.

4.2 Fault tolerant

Except the case for packet loss, an *Xcast* packet may not arrive at some of its final destinations if and only if those destinations are not reachable at all by unicast routing temporarily or permanently. However, routers failure has no effect on delivery of *Xcast* packets because they can use alternative routes same as unicast packets. For *Bcast* and *ERM* packets, when a BP fails, all the receivers in its sub-tree will loss prospect data packets until situation is fixed by *Bcast* (*ERM*) recovery mechanism. But, failure of non-branching routers has no effect on delivery of data packets. It is worth noting that a link failure has no effect on the packet delivery of all explicit multicast protocols except *Linkcast*. *Linkcast* behavior against

link failure is the same as conventional multicast protocols. In *Linkcast*, any type of link or router failure will partition the multicast distribution tree and disrupt multicast service of downstream members.

4.3 Other issues

Since explicit multicast uses one-to many model introduced in EXPRESS [15], all explicit multicast protocols can perform sender access control authenticating the *Join* messages of receivers in order to determine their access level. In addition, reliable multicast protocols will find explicit multicast beneficial to some extent, because the sender can easily address a subset of the receivers for retransmission purposes.

4.3.1 Gradual deployment. Although some inefficient mechanisms are provided for this problem in *Xcast* proposal, it seems that *Xcast* can not be gradually deployed in the Internet. The same is true for *Linkcast* and *Xcast+*. On the other hand, *Bcast* and *ERM* have native support for incremental deployment the same as other BP-based protocols [19] [13].

4.3.2 IPv6. The length of IPv6 addresses increases the header size for all explicit multicast protocols except *Linkcast* by a factor of 4. Since *Linkcast* uses link IDs to encode the multicast distribution tree, it naturally has no problem with IPv6. This fact combined with Chuang-Sirbu scaling law makes *Linkcast* a superior choice for supporting larger groups.

4.3.3 State maintenance. Although main benefit of explicit multicast is its stateless design, all proposals need to maintain some sort of state. In *Bcast* and *Linkcast*, the sender host must maintain paths information of all members. In *ERM*, source *ERM* router maintains paths information of the members. Meanwhile, *Xcast* and *Xcast+* use some sort of cache to alleviate the effect of unnecessary lookups problem.

4.3.4 Forward vs. reverse direction tree. *Xcast* packets are always forwarded through direct path between sender and receivers. Although receiver *Join* (*Trace*) messages record the reverse path, *Bcast* (*ERM*) data packets are forwarded along direct path between BPs. One can imagine *Bcast* (*ERM*) tree as a direct path tree which employs BPs of reverse path tree. *Linkcast* data packets are forwarded along a reverse direction tree constructed based on the reverse paths information.

4.3.5 Accounting and billing. Since *Bcast* (*ERM*) header contains the IP addresses of all BPs of multicast tree, all ISP on the path can easily distinguish whether a *Bcast* (*ERM*) packet is duplicated in their network.

They can also determine the number of generated copies and their generation places. This will facilitate the accounting and billing of *Bcast* (*ERM*) data packets.

5. Performance evaluation

We have implemented *Xcast+*, *ERM*, *Linkcast* and *Bcast* in “myns” packet level simulator which is publicly available [2]. We chose *Xcast+* for our comparisons since it is slightly more efficient than *Xcast*. The network topologies were generated based on Transit-Stub graph model, using GT-ITM topology generator [22]. We fixed the network size at 10100 nodes and performed the simulations with various group sizes ranging from 5 to 50. The average node degree of generated topologies was fixed approximately at 3.5. Each point in each graph is resulted from 10 simulation runs for 10 different topologies. The identities of the group members, i.e. sender and receivers, were randomly selected in each simulation run.

In [9], many different-size real-trees are generated executing trace-route command from receivers’ sites toward a single and fixed sender. These real trees are available at [10]. We used the largest sets, which have 1950 receivers. Then, we randomly chose smaller subset of the receivers and reconstructed the multicast tree and measured its performance. We repeated the process for various group sizes between 5 and 50. Also, for each group size, we repeated the process 100 times.

We have evaluated the performance of the selected methods using following metrics:

Header Size: For each method, we calculated the full header size required to support given number of receivers. We normalized all header size plots to corresponding *Xcast+* value. In its simplest form, *Xcast+* header size is $12 + (\text{Number of receivers}) * 4$ bytes.

Number of required unicast lookups: For a given number of receivers, each method (except *Linkcast*) has to perform exact amount of unicast lookups to deliver a multicast packet to all receivers. We define a special metric named Forwarding Gain (*F_G*) which is the ratio between the numbers of required unicast lookups in *Xcast+* to *Bcast* or *ERM*.

Stress: The stress of each physical link is defined as the number of distinct copies of the same packet that pass through that link in data distribution phase.

