July 11, 2002
RT0475
Network 16 pages

Research Report

Reliable Multicast based on Peer-to-Group Dissemination

Shuichi Shimizu, Taiga Nakamura

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Abstract

This article presents a new reliable multicast scheme for "one-to-many" distributions such as a streaming of digital media in networks. It is based on *Peer-to-Group (P2G) Dissemination*, in which a server simply transmits a single set of data packets toward a group of receiving hosts; each host, on an equal basis in a local network, then makes copies of data packets it received and shares the data with every other receiving host in order to compensate and allow for a full assembly of the original data. By doing so, the system can avoid server overload and network congestion, both of which are serious issues in most ont-to-many distributions. The system is reliable and remains stable even when clients frequently join and leave the service because the server optimizes the flow of packets and leaves the clients with no fixed role in copying packets. P2G Dissemination requires no infrastructure support, unlike IP Multicast, and no secondary servers, unlike the Edge Architecture, but it only requires application-level server and client hosts, where the clients assist in the distribution of packets. It is, especially, suitable for a network infrastructure with large-scale switching hubs. The article also shows some implementation examples and experimental results performed on an actual network in commercial environments.

1 Introduction

In general, server overload and network bandwidth consumption are serious issues especially in "one-to-many" distributions of digital media such as video and audio. If a server duplicates a one-to-one communication by sending the same data set to each client, thus n sets for n clients, then the server may suffer from overload as the number of clients grows increasingly large. In addition, network may be wasted or congested between routers when some clients are located in the same direction seen from the server.

IP Multicast [1] is a sophisticated architecture that addresses the issues of server overload and network congestion in wide area networks. It is implemented at the IP layer, where a server sends out a single set of data which is forwarded by routers to other routers without any duplication until it is received by subscribed hosts. This architecture has been widely accepted and recently implemented in many routers, however, it contains several drawbacks which have prevented it from being deployed. First, in order to keep forwarding tables up-to-date for calculating an optimized distribution tree, all routers have to maintain information of host group membership, even though membership may be changing rapidly. This creates overhead for updating membership information and for re-calculating the distribution tree. Second, it requires a global address assigned from the multicast address space [2] which must be unique throughout routers and host group members. This may lead to another scalability problem. Third, it may require substantial replacement of network infrastructures such as routers and

switches, because every router and switch must be capable of IP Multicast between server and client hosts.

Some application-level multicast approaches [3, 4] have been proposed to overcome one or more of such drawbacks in the IP Multicast system. Unlike IP Multicast, they are implemented at the application layer instead of the IP layer, and thus have no need for any infrastructure supports. However, these application-level multicast systems possess overhead from having to maintain group membership and re-calculate an optimized distribution tree.

Edge Architecture [5, 6] provides another technique to realize efficient data distributions. It introduces secondary servers near the client group in order to reduce load on the primary servers and network traffic between the primary servers and secondary servers. In other words, it uses many replica servers near the clients. However, efficient data distributions in this system is gained only at the expense of higher initial investments and maintenance costs to support the secondary servers.

In this paper, we present our new scheme, *Peer-to-Group (P2G) Dissemination*, which provides reliable one-to-many distributions without the need for network infrastructures support or secondary servers. P2G dissemination allows a server to transmit just one full sef of data packets to a host group regardless of the number of members in the group, through connection-oriented communications. The members then copy and share the data packets with each other, typically in a local network, thus compensatory for each other without the need for a "leader" or secondary servers. Because copying and distribution of data packets are spread evenly, any client can join and leave at any time without affecting the others. In P2G dissemination, server overload would be shared and distributed by exploiting client's computing and networking resources, which are now adequate for such tasks. The scheme is not exclusive but compatible with IP Multicast or the Edge Architecture, and so it can coexist to further make use of their advantages.

In Section 2, we present two sub-schemes, *Peer-to-Group Unicast* and *Group Packet Sharing*, and how they work for multicast. In Section 3, we present an optimization of packet flow and introduce an error recovery and discuss a basic performance of the scheme regarding scalability and reliability. In Section 4, we show some implementation examples and experimental results on an actual network in commercial environments. Finally, we conclude in Section 5.

