Journal of Information Technology and Engineering
Vol. 1 No. 2 (December, 2016)

Compatibility and Analysis of Explicit Multicast for Network Games oriented Applications

Ali Boudani[1]

1Thomson R&D France, av Belle-Fontaine, 35700 Rennes
E-mail: ali.boudani@thomson.net, ali boudani@hotmail.com

Received: 03rd June 2016 Revised: 14th August 2016 Accepted: 01st November 2016

Abstract: *Network game traffic generates a significant share of today's Internet traffic. Network games can be seen as a multicast group where players are the members. Each player in the group is considered as a source and other players as destinations. Recently several multicast mechanisms were proposed that scale better with the number of multicast groups than traditional multicast does. These proposals are known as small group multicast (SGM) or explicit multicast (Xcast). Explicit multicast protocols, such as the Xcast protocol, encode the list of group members in the Xcast header of every packet. If the number of members in a group increases, routers may need to frag-ment an Xcast packet. Fragmented packets may not be identified as Xcast packets by routers. In this paper, we show that the Xcast protocol does not support the IP frag-mentation and we show also that avoiding fragmenta-tion induces hard-coded limits inside the protocol itself in terms of group size. First, we describe the Xcast proto-col, the Xcast+ protocol (which is an extension of Xcast) and we compare these two protocols with traditional multicast protocols.We propose then a generalized version of the Xcast protocol, called the GXcast protocol, intended to permit the Xcast packets fragmentation and to sup-port the increasing in the number of members in a multicast group. We analyze and evaluate with simulations the impact on the GXcast protocol in terms of scalability and efficiency. In our evaluation, we considered the network game as an application case of the Xcast protocol. Finally, we conclude that the GXcast protocol is a feasible and promising protocol and very adequate to network applications.*

Keywords: *Author Guide, Article, Camera-Ready Format, Paper Specifications, Paper Submission*

1. INTRODUCTION

Interactive, multi-player network games are becoming more common. The amount of Internet traffic generated by computer games can be expected to increase fast, especially when the new wave of players enters the Internet with the next generation game consoles that support Internet connections. The observed game traffic still follows the transmit cycle described in [1]: the server sends game state information to each client where packets are read and processed. Clients synchronize the server game state with their local game state, process player commands and return update packets with the player's movement and status information. Since slower client machines require more processing time for rendering, their packet rate may be lower. Both update and server information packet's size is usually very small since they only contain movement and status information. A high market potential, increasing usage as well as sharp real time requirements make this kind of traffic interesting for Internet ervice providers and manufacturers.

Important problems need to be studied with Internet gaming:

- The effect of increasing the number of users.
- The effect of reduced QoS (i.e. packet loss, delay) on the performance of games.
- Comparison of different connection types.

There exist no publicly available, standardized protocols for exchanging network gaming data. This means that different game software manufacturers utilize either their own or licensed protocols for network gaming.

Network games can be seen as a multicast group where players and the server are the members. Each player in the group is considered as a source and other players as destinations. This is more true for MMORPG than FPS games.

Multicast, the ability to efficiently send data to a group of destinations, has become increasingly important with the emergence of network-based applications like video-conferencing, distributed interactive simulation and software upgrading. A multicast routing protocol should be simple to implement, scalable, robust, use minimal network overhead, consume minimal memory resources, and inter-operate with other multicast routing protocols.

Most of proposed multicast protocols like DVMRP[2] and MOSPF ([3, 4]) perform well if group members are densely packed. However, the fact that DVMRP periodically floods the network and the fact that MOSPF sends group membership information over the links make these protocols not efficient in cases where group members are sparsely distributed among regions and the bandwidth is not plentiful.

To address these issues, the Protocol Independent Multicast (PIM) routing protocols are being developed by the Inter-DomainMulticast Routing (IDMR), working group of the IETF. PIM contains two protocols: PIM-Dense Mode (PIM-DM) [5] which is more efficient with applications where group members are densely distributed, and PIM-SparseMode (PIM-SM) [6] which performs better with applications where group members are sparsely distributed. Although these two protocols share similar control messages, they are essentially proposed for two different kinds of applications.

Today's multicast protocols [7] can be used to minimize bandwidth consumption, but it suffers from a scalability problem with the number of concurrently active multicast groups because it requires a router to keep a forwarding state for every multicast tree passing through it and the number of forwarding states grows with the number of groups.

There seem to be two kinds of multicast that are important: a broadcast-like multicast that sends data to a large number of destinations and a narrow cast multicast that sends data to a fairly small group. An example of the first kind of multicast is the audio and video multicasting of a presentation to all employees in a corporate intranet. An example of the second kind of multicast is a video conference involving three or four parties [7]. Thus, a one-size-fits-all protocol will be unable to meet the requirements of all applications [8]. Providing for many groups of small conferences (a small number of widely dispersed people) with global topological scope scales badly given the current multicast model.

Recently several multicast mechanisms were proposed that scale better with the number of multicast groups than traditional multicast does. These proposals are known as small group multicast (SGM) [10] or explicit multicast (Xcast) [11]. Explicit multicast protocols, such as the Xcast protocol, encode the list of group members in the Xcast header of every packet. Xcast assumes that there is no packet fragmentation. However, if fragmentation occurs (e.g. if the group size or the data is too large) the fragmented packets will not be identified as Xcast packets by routers. In this paper we propose a generalized Xcast protocol to support the group size increasing and to overcome the fragmentation problem.

In section 2, we describe the Xcast protocol, the Xcast+ [12] protocol (which is an extension of Xcast), we present the Xcast packets fragmentation problem and we compare these two protocols with traditional multicast protocols. In section 3, we describe the GX-cast protocol and we study some parameters. In section 4, we evaluate GXcast in terms of the number of generated packets, the duplicated packets overhead, the global processing time and the average supplement delay. Finally, we conclude in section 5 that the GXcast protocol is feasible and promising.

2. THE XCAST AND THE XCAST+ PROTOCOLS

To solve the problems of traditional multicast protocols, Boivie *et al.* proposed the Explicit Multicast protocol (Xcast). In this section, we describe the Xcast he Xcast+ protocol (which is an extension of Xcast) and we compare them with traditional multicast protocols.

2.1 The Xcast Protocol

The Xcast protocol [11] is a newly proposed multicast protocol to support a very large number of small multicast groups. To send data to a given group, the source first explicitly encodes the list of destinations in the Xcast header of the packet. Then, the source parses the header, partitions the destinations based on each next unicast hop and forwards a packet with an appropriate header to each of the next hops. Each router along the path to destinations repeats the same processing on receiving an Xcast packet. If a router detects that there is only one destination in the destination list of a packet, the packet is converted to unicast. The algorithm realizing the conversion of an Xcast packet to a unicast packet is called Xcast-to-Unicast (X2U). This packet is then forwarded in unicast along the remainder of the route.

Example: consider the network represented on figure 1 and the group G formed from the source S and the six destinations D1, D2, D3, D4, D5 and D6. The nation list (D1, D2, D3, D4, D5, D6). S proceeds the packet and remarks that R1 is the next unicast hop for all the destinations. Consequently, S sends the Xcast packet to R1. R1 receives the packet and proceeds it similarly. It forwards the packet to the R2 router which also forwards it to R3.While proceeding the packet, the R3 router remarks that R4 is the next unicast hop for the two destinations D1 and D2 and that R5 is the next unicast hop for the remaining destinations D3, D4, D5 and D6. R3 sends to R4 an Xcast packet containing the destination list (D1, D2) and to R5 an Xcast packet containing the destination list (D3, D4, D5, D6). Upon receiving the Xcast packet, the R4 router detects that D1 and D2 are two separated hosts. R4 then generates two unicast packets using the X2U algorithm and sends them to D1 and D2. Upon receiving the packet, D1 extracts the

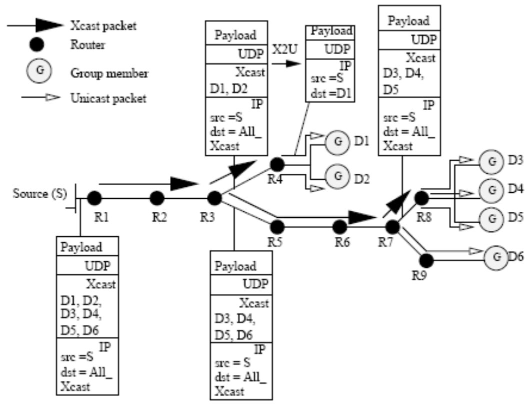

Figure 1: The Forwarding of Data in theXcast Protocol.

data from the packet. The process is similar for routers R5 to R9 and for the five remaining destinations.