5.1 Header size

Figures 4a and 4b show the header size overhead of the evaluated methods for generated topologies and the Internet respectively. We experimented with two different encoding methods of *ERM* as discussed in [3]. The *ERM-1* uses plain unicast tunneling between *ERM*

router pairs, while *ERM-2* uses minimal encapsulation within IP header. We augmented *Xcast+* with route information in order to enable *Xcast+* to get benefit from the first-hop branching possibility (named as *Xcast+-M* in presented figures). As one can see, the header size for *Xcast+-M* is slightly better than *Xcast+* in our simulations. However, source branching was fairly rare event both in the generated random networks and real trees. The *ERM* headers are largest among all methods. They are even larger than *Bcast* header size because of *Bcast* superior tree encoding method. *Linkcast* header size is less than *Bcast* and *ERM*. In addition, *Linkcast* overhead decreases when the group size increases as predicted by Chuang-Sirbu scaling law. Also, simulation results are nearly in full conformance with the Internet results.

5.2 Processing overhead

Figure 5a shows the comparison between the required unicast lookups in *Bcast* (*ERM*) and *Xcast+*. As stated before, *Linkcast* processing overhead is the smallest among all methods and it does not require any sort of table lookups. In this figure, “*Xcast+* Internet” and “*Bcast* Internet” stand for *Xcast+* and *Bcast+* simulation results in the Internet. As figure suggests, *Bcast* (*ERM*) reduces the number of required unicast lookups efficiently. The number of performed unicast lookups in the Internet experiments is more due to a higher average path length in the Internet. In the generated topology, *F_G* was 1.58 for 5 receivers and grew linearly until it reached 2.66 for 50 receivers. In the Internet, *F_G* was 1.19 and 2.98 for 5 and 50 receivers respectively (see figure 5c).

5.3 First BP bypassing

In the experiments presented in this section, *Bcast* sender always bypassed the first BP of the tree. The result for the header size is shown in figure 4c. The header size reduction pattern of *ERM* was the same as *Bcast*. Therefore, it was omitted from this figure for the sake of clarity. Using bypassing capability, *Bcast* is able to achieve the same or less header size in comparison to *Xcast+*. Figure 5c shows the *F_G* with and without bypassing. The figure confirms that the first BP bypassing has negligible effect on the number of unicast lookups: 9.33% and 4.1% rise on average for generated networks and the Internet respectively.

Figure 6 shows the length and stress of the path between the sender and first BP of the tree. The average values for length and stress of the path in the generated topologies were 2.57 and 3.46 respectively. The value 2.57 means that there are less than 3 routers between

the sender and first BP. The average branching factor of the first BP is actually larger than 3.46. The reason is that when the sender bypasses the first BP, it calculates the size of largest induced sub-tree. Then, it uses

this value as an indicator to merge smaller sub-trees. The number presented here is the number of merged sub-trees. The average length and stress of the path in the Internet tests were 2.1 and 2.16 respectively.

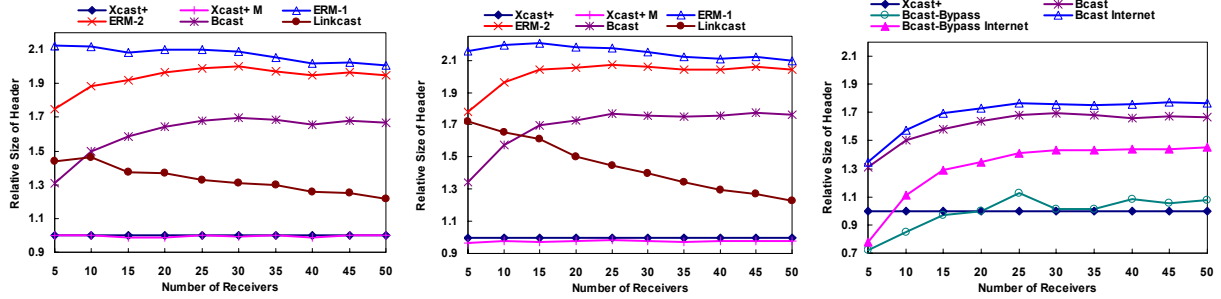


Figure 4. The header size comparison when group size changes from 5 to 50 (a) in transit-stub random topology model (b) in the Internet. (c) Effect of bypassing feature on the size of Bcast header.