2 Peer-to-Group Dissemination of Packets

Peer-to-Group (P2G) Dissemination refers to the communications between a peer (server) and a host group (clients), in which the data packets are not necessarily delivered directly to each client, but may take additional hops

until destinations. It consists of two sub-schemes, *Peer-to-Group Unicast* and *Group Packet Sharing*, which enables the server to multicast data packets to group members with significantly reduced network congestion and server overload even in a rapidly changing group environment.

2.1 Peer-to-Group Unicast

For data distributions based on P2G dissemination, as well as the other distribution services such as streaming through RTP [7], original data such as video and audio is segmented into packets, to each of which is attached a serial number, before they are sent out to clients. In IP Multicast, all data packets are sent to a virtual host group through a single multicast address to which clients subscribe.

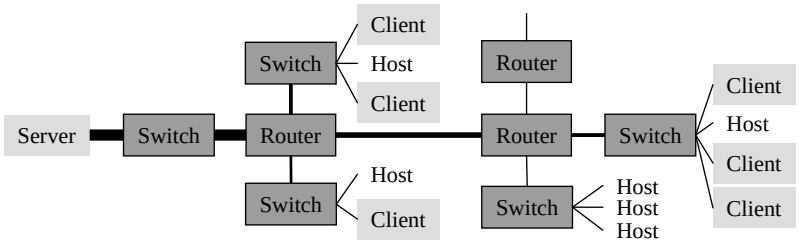
P2G dissemination also employs the concept of a host group, however, unlike IP Multicast a server explicitly and exclusively sends data packets to each client in a group. Since it is a unicast per packet to each client address, no additional multicast global address is required. The transmission is connection-oriented for reliable communication between the server and clients. Another important point is that no clients receive packets with the same serial numbers, which means no packet is ever duplicated between the server and the group. The server decides which client each packet should be sent to. The details are discussed in Section 3.1 below. Since the server exclusively scatters packets, we call it a *spreading server*.

Note that P2G dissemination provides no dynamic routing to the host group, because it does not depend on routing. Instead, it utilizes unicast routes to one or more predefined subgroups provided by conventional network infrastructure such as routers, as shown in Figure 1.

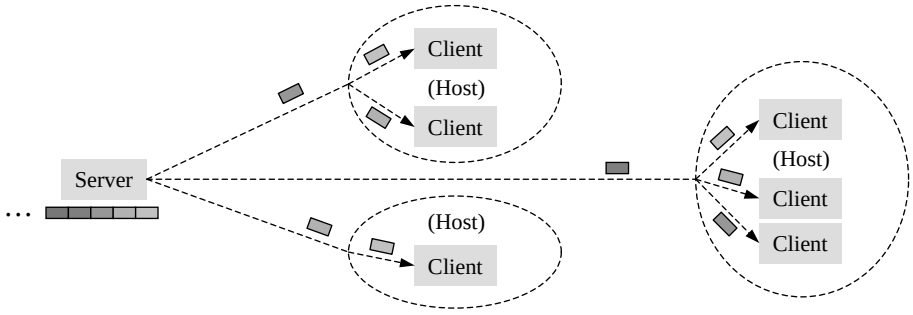
2.2 Group Packet Sharing

As a result of the exclusive unicasts, each host subgroup will receive a single set of data packets, but each client will receive only a partial incomplete set of the packets from the server. Thus, it is necessary for each client to compensate by receiving and sharing the packets with peer clients in the same group. Each client receives *source packets* directly from the server and *copied packets* from peer clients, and assembles them into the original complete data sets. Since each client collects data packets from the server and peers, and it reproduces a single set of data, we call it a *sink client*.

When clients receive source packets, they have to copy and send them to their peer clients as quickly as possible. In this way, the packets are relayed in multiple steps to reach their destination, hence the term, *dissemination of packets*, as shown in Figure 2. Below, we introduce two forms of dissemination of packets, according to the network infrastructures used.



(a) IP Multicast: No packet duplicates by