2.2 The Xcast + Protocol

Xcast+ is an extension of Xcast for a more efficient delivery of multicast packets [12]. Every source or destination is associated to a Designated Router (DR). Instead of encoding in the Xcast packet header the set of group members, Xcast+ encodes the set of their DRs.When a new member wants to join the group G of source S, it sends an IGMP-join message [13] to its DR. The DR will send a join-request message to the source S. The DR of the source intercepts this message and analyzes it in order to keep track of all concerned DR addresses.When the source S wants to send a message to the group G, it sends a multicast packet.This packet is received by its DR and converted to an Xcast packet using the Multicast-to-Xcast algorithm (M2X). The packet is then forwarded as in Xcast to all the DRs associated to the group G, since the destination list in the Xcast header contains the DR addresses instead of the member addresses. Then, each DR converts the Xcast packet to a multicast packet using the Xcast-to-Multicast protocol (X2M) and sends it in its subnetworks.

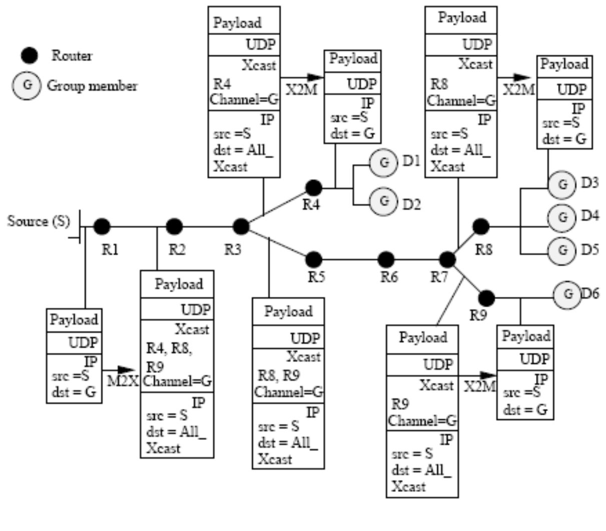

Figure 2: The Forwarding of Data in the Xcast+ Protocol

Example: consider the same network represented on figure 2 and the group formed from the source S and the five destinations D1, D2, D3, D4 and D5. Suppose that D6 is a new member which wants to join the group G. D6 initiates the join of the group by sending an IGMP message for the group (S,G). The DR of D6, R9, receives the join request and sends a registration request message toward S. When the DR of S, R1, receives the registration request message, it sends back to R9 a registration reply message and does not forward the registration request message to S. Thus, R1 is able to know dynamically the set of DRs of the receivers and can fill the destination list of Xcast packets on receiving multicast packet from S. The figure 2 shows where M2X and X2M algorithms are used. Between the DR of the source and the DRs of the destinations, packets are forwarded as normal Xcast packets.

2.3 The IP Fragmentation Mechanism

Due to physical reasons, every link can transfer only a limited volume of information in each packet. The Internet protocol (IP) [14] contains a mechanism called fragmentation which makes this limitation transparent.

The fragmentation mechanism allows a packet to be cut into fragments in order to be suitably transferred on a link. Suppose that a router receives a packet. After having decided on which link this packet should be forwarded, the router checks the maximum capacity of this link which is the Maximum Transmission Unit (MTU). If the packet is too large and unless it is explicitly forbidden, the router cuts it out in order to respect the following constraints:

- each resulting fragment is an autonomous IP packet, with a valid IP header,
- each resulting fragment has a size less than or equal to the MTU,
- the data is distributed between the fragments.

The algorithm used to fragment IPv4 packets is explained in [14]. The IPv6 protocol also has a fragmentation mechanism, described in [15]. Note that one goal of IPv6 is to avoid the fragmentation. This will be discussed later.

2.4 Xcast Packet Fragmentation

Let us consider the Xcast packet fragmentation in a router. Since the Xcast packet header may be large, two cases can be considered as depicted on figure 3: either the whole Xcast packet header is short enough to be kept in the first fragment, or the Xcast header has to be cut out.

In both cases, the second fragment is not a valid Xcast packet since it has no Xcast header. Thus, these packets cannot be forwarded (Xcasted) to the receivers and the data they contain is lost. Moreover, in the second case the first fragment contains only a subset of receivers and no data. The first fragment may however be forwarded up to the mentioned receivers, inducing meaningful traffic.

These problems show that the fragmentation of an Xcast packet should be forbidden. This can be done in IPv4 by setting the appropriate flag (Don't Fragment, DF) in the IP header. If a packet having the DF flag set and has to be fragmented by a router, it is simply dropped. In order to reach the receivers, the source has to limit the size of its packets to 576 bytes which is the minimum MTU guaranteed by IPv4 on any link. This size limits the number of receivers in an Xcast group to 134. In IPv6, since the minimum MTU is 1280 bytes and since IPv6 addresses are stored using 16 bytes, the limit in the size of the Xcast group is 76. Having these limits hardly coded in protocols is restrictive. What we propose is a simple mechanism to cancel these limitations in the size of Xcast groups. The performance and the scalability of our proposition will be analyzed.

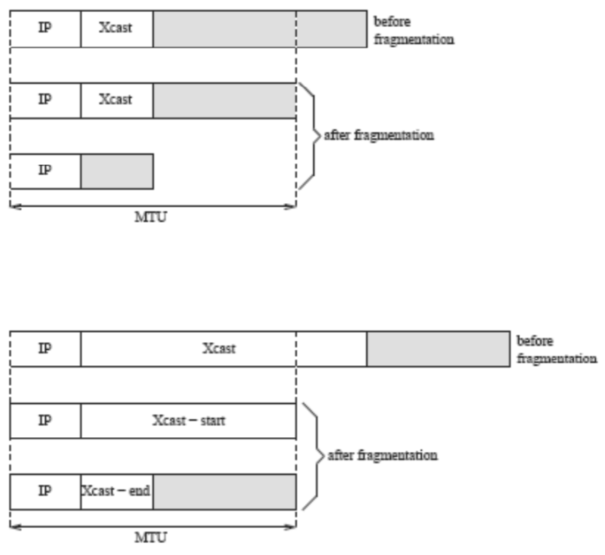

Figure 3: Fragmentation of Two Different Xcast Packets.

2.5 Comparison between Explicit Multicast Protocols and Traditional Multicast Protocols

Traditional multicast protocols and explicit multicast protocols are two different approaches designed to handle multicast groups. We will try to emphasize the main advantages of each method, compared to the other one.

2.5.1. Drawbacks of explicit multicast protocols: In addition to the important Xcast packet fragmentation problem, other related drawbacks also exist.

Limitedpayload Packet size is limited as a result of network MTU. In explicit multicast protocols, the larger the list of destinations is, the lower the payload is. As a consequence, more packets should be generated to transmit a given amount of data.

Complex header processing In explicit multicast protocols, each destination in the header needs a routing table lookup. A packet with n destinations in the list of destinations will require n + 1 unicast routing table lookups1. Additionally, a different header has to be constructed per next hop. However, it can be noticed that since such protocols are typically designed for sparse sessions, there will be a limited number of branching routers compared to non-branching routers. The construction of different headers only occurs in branching points. The header processing can moreover be reduced to a simple operation: the modification of a bitmap.

2.5.2. Advantages of explicit multicast protocols: Explicit multicast protocols make easier some aspects of the routing of multicast packets. It has many advantages over traditional multicast protocols.

Routing state and signalisation messages management in explicit multicast protocols, routers do not have to maintain a state per group. Indeed, there is no multicast forwarding table since only unicast tables are used. This makes the Xcast protocol very scalable in terms of the number of groups that can be supported simultaneously since the routers in the network do not need to disseminate information for the groups.

Automatic reaction to unicast reroutes and simplified traffic engineering explicit multicast protocols react immediately to unicast route changes. Traditional multicast protocols need to exchange information with unicast protocols in order to have an adequate reaction. This is achieved on a polling basis in many implementations, yielding a slower reaction to e.g. link failures. This delay may also depend on the number of concerned groups. In addition, there is no need for a specific multicast traffic engineering tool since packets follow traffic engineered unicast paths.

Easier security and accounting in explicit multicast protocols, the source has a complete knowledge about members (or about DR members). It will be able to drop dynamically some members and a border router can be able to determine approximately how many times a packet will be duplicated in its domain (especially when link-state protocols like OSPF [16] are used in the domain).

Other advantages can be mentioned:

- No multicast address allocation is needed except eventually in the DR of the source in the case of the Xcast+ protocol.
- Shortest path is always used even in an asymmetric network.

3. THE GXCAST PROTOCOL

As explained in the previous section, the Xcast protocol can not support large groups due to its incompatibility with the IP fragmentation mechanism. In this section, we propose a generalized Xcast routing protocol, the GXcast protocol, which is designed basically to avoid the fragmentation. Moreover, the GXcast protocol can be parameterized in order to improve the Xcast behavior.