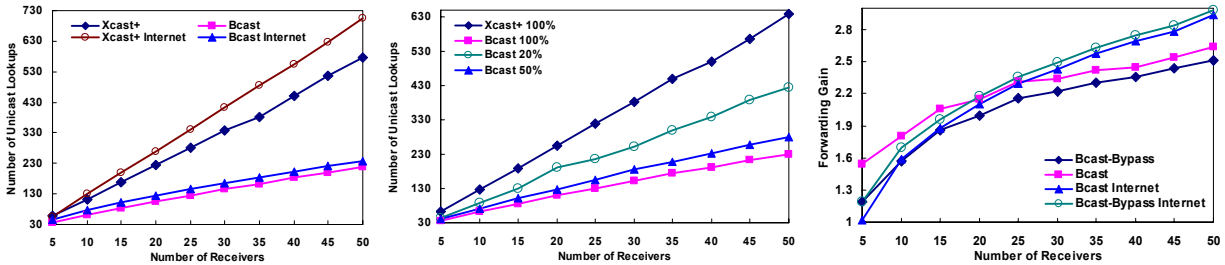


Figure 5. The number of performed unicast lookups (a) in generated topologies and the Internet (b) for various deployment ratios of Bcast. (c) The forwarding gain of Bcast.

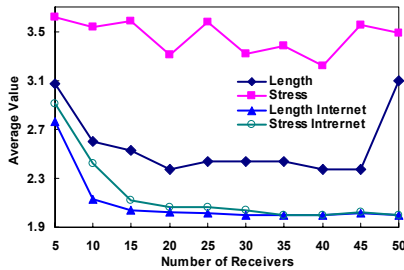


Figure 6. Length and stress of path between the sender and first BP.

5.4 Incremental deployment

We examined incremental deploy-ability feature of *Bcast* (and hence *ERM*) with different mix of *Bcast*-aware and *Bcast*-unaware routers. In all experiments, we chose the receivers merely among *Bcast*-aware routers. We first assumed that only 20% of routes upgraded with *Bcast* code. Then, we incremented the ratio of *Bcast* deployment to 50% and performed the simulation again. The result for required unicast lookups is presented in figure 5b. As expected, the F_G increases with deployment ratio. The F_G for *Bcast*-20% is halfway between *Xcast+* and *Bcast*-100%. The average F_G values for *Bcast*-20%, *Bcast*-50% and *Bcast*-100% are 1.45, 2.00 and 2.38 respectively.

The results for header size overhead are shown in figures 7a and 7b for 20% and 50% deployment ratio respectively. The *Bcast*-Bypass results presented in these figures are different from those in the previous subsection in a sense that we bypassed the first BP in the overlay topology. As one can see, the difference between header size of *Xcast+* and *Bcast* becomes smaller for lower deployment ratio.

In figure 7c, we plotted a cumulative distribution of the stress value for data delivery to 50 receivers using *Bcast* with 20% and 50% deployment ratio and multi-unicast approach. Clearly, full *Bcast* stress value is 1 for all physical links same as IP multicast. The three methods use roughly the same number of links (near 230) on average to deliver multicast data packets. Though most of the links have stress equal 1, multi-unicast has very high load on the nodes near sender. The maximum stress values for multi-unicast, *Bcast*-20% and *Bcast*-50% are 50, 45 and 30 respectively. Although *Bcast*-20% removes efficiently the heavy tail of multi-unicast distribution, the slope of line for stress values between 20 and 40 shows that low deployment ratio imposes high load on many network links. But in the case of *Bcast*-50%, the stress values greater than 10 are fairly negligible. For example 97.41% of links have stress less than 4 and 99.04% of them have stress val-

ues less than 7. For *Bcast*-20%, 97.35% and 99.04% of

links have stress less than 10 and 30 respectively.

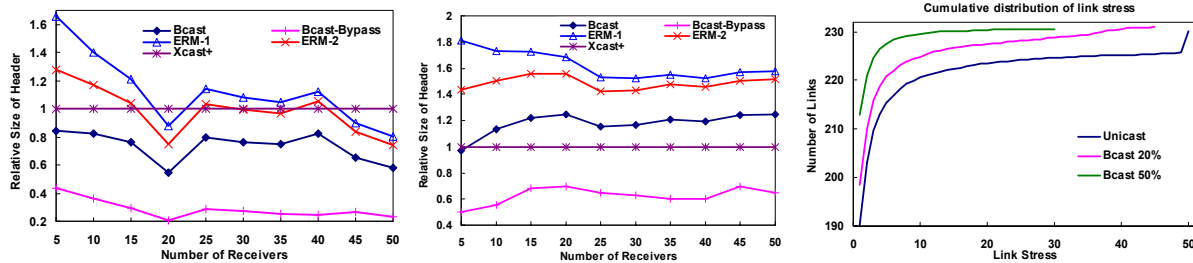


Figure 7. The header size comparisons for (a) 20% deployment ratio (b) 50% deployment ratio. (c) Cumulative distribution of link stress values for a group of size 50.