3.1 The GXcast Protocol

The GXcast protocol is a simple generalized version of the Xcast protocol: instead of sending a message to the n destinations, the source limits the number of destinations in a packet to n_M. Thus, the list of n destinations is divided into sub-lists of at most n_M destinations. Each sub-list corresponds to a destination list for an Xcast packet. Several packets may have to be sent in order to deliver data to all the n destinations.

n_M is the parameter of the GXcast protocol and it impacts the protocol performance in terms of several criteria. The choice of nM is justified in section 3.2. GXcast packets are similar to Xcast packets: they have the same header and are treated in the same way by intermediate routers, DR destinations and user destinations. The only difference between the Xcast protocol and the GXcast protocol appears in the source or in the DR of the source. The Xcast protocol and the GXcast protocol can therefore interperate easily.

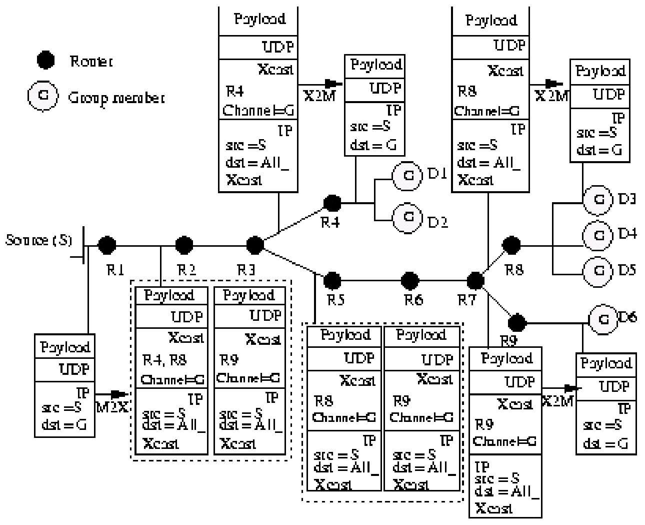

Figure 4: The Forwarding of Data in the GXcast Protocol

Example: consider the same network represented on figure 4 and the group formed from the source S and the six members D_1, D_2, D_3, D_4, D_5 and D_6. As in the Xcast+ protocol, the DR of the source keeps track of only the three DRs representing the subnetworks that contain all the destinations: R_4, R_8 and R_9. For this example[2], n_M is fixed to 2. The source sends a multicast packet to its DR, R_1. R_1 translates it from a multicast packet to an Xcast packet using the M2X algorithm. R_1 notices that there are three destinations in the list for the next hop. Since n_M equals to 2, this list is divided into two sub-lists: one contains the first two destinations R_4 and R_8 and the second contains the last destination, R_9[3]. Each generated packet is treated as a normal Xcast packet as shown on figure 4.

3.2 Study of the GXcast Parameter

The behavior of the GXcast protocol greatly depends on the value of the nM parameter. Indeed, as we will see in this subsection, there is a number of criteria that are directly influenced by the chosen value. In the following, we will denote by *MTU* the value of the *MTU* which depends on the IP version used, by E the size of the IP header plus the size of the Xcast header (typically 16 bytes) and by IP the size of an IP address. *n* will represent the number of destinations in the group and d the volume in bytes of data to transfer.

3.2.1. Simple behavior: As we have seen in subsection 2.3, since a packet has to contain at least one byte of data, the maximum number of destinations n_{max} allowed in an Xcast packet is defined as:

$$n_{\max} = \left\lfloor \frac{MTU - E - 1}{IP} \right\rfloor$$

The values n_{max}=134 and n_{max}=76 correspond respectively to the IPv4 and to the IPv6 specifications. The simplest behavior GXcast can adopt is to fix the n_M value to the n_{max} value. However, this is not efficient for groups having a lot of members (typically more than n_{max}). For example, suppose that IPv6 is used and suppose that *n*=70 members have joined the group. Each message can contain only 104 bytes of information[4]. In order to send a volume of 10000 bytes, 97 packets are needed. However, less packets would be the result of a better choice of n_{max}. Choosing n_{max} equals to 38 allows 616 bytes per packet, which results into the emission of only 34 packets to reach the *n* destinations, which is approximately three times less.

3.2.2. The number of members influenced by a fault: If a drop occurred on a GXcast packet, every member having its address in the member list will be concerned by the drop. To reduce the number of destinations concerned by such errors, small values of n_{max} should be chosen.

3.2.3. Number of generated packets: Considering a group of *n* destinations and a volume of d bytes to transmit to these members, the number of packets $p(n, d, n_M)$ sent by the GXcast protocol with a parameter of n_M is defined as:

$$p(n,d,n_M) = \left\lceil \frac{n}{n_M} \right\rceil \left\lceil \frac{d}{MTU - E - IP * n_M} \right\rceil$$

$$p(n,d,n_M) = \left\lceil \frac{n}{n_M} \right\rceil \left\lceil \frac{d}{MTU - E - IP.\min(n,n_M)} \right\rceil$$

Recall that in the GXcast protocol, the list of *n* destinations is cut out into sub-lists of size at most n_M. The left part of the expression of p represents the number of sub-lists that will be generated by the GXcast protocol. The right part of the expression of *p* represents the number of packets needed to transmit the d bytes of data. In order to study the behavior of *p* in terms of n_M, we will consider two cases: $n < n_M$ and $n_M \le n$. In the first case, we have:

$$p(n,d,n_M) = \left\lceil \frac{d}{MTU - E - n.IP} \right\rceil$$

This expression of *p* does not depend on n_{max}. The GXcast protocol behaves in this case in the same way than the Xcast protocol. In the second case where , we propose to study the behavior of the function which is an approximation of the *p* function and is defined by:

$$\overline{p}(n,d,n_M) = \frac{n}{n_M} \frac{d}{MTU - E - IP.n_M}$$

Since we have

$$\frac{\delta \overline{p}(n,d,n_M)}{\delta n_M} = -n.d \frac{MTU - E - 2.IP.n_M}{MTU.n_M - E.n_M - IP.n^2{}_M}$$

the $\overline{p}$ function admits a minimum value for:

Since this optimal value does not depend on n and on d and since it is very simple to calculate it and provides good results in terms of the number of generated packets, we propose it as a default value for the GXcast protocol.

4. GXCAST EVALUATION AND SIMULATION

In this section, the GXcast protocol is evaluated in terms of scalability (generated packets cost overhead) and efficiency (delay from the source to destinations and global processing time).

4.1 The GXcast Cost Overhead Rate

The number of generated packets by GXcast depends of the three parameters n, d and n_M. The choice of the value of nM has been justified in the paragraph 3.2. Let us note that a traditional multicast routing protocol sends $p(n,d) = \left\lceil \frac{d}{MTU - E} \right\rceil$ packets to deliver d data bytes to a group of n members[5]. We define $\delta_{pGXcast/multicast}$ as the cost overhead rate generated by the GXcast protocol compared to a traditional multicast routing protocol and we distinguish the two cases: $n_M \le n$ and $n > n_M$.

$$\delta_{pGXcast/multicast} = \frac{p(n,d,n_M = \frac{n_{\max}}{2})}{p(n,d)}$$

$$= \frac{\left\lceil \frac{2*n}{n_{\max}} \right\rceil \left\lceil \frac{2*d}{2*(MTU-E) - n_{\max}*IP} \right\rceil}{\left\lceil \frac{d}{MTU-E} \right\rceil}$$

$$\approx \frac{\left\lceil \frac{2*n}{n_{\max}} \right\rceil \left\lceil \frac{2*d}{MTU-E} \right\rceil}{\left\lceil \frac{d}{MTU-E} \right\rceil} \le 4*\left\lceil \frac{n}{n_{\max}} \right\rceil$$

For : $n \le n_M$:

$$\delta_{pGXcast/multicast} = \frac{p(n,d,n_M)}{p(n,d)}$$

$$= \frac{\left\lceil \frac{n}{n_M} \right\rceil \left\lceil \frac{d}{MTU - E - n*IP} \right\rceil}{\left\lceil \frac{d}{MTU-E} \right\rceil}$$

$$= \frac{\left\lceil \frac{d}{MTU - E - n*IP} \right\rceil}{\left\lceil \frac{d}{MTU-E} \right\rceil}$$

Figure 5 presents the number of packets generated by the GXcast protocol compared to the number of packets generated by a traditional multicast routing protocol for $n \le n_M$. We notice that the value of cost overhead rate does not exceed the value of 2 and that in the majority of cases this value is equal to 1, which implies that the cost overhead generated by the GXcast protocol is small if the number of receivers is lower than n_M.

For $n > n_M$:

$$\delta_{pGXcast/multicast} = \frac{p(n,d,\frac{n_M}{2})}{p(n,d)}$$

$$= \frac{\left\lceil \frac{2*n}{n_{\max}} \right\rceil \left\lceil \frac{2*d}{2*(MTU-E) - n_{\max}*IP} \right\rceil}{\left\lceil \frac{d}{MTU-E} \right\rceil}$$

$$\approx \frac{\left\lceil \frac{2*n}{n_{\max}} \right\rceil \left\lceil \frac{2*d}{MTU-E} \right\rceil}{\left\lceil \frac{d}{MTU-E} \right\rceil} \le 4*\left\lceil \frac{n}{n_{\max}} \right\rceil$$

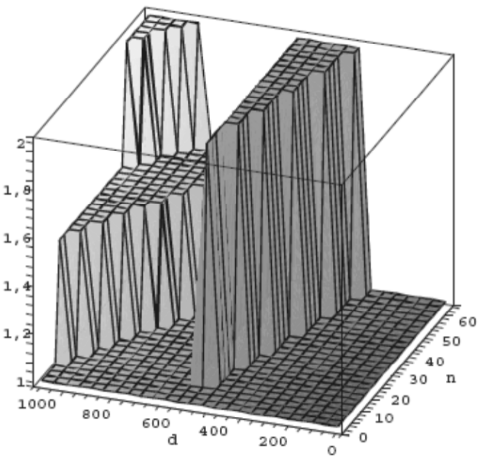

Figure 5: The Cost Overhead Rate of the Protocol GXcast Compared to a Traditional Multicast Routing Protocol in Terms of n and d for $n \le n_M$.

The cost overhead rate of the protocol GXcast compared to a traditional multicast routing protocol in terms of n and d for $n \le n_M$.

Figure 6 presents the number of packets generated by the GXcast protocol compared to the number of packets generated by a traditional multicast routing protocol for $n > n_M$.

We notice that the cost overhead rate is small if the payload size is lower than 250 bytes. Moreover, we notice that this rate is almost linear according to n: for a group of 150 members, it will be necessary to send approximately 5 times more packets with the GXcast protocol than with a traditional multicast routing protocol. For a group of 300 members, it will be necessary to send 10 times more packets.

To transmit to respectively $n = 150$ and $n = 300$ receivers a message of $d = 1000$ bytes, a protocol based on unicast messages would have required 300 and 600 packets, which is respectively 150 and 300 times more than a traditional multicast routing protocol[6]. It should be noted that a traditional multicast routing protocol requires the presence of multicast routing states in all the routers on the multicast trees of the various group. Also, control packets are permanently sent between routers on a tree to maintain these routing states. In Gxcast, no need for multicast states in routers neither for signaling messages between these routers.

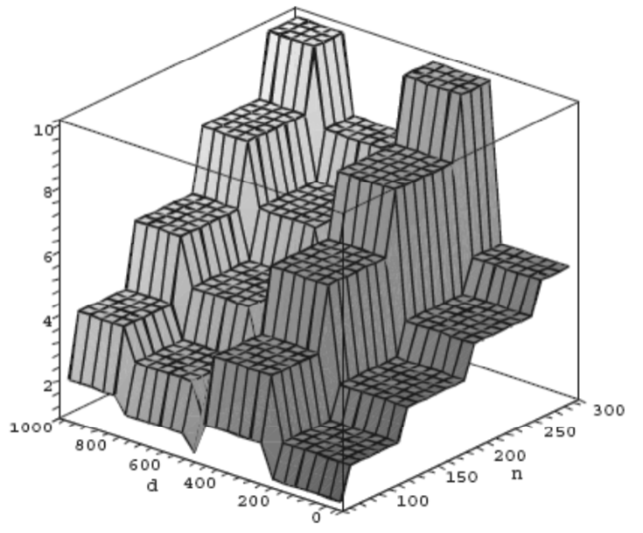

Figure 6: The Cost Overhead Rate of the Protocol GXcast Compared to a Traditional Multicast Routing Protocol in Terms of *n* and d for $n > n_M$.

The GXcast protocol and unicast to send *d* data bytes to n members of a group, the number of packets $p_{Unicast}(n,d)$ generated by the source by using the unicast transmission mode is defined by the following formula:

$$p_{Unicast}(n,d) = n * p(n,d) = n * \left\lceil \frac{d}{MTU - E} \right\rceil$$

Figure 7 presents the number of packets generated by a unicast routing protocol compared to the number of packets generated by the GXcast protocol. We noticed that the value of the rate of cost overhead is very high what implies that the cost overhead generated by a unicast protocol compared to GXcast becomes very significant if the number of receivers increases.

The GXcast protocol and the Xcast protocol we consider two cases only since the Xcast protocol is unable to manage a group having more n_{max} members:

$n \le n_M = \frac{n_{\max}}{2}$ and $\frac{n_{\max}}{2} < n \le n_{\max}$

To send *d* bytes of data to *n* members of a group, the number of packets $p_{Xcast}(n,d)$ generated by the Xcast protocol is defined by the following formula in both cases:

$$p_{Xcast}(n,d) = \left\lceil \frac{d}{MTU - E - IP * n} \right\rceil$$
$$\approx \left\lceil \frac{d}{IP * (n_{\max} - n)} \right\rceil$$

For $n \le \frac{n_{\max}}{2}$, the two protocols GXcast and Xcast generate the same number of packets, therefore the cost overhead rate of the GXcast protocol compared to the Xcast protocol is equal to 1.

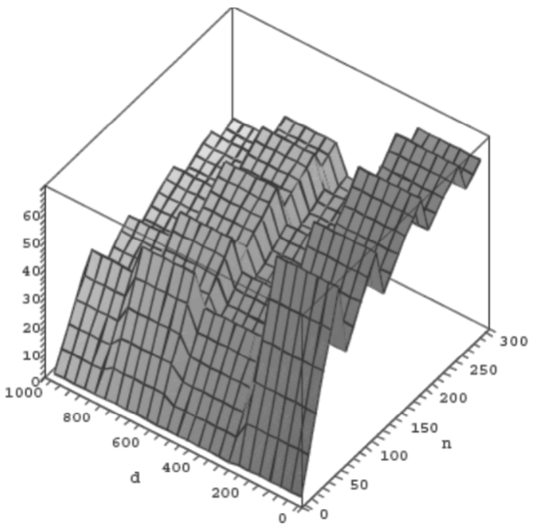

Figure 7: The Cost Overhead Rate of a Unicast Routing Protocol Compared to the GXcast Protocol in Terms of *n* and *d*.

For $\frac{n_{\max}}{2} < n \le n_{\max}$, we have:

$$p(n,d,\frac{n_{\max}}{2}) = \left\lceil \frac{n}{\frac{n_{\max}}{2}} \right\rceil \left\lceil \frac{d}{MTU - E - IP * \frac{n_{\max}}{2}} \right\rceil$$
$$\approx 2 * \left\lceil \frac{2 * d}{IP * n_{\max}} \right\rceil$$

We deduce that the value of

$$\delta_{pGXcast/Xcast} = \frac{p(n,d,\frac{n_{\max}}{2})}{p_{Xcast}(n,d)} \approx \frac{2 * \left\lceil \frac{2 * d}{IP * n_{\max}} \right\rceil}{\left\lceil \frac{d}{IP * (n_{\max} - n)} \right\rceil}$$

$$= \frac{\left\lceil \frac{2*n}{n_{\max}} \right\rceil \left\lceil \frac{2*d}{2*(MTU-E)-n_{\max}*IP} \right\rceil}{\left\lceil \frac{d}{MTU-E} \right\rceil}$$

$$\approx \frac{\left\lceil \frac{2*n}{n_{\max}} \right\rceil \left\lceil \frac{2*d}{MTU-E} \right\rceil}{\left\lceil \frac{d}{MTU-E} \right\rceil} \leq 4* \left\lceil \frac{n}{n_{\max}} \right\rceil$$

is always ≤ 2.

Figure 8 presents the number of packets generated by the GXcast protocol to the number of packets generated by the Xcast protocol. It should be noticed that the value of the cost overhead rate never exceeds 2 and that in the majority of cases this value is largely smaller than 1[7] which implies that $\delta_{pGXcast/Xcast}$ can according to the size d takes values ≤ 1 and thus GXcast generates less packets than the Xcast protocol.

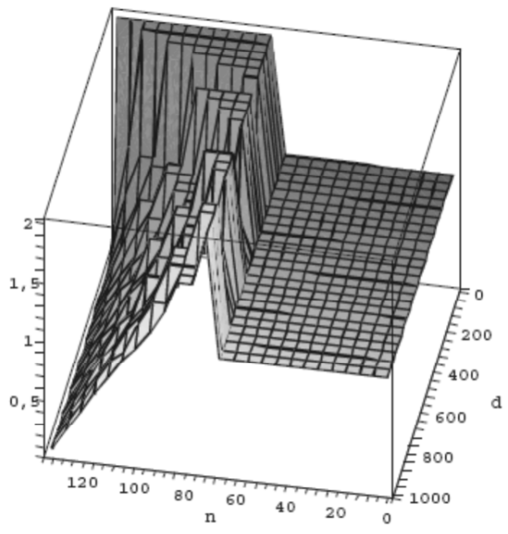

Figure 8: The Cost Overhead Rate of the GXcast Protocol Compared to theXcast Protocol in Terms of *n* and *d*.