6. Conclusion

We studied explicit multicast routing protocol through detailed discussion and simulation experiments in the Internet and generated random networks. Our findings imply that the basic approach, i.e. *Xcast*, suffers from a high processing overhead. Although other proposals reduces the processing overhead of *Xcast*, they increase the header size overhead. Each method has some unique features. *Xcast* and *Xcast+* data packets exactly endure same conditions and go through same paths as unicast data packets. *Linkcast* has the minimum processing overhead among all explicit multicast approaches. Furthermore, *Linkcast* header size obeys Chuang-Sirbu scaling law [12], which implies that multicast cost is more reasonable for larger groups. In addition, *Linkcast* extension for IPv6 is straightforward. *ERM* and *Bcast* can be gradually deployed in the Internet. Their other noticeable features are low processing overhead, the possibility to decrease header size overhead with a reasonable cost and easiness of billing and accounting. We also showed that the header size overhead of *Bcast* is better than *ERM* due to its subtle tree encoding scheme.

7. References

[1] M. Bag-Mohammadi, S. Samadian-Barzoki, N. Yazdani, "Linkcast: Fast and Scalable Multicast Routing Protocol", Proc. of Networking 2004, Athens, Greece, 2004.
 [2] S. Banerjee, myns Simulator. <http://www.cs.umd.edu/~suman/research/myns/index.html>
 [3] J. Bion, D. Farinacci, M. Shand, A. Tweedly, "Explicit Route Multicast (ERM)", draft-shand-ERM-00.txt, June 2000.
 [4] R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms, O. Paridaens, "Explicit Multicast (Xcast) Basic Specification", draft-ooms-Xcast-basic-spec-07.txt, 2005.
 [5] R. Boivie, N. Feldman, C. Metz "Small Group Multicast: A New Solution for Multicasting on the Internet", Internet Computing, Vol. 4, No. 3, May/June 2000.
 [6] A. Boudani, B. Cousin, "SEM: A New Small Group Multicast Routing Protocol", Proc. of IEEE ICT 2003, Tahiti, Feb. 2003.

[7] A. Boudani, A. Guitton, B. Cousin. "GXcast: Generalized Explicit Multicast Routing Protocols", Proc. Of ISCC 2004, Alexandria, Egypt, 2004.
 [8] B. Cain, S. Deering, I. Kouvelas, B. Fenner, A. Thyagarajan, "Internet Group Management Protocol, Version 3" RFC 3376, October 2002.
 [9] R. Chalmers, K. Almeroth, "On the Topology of Multicast Trees", IEEE/ACM Trans. on Networking, Vol. 11, No. 1, pp.153-165, Feb. 2003.
 [10] R. Chalmers, K. Almeroth, <http://www.nmsl.cs.ucsb.edu/mwalk/>
 [11] Y. Chu, S. Rao, H. Zhang, "A Case for End System Multicast" ACM SIGMETRICS 2000, CA, June 2000.
 [12] J. Chuang and M. Sirbu, "Pricing Multicast Communication: A Cost-Based Approach", Proc. of INET'98, Geneva, Switzerland, July 1998.
 [13] L. H. M. K. Costa, S. Fdida, O. C. M. B. Duarte, "Hop By Hop Multicast Routing Protocol", SIGCOMM'01, San Diego, USA, August 2001.
 [14] S. Deering, et al., "The PIM architecture for wide-area multicast routing", IEEE/ACM Trans. on Networking, Vol.4, No.2, April 1996.
 [15] H.W. Holbrook, D.R. Cheriton, "IP multicast channels: EXPRESS support for large-scale single-source applications", ACM SIGCOMM'99, Sept. 1999.
 [16] D. Katz, "IP Router Alert Option", RFC 2113, February 1997.
 [17] O. Paridaens, D. Ooms, "Security Framework for Explicit Multicast", draft-paridaens-Xcast-sec-framework-01.txt, November 2000.
 [18] M.K. Shin, Y.J Kim, K.S Park, S.H Kim, "Explicit Multicast Extension (Xcast+) Supporting Receiver Initiated Join", draft-shin-Xcast-receiver-join-02.txt, October, 2002.
 [19] I. Stoica, T. S. Eugene Ng, H. Zhang, "REUNITE: A Recursive Unicast Approach to Multicast", IEEE INFOCOM'2000, Mar. 2000.
 [20] V. Visoottiviseth, H. Kido, Y. Kadobayashi, S. Yamaguchi, "Sender-Initiated Multicast Forwarding Scheme", In Proceedings of IEEE ICT2003, Tahiti, Feb 2003.
 [21] D. Waitzman, C. Partridge, S. Deering, "Distance Vector Multicast Routing Protocol", RFC 1075, Nov. 1988.
 [22] E. W. Zegura, K. Calvert, S. Bhattacharjee., "How to model an Internetwork.", Proc. of IEEE Infocom '96, San Francisco, CA