4.2 Network Games Kind Applications Parameters

The GXcast protocol can be interesting, useful and promising for applications like network games and distributed interactive simulations. Indeed, the observed game traffic always follows the following transmit cycle: the server sends the game state information to each client (player) where packets are read and processed. The clients synchronize the server game with their local game state, process players commands and return update packets with the player's movement and status information. Both update and server information are usually of very small size since they only contain movement and status information.

Thus, the study of [18] shows that typically the mean packet size of server generated traffic is approximately 127 bytes with a coefficient of variation of 0.73. Moreover, around 99% of all packets are smaller than 250 bytes and no packet is larger than 1500 bytes. Most of server packets with a 1000 bytes size are assigned to gameplay interruptions (for example due to an end of turn or a change of scenario in which cases more information has to be transferred to the players). The client traffic is characterized by an almost constant size of packet. The mean packet size of the clients is of 82 bytes with a coefficient of variation of 0.12. Moreover, 99% of all packets range between 60 and 110 bytes.

Another study [19] on networks games shows that the mean packet size of server generated traffic is approximately 130 bytes while that of the client is about 40 bytes. The mean packet size of packets received and sent by the server is about 80 bytes. This study showed that over one week period, 16000 clients (average of 95 players per hour) established a connection with the server (they took part in the play) and 8000 were refused (since the server reached its limit of connections).

Same as network games, in applications like distributed interactive simulation (DIS) [20], informations on a virtual environment are exchanged between various machines in a distributed system. That makes possible to simulate the behavior of the objects in this environment. The objects are capable of physical interactions between them and can detect the other objects by the visual one or other means (infra-red, etc.).The DIS real time flow is made of packets of size of 2000 bits (250 bytes) and normally transmitted by using the UDP transport protocol [20].

4.3 The Simulation Scenario

We used NS [21] to develop our simulator of SGM protocols. This simulator can be used by researchers who want to test various types of protocols of explicit multicast routing. It is the only one to our knowledge which exists in the field.

We used the Abilene network [22] as a test network and we chose 5 as average value of the number of edge routers by Abilene node. With each of the 5 edge routers 5 possible destinations are connected[8]. Our network thus contains 341 nodes, connected by bidirectional links of 10 Gb/s with a delay time that varies between 2*ms* and 12*ms* in the network core and by bidirectional links of 155 Mb/*s* with a delay time of 0.3*ms* to 0.5*ms* in the remainder of the topology (between the Abilene nodes and the edge nodes as between the edge nodes and the destinations) (cf. figure 9).

We consider only one multicast group having a source belonging to a sub-network and n receivers (in the other sub-networks). The receivers join and leave randomly the group. The maximum number of receivers per GXcast packet is 70 (near to $\frac{n_{\max}}{2}$)[9]. We chose our simulation parameters by considering applications like DIS and networks games. Table 1 recapitulates the parameters used in the simulation.

Figure 10 represents the estimated number of packets (according to the quantity of data (*d*) to be transmitted) to be generated by the source while figure 11 represents the effective number of packets generated by the source.

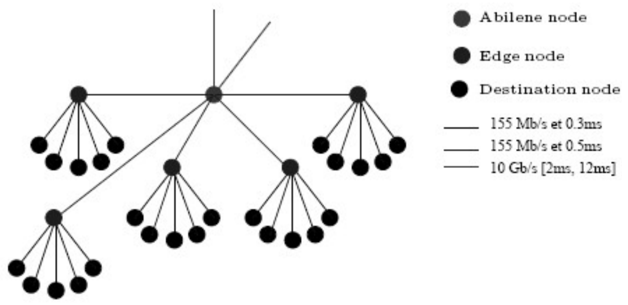

Figure 9: An Abilene Network Nodewith Edge and Destination Nodes.

Table 1
The Simulation Parameters for the GXcast Protocol

d	80, 130, 250, 1000	Data size to be transmitted to each receiver
n	80, 90, 100, 110, 120,130	Number of destinations per group
n_M	70	Maximum number of destinations in a GXcast packet

The horizontal axis represents the number of packets generated by the source and the vertical axis represents the number of receivers (*n* that varies from 80 to 130) in a group.

We notice that the effective number is equal to the number estimated except the case where *d* = 1000 bytes. This is due to the fact that formula (1) supposes that the packets generated by the source for n receivers in GXcast contain nM receivers. Let us take the case where $n_M < n \le 2 * n_M$: the source generates a packet of n_M receivers and another of n-n_M receivers and thus this last can contain more data than the first.

To measure the cost of generated packets, we define , the estimator of the average charge of the network by link and , the average load factor of the network by link by the following equations:

$$\gamma = \frac{\sum_{i=1}^{N_l} N_{paq}(l_i)}{t_{sim} * N_l}$$

$$\sigma = \frac{\sum_{i=1}^{N_l} \left(\frac{\sum_{j=`}^{N_{paq}(l_i)} (L_{pj})}{D_{l_i}} \right)}{t_{sim} * N_l}$$

where N_l is the number of links, $N_{paq}(l_i)$ is the number of packets on the link l_i, L_{pj} is the size of a packet *pj*, D_{li} is the throughput of the link l_i and t_{sim} is the duration of simulation.

We make this distinction between γ and σ because even if the number of packets generated by the source with the GXcast protocol is higher than that the number of packets generated with the Xcast protocol, the size of a GXcast packet is smaller or equal than that of an Xcast packet.

We make an assumption that the throughput of all the source or destination links are identical and all the network core links have identical throughput. We have chosen to analyse these characteristics (i.e. number of packets and quantity of data) on 3 critical points of the network: on the source link, on the destination link, on the network core link. We will see how GXcast behaves on these critical points. Thus, we consider only the number of circulating packets and the quantity of data transmitted in each link during each simulation. Figures 12 and 13 respectively represent the number of packets and the quantity of data transmitted on all the links of the network of the source. Figures 14 and 15 represent respectively the number of packets and the quantity of data transmitted on all the links of the network core. Figures 16 and 17 respectively represent the number of packets and the quantity of data transmitted on all the links of the networks of the receivers.

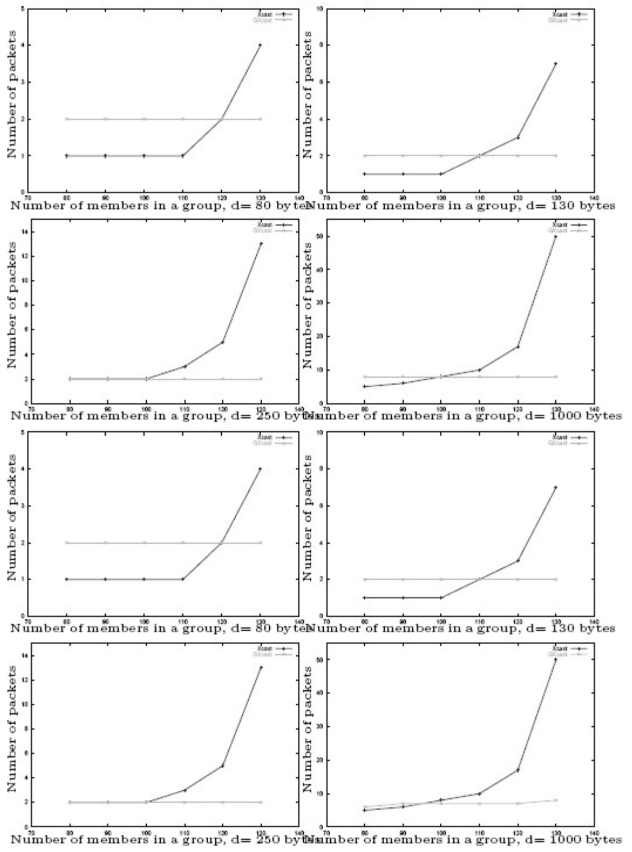

Figure 11: The Number of Packets Generated by the Source with the Protocols GXcast and Xcast.

GXcast and Xcast behavior at the source network we notice initially that the Xcast protocol is unable to manage groups having more than 100 members and especially for a payload greater than 250 bytes. For this type of groups, we notice that not only the GXcast protocol generates less packets than the Xcast protocol but also that the data volume transmitted through the source network is much smaller than that of Xcast. For a payload of 80 to 130 bytes, and although the GXcast protocol generates twice more packets in the source network, the cost overhead generated by the GXcast protocol in term of transmitted volume is weak. Indeed, this ratio decreases since the size of data transmitted with the GXcast protocol is lower than twice the size of data transmitted with the Xcast protocol.

GXcast and Xcast behavior at the core network we notice that in spite of the fact that the number of packets transmitted by the core network with the GXcast protocol is higher than the number of packets generated with the Xcast protocol, the volume transmitted through this network core is smaller than the data volume generated with the protocol Xcast and thus the GXcast protocol behaves better than the Xcast protocol in the core network. This constitutes an advantage for the GXcast protocol since with the GXcast protocol we reduce the cost of the tree inside the core network. For the groups of less than 100 members and for a payload of less than 130 bytes, the GXcast protocol has a behavior similar to the Xcast protocol and however sometimes it generates a light overhead.

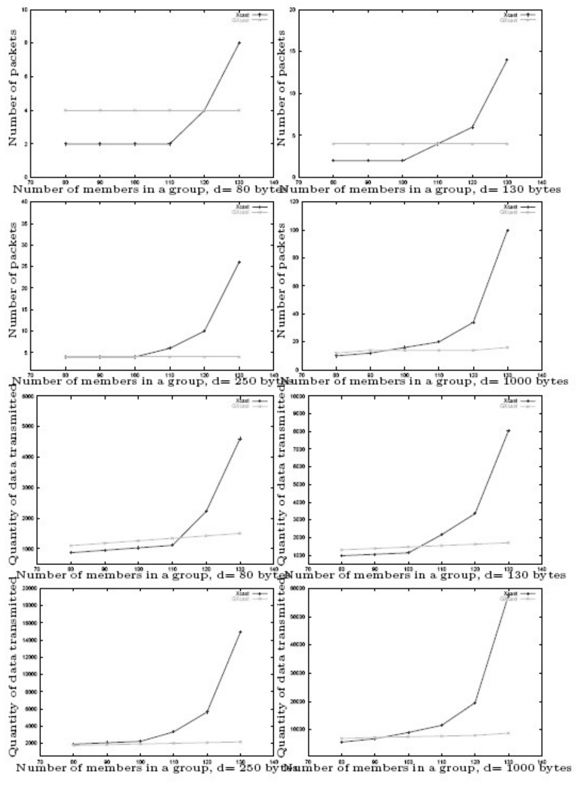

Figure 13: The Transmitted Volume in the Source Network with the Protocols GXcast and Xcast

GXcast andXcast in the networks of the receivers finally, we notice a clear reduction of the number of packets and volume transmitted over the networks of the receivers with the GXcast protocol compared to the Xcast protocol. This is due to the fact that each receiver will receive an Xcast packet of fixed size while the GXcast packet intended for the same receiver have a smaller size, which reduces the total data volume transmitted through the networks of the receivers.

Currently, many network games send their data using *unicast* packet. After we noticed that the cost overhead comes from the size of the packets and that the number of packets cannot be considered as the only factor of cost overhead, we made a comparison with a unicast routing protocol. Indeed, with a unicast routing protocol, the source sends a separated packet to each receiver but the size of this packet is smaller than the size of a GXcast packet as soon as the number of members of the group exceeds a few units. It should be noted that the number of generated GXcast packets is smaller than that in unicast. Since the GXcast protocol and the transmission in unicast mode can support more than 130 members limited by the Xcast protocol, we use the 3 following values: 90, 140 and 190 receivers in a group. Indeed, these values were selected by supposing that the number of receivers in a network game will double: from 95 per hour to 190 per hour. We take the same simulation parameters than previously. Thus, figures 18 and 19 respectively represent the number of packets and the quantity of data transmitted on all the links of the source network. Figures 20 and 21 represent respectively the number of packets and the quantity of data transmitted on all the links of the core network and figures 22 et 23 respectively represent the number of packets and the quantity of data transmitted on all the links of the receiver’s networks. The horizontal axis represents the number of packets or the quantity of transmitted data and the vertical axis represent the number of receivers belonging to a group. The axes with the label “Unicast” represent the values obtained by using the transmission in unicast mode.

GXcast and Unicast in the source network we notice, first of all, that the number of generated packets is much higher with the Unicast protocol than with the GXcast protocol. This cost overhead is confirmed with the volume transmitted through the network of the source. With the selected parameters we notice that by using GXcast we reduce 20 times the cost of the links of the source network. This constitutes a major advantage since the links near the sources are not always able to support a significant load and that is why we have an interest to transmit in multicast mode instead of unicast mode.

GXcast and Unicast in the core network the advantage of the use of the GXcast protocol instead of a protocol using the transmission in unicast mode appears once again in the core network. Indeed, the number of packets which crosses the network core is about 15 times more significant with the

Unicast protocol than with the GXcast protocol. Transmitted volume is about 7 times less and can reach 12 times less with the GXcast protocol than that with the Unicast protocol according to the data payload.

GXcast and Unicast in the networks of the receivers finally, in the networks of the receivers, the number of packets transmitted with the GXcast protocol is less than with the Unicast protocol. This is an expected result since GXcast uses only one packet for nM receivers while Unicast uses a packet for each receiver. But this high number of packets is not translated into transmitted volume, since the size of a unicast packet is smaller than the size of a GXcast packet. We deduce that the Unicast protocol has more advantages on the GXcast protocol in the networks of the receivers.

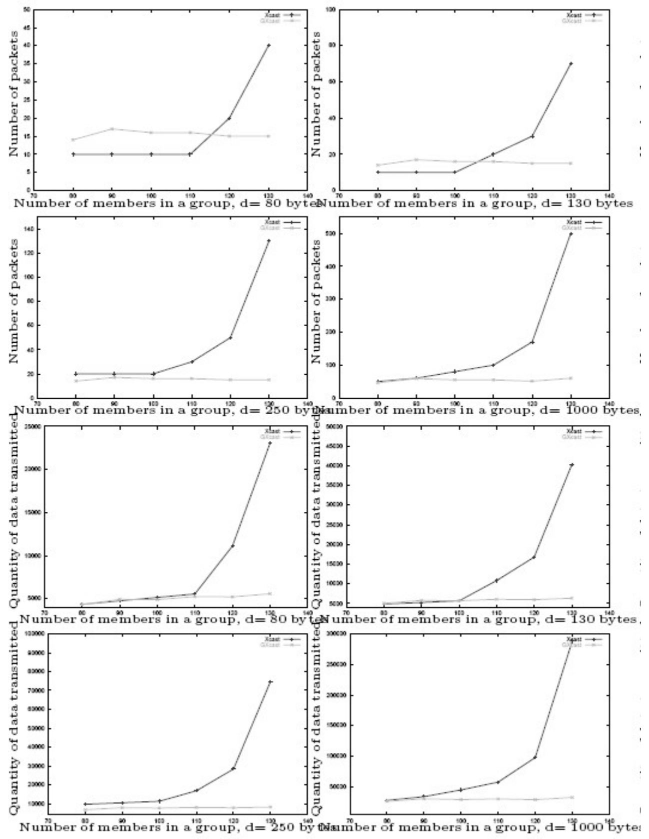

Figure 15: The Transmitted Volume in the Core Network with the Protocols GXcast and Xcast.

4.4 The Study of the Delay of the GXcast Protocol

Most popular network games sell millions of copies. Most multi-player games support network play over a LAN or over the Internet. When playing a game over the Internet, most players will log on to the network from home using a dial-up PPP connection via a modem. The high latencies that are common on the Internet (typically 50-150*ms* roundtrip delay) as well as the even higher latencies that modem connections exhibit (typically 150-400*ms* roundtrip delay) results in a large user base that is very concerned with network delay. Game players refer to these delays as “lag”, due to the deleterious visual impact that it has on their games. Video applications (such as Internet telephony or video-conferencing) require roundtrip delays of less than about 300*ms*. However, very few individuals using telephony can tell the difference between 50*ms* and 150*ms* of roundtrip delay. Game players have found that the difference between 50*ms* and 150*ms* of delay can determine who wins or loses a game [23].

The delay is the time passed between the sending of a packet by the source and its reception by the receiver. To illustrate the impact of the header processing cost of a GXcast packet, we briefly discuss the various delays added to a packet while it is passing from a node to another in a network. The delay contains the header processing time of a packet within a node, propagation delay along the way (time necessary to send a packet on a link), delay transmission (time necessary to send (to inject) a packet to a link) and queue (time that the packet passes in the queue before it can be sent) induced by the setting in queue of the packets in the intermediate systems (*cf*. figure 24).

Table 2 shows a simple calculation of these delays for links of 200 km and 1 Gb/s, and packets of size 1250 bytes. In the majority of cases, the principal delays are the

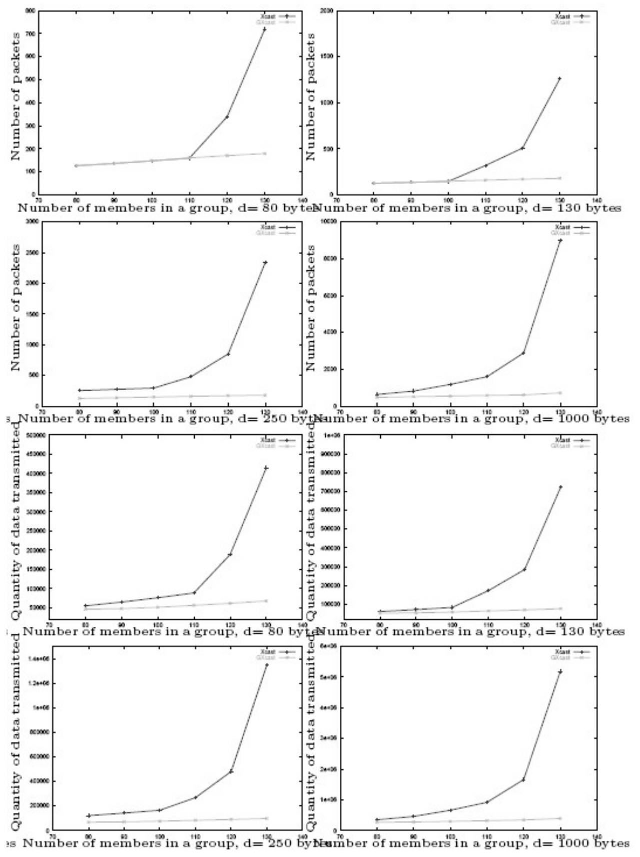

Figure 17: The Transmitted Volume in the Destinations Network with the Protocols GXcast and Xcast.

propagation and queuing delays and are thus considered in simulations and measurements. The delay of transmission is generally small for fast links and small packets and is not thus considered. Traditionally (the column “Simple routing” of Table 2), the processing delay is also negligible. However, the processing of a packet can take considerable values when ever the modification of the payload is necessary (as in IPsec, where one needs approximately 100 instructions by bytes of payload) [24]. Consequently, the processing can contribute not less than 50% of the total processing delay of a packet (the column “Complex modification of the payload” of Table 2).

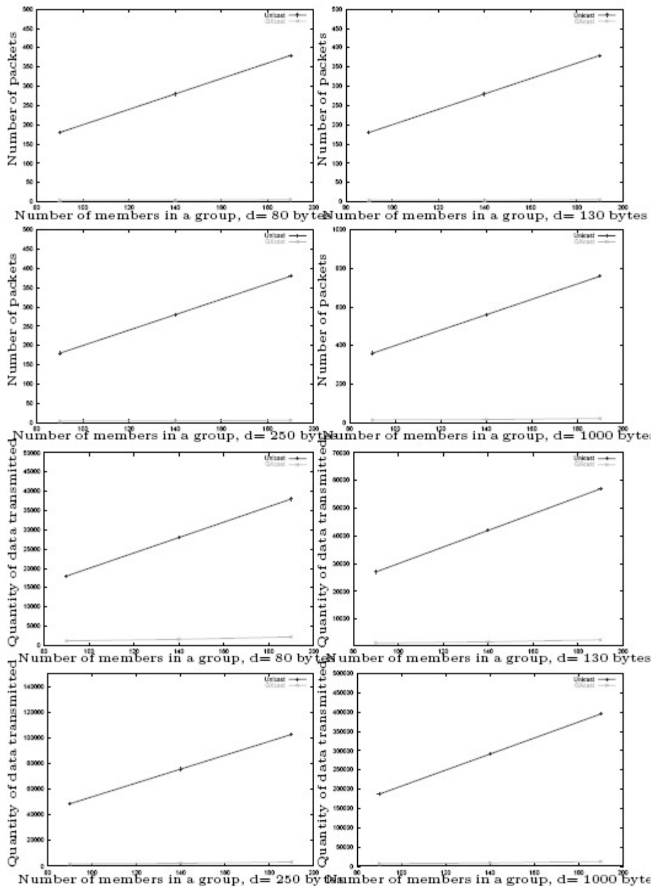

Figure 19: The Transmitted Volume in the Source Network with the Protocols GXcast and Unicast.

Table 2
The Various Delays for a 1Gb/s and 200 km Link, a Packet of 10kb Size and a 100MIPS Processor

Delay	*Simple Packet Forwarding*	*Complex Payload modifications*
Processing delay	10 µs	1000 µs
Propagation delay	1000 µs	1000 µs
Transmission delay	10 µs	10 µs
Queuing delay	0.. ∞	0.. ∞

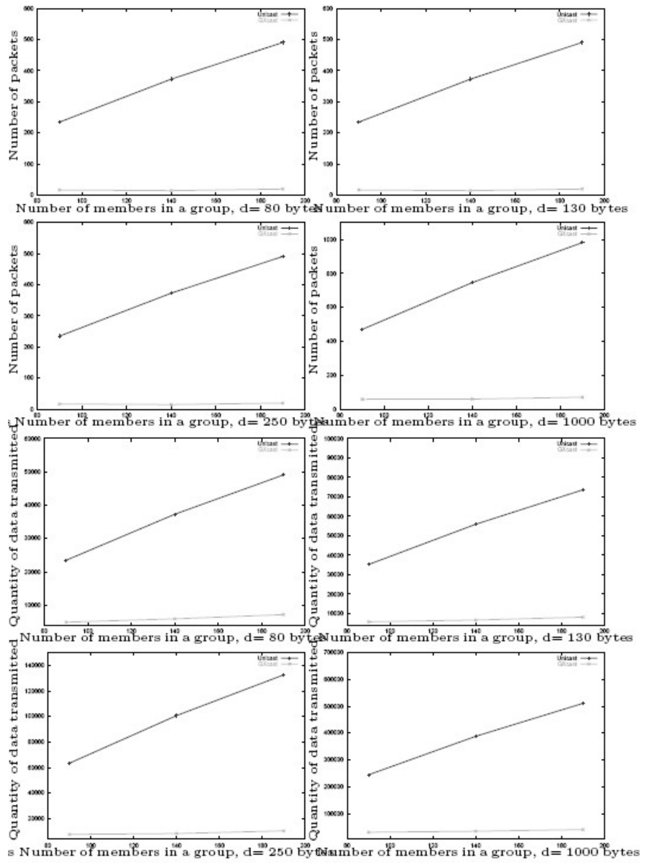

Figure 21: The Transmitted Volume in the Core Network with the Protocols GXcast and Unicast

4.4.1. Global processing time: We define the protocol global processing time as the sum of packet header processing times for every packet needed to send a fixed amount of data. The global processing time for a GXcast packet having n_M destinations is approximately $t_{nM} = \tau_1 + \tau_2 n_M$, where τ_1 is the IP and GXcast header processing time and τ_2 is the processing time for an entry in the list of destinations (lookup in routing table, creation of packets per outgoing interface, etc.). We have then:

$$t_G(n_M) = \left\lceil \frac{n}{n_M} \right\rceil t_{nM}$$

$$\approx \left\lceil \frac{n}{n_M} \right\rceil * (n_M + 1) * \tau_1 \approx n * \left(1 + \frac{1}{n_M}\right) * \tau_1$$

The $t_G(n_M)$ function is strictly decreasing and admits a minimum for $n_M = n_{max}$. Meanwhile, choosing $n_M = n_{max}$ is not realistic as shown in section 3.2.1 and 3.2 since this value does not take into account the quantity of data effectively transported. Meanwhile, choosing a small value for $n_M = n_{max}$ greatly increases the global processing time as a result of the high number of generated packets and the global processing time. The default value of $n_M = n_{max}, \dfrac{n_{\max}}{2}$, leads to a global processing time which is very close to the optimal one and is therefore a good compromise.

However, in our study on the delay, t_{nM} represents the processing time (in the processor of the node) for a GXcast packet while the value of $t_G(n_M)$ is included in an indirect way in measurement of the latency in the queue of the node. It is clear that $t_{nM} = t_n$ if $n \le n_M$ and $t_{nM} < t_n$ if $n > n_M$.

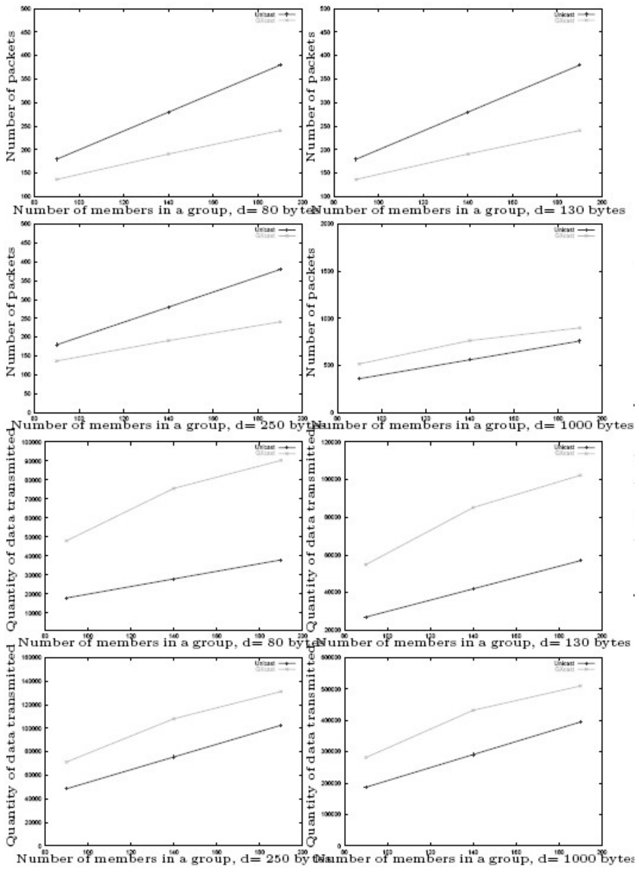

Figure 23: The Transmitted Volume in the Destinations Network with the Protocols GXcast and Unicast.

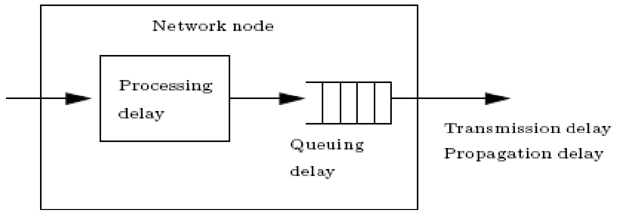

Figure 24: The Various Delays in a Node of the Network.

4.4.2. The additional delay compared to the protocols Xcast and Unicast: The total packet routing delay is the time a packet needs to reach its destination starting from the source. It greatly depends on the number of packets in the routers queues and the packet header processing time of each one of them. The difference in delay perceived by the end-user (i.e. the receiver) plays a significant role in the choice of the protocols.

Let us denote by T the multicast tree (in our case it is the set of paths followed by GXcast packets) from a source S to all destinations. The list of destinations will be denoted by L. Given a network represented by a weighted directed graph $G = (V, A, \delta)$ where V denotes the set of nodes, A the set of arcs and δ the delay-link function[10] : $\delta : A \to \mathbb{R}^+$.

Let $P_T(S, v)$ denotes the unique path from the source S to the destination $v \in L$[11], in the tree T, such that:

$$\delta_{T,v} = \sum_{l \in P_T(S,v)} \delta(l), \text{ for all } v \in L,$$

where $\delta_{T,v}$ is the delay for a destination v in the tree T.

We define the average global delay for a protocol P by:

$$\delta_g(P) = \frac{\sum_{i=1}^{n} \delta_{T,v_i}}{n}, v_i \in L.$$

We define the average additional delay $\delta_+(P)$ for a protocol P compared to the GXcast protocol by the following quation:

$$\delta_+(P) = \delta_g(P) - \delta_g(\text{GXcast}).$$

This average additional delay measures the delay overhead compared to the GXcast protocol.

In order to measure the average,minimum and maximum additional delays introduced on the level of the receiver by the useful load variation, the GXcast protocol was implemented under the network simulator (NS) with the same simulation parameters described previously. Thus, figures 25 and 26 respectively represent the additional delay (average, minimum and maximum) of the protocol Xcast compared to the GXcast protocol and that of the Unicast protocol compared to the GXcast protocol.

The additional delay of theXcast protocol compared to the GXcast protocol we notice first that the average, minimum and maximum additional delays are positive in the majority of cases. That is an advantage of the GXcast protocol on the Xcast protocol. We also notice that the curve of delay grows quickly with the growth of the number of members in a group. We notice that whenever GXcast is less powerful than Xcast, the additional delay of GXcast compared to Xcast does not exceed 10¼s while in the majority of the other cases Xcast largely exceeds these values.

The additional delay of the Unicast protocol compared to the GXcast protocol We notice that the GXcast protocol is faster than the Unicast protocol everywhere.

4.4.3. Using Path MTU instead of minimum MTU: In all this paper, we defined MTU as the minimum MTU guaranteed by IP. However, the value of the Path MTU (PMTU [25]) can also be used since we don't make any assumptions on the stability of the MTU value in our study. The PMTU is the minimum value of the MTU on the links of a path. It can be noticed that the PMTU value is easy to obtain in GXcast, since unicast paths are used[12].

5. CONCLUSION

A significant share of today's Internet traffic is generated by network gaming. This kind of traffic is interesting in regard to it's market potential as well as to it's real time requirements on the network. Distributed interactive simulations and other network games fit into the multicast applications. The Xcast and Xcast+ protocols permit to manage efficiently a large number of small multicast groups. A major drawback for these protocols is that they are incapable to manage packet fragmentation. Consequently, there is a limit for the multicast group size. In this paper, we proposed an extension to these protocols, named the GXcast protocol.

The GXcast protocol make possible to solve the fragmentation problem and optimizes some criteria like sending less packets and minimizing the header processing time in routers. At the end of the study of this protocol, we showed that GXcast manages easily with a reduction of the cost and time, a great number of groups of average size, even when the members are disseminated on a few hundreds of sub-networks. We concluded that the GXcast protocol is a feasible and promising protocol and very adequate to network games applications kind.

REFERENCES

[1] M. Borella, “SourceModels of Network Game Traffic”, in: *Networld + Interop'99 Engineer's Conference*, 1999.

[2] D.Waitzman,C. Partridge, S.Deering, “Distance Vector Multicast Routing Protocol”, *IETF RFC 1075*, 1988

[3] J. Moy, “Multicast Extensions to OSPF”, *IETFRFC 1584*, 1994.

[4] J. Moy, MOSPF: “Analysis and Experience”, *IETF RFC 1585*, 1994.

[5] A. Siadak, J. Adams, N. W., “Protocol Independent Multicast-Dense Mode (PIM-DM): Protocol Specification” (Revised), *IETF Internet Draft*, 2003.

[6] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, L. Wei, “Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification”, *IETF RFC 2362*, 1998.

[7] M. Ramalho, “Intra- and Inter-domain Multicast Routing Protocols: A Survey and Taxonomy”, *IEEE Communications Surveys and Tutorials*, Vol. 3 (1), 2000, pp. 2–25.

[8] IETF, Reliable Multicast Transport IETF Working Group website, *http://www.ietf.org/html.charters/rmtcharter.html,2003.*

[9] S. Deering, S. Hares, C. Perkins, R. Perlman, “Overview of the 1998 IAB Routing Workshop”, *IETF RFC 2902*, 2000.

[10] D.Ooms,”Taxonomy ofXcast/SGMproposals”, *IETF Internet draft*, 2000.

[11] R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms, O. Paridaens, “Explicitmulticast (Xcast) Basic Specification”, *IETF Internet Draft*, 2005.

[12] S. MyungKI, K. YongJin, P. KiShik, K. SangHa, “Explicit Multicast Extension (Xcast+) for EfficientMulticast Packet Delivery”, *ETRI Journal*, Vol. 23 (4).

[13] B. Cain, S. Deering, A. Thyagarajan, “Internet Group Management Protocol”, version 3, *IETF RFC 3376*, 2002.

[14] J. Postel, “Internet Protocol”, *IETF RFC 791*, 1981.

[15] S. Deering, R. Hinden, “Internet Protocol, version 6 (IPv6) Specification”, *IETF RFC 2460*, 1998.

[16] J. Moy, “OSPF Version 2”, *IETF RFC 1247*, 1991.

[17] A. Boudani, A. Guitton, B. Cousin, GXcast, “Generalized Explicit Multicast Routing Protocol”, in: *The 9th IEEE Symposium on Computers and Communications (ISCC 2004)*, pp. 1000–1005, 2004.

[18] J. Farber, “Network Game Traffic Modelling”, in: *The First Workshop on Network and System Support for Games (Netgames)*, 2002.

[19] W. Feng, F. Chang, W. Feng, J. Walpole, “Provisioning Online Games: A Traffic Analysis of a Busy Counter Strike Server”, in: *The Second ACM SIGCOMM Workshop on Internet Measurement*, 2002.

[20] M. Pullen, M. Myjak, C. Bouwens, “Limitations of Internet Protocol Suite for Distributed simulation in the large Multicast Environment”, *IETF RFC 2502*, 1999.

[21] K. Fall, K. Varadhan, “The NS Manual”, *UC Berkeley, LBL,USC/ISI, and Xerox PARC*, 2001.

[22] Internet2, http://abilene.internet2.edu/, Abilene Network, 2000.

[23] G. Armitage, “An Experimental Estimation of Latency Sensitivity In Multiplayer Quake 3”, in *11th IEEE International Conference on Networks (ICON).*

[24] R. Ramaswamy, T. W. N. Weng, “Considering Processing Cost in Network Simulations”, in: *Workshop on Models, Methods and Tools for Reproducible Network Research (MoMeTools) in conjunction with ACM SIGCOMM*, 2003.

[25] J. McCann, S. Deering, J.Mogul, “Path MTU Discovery for IP version 6”, *IETF RFC 1981*, 1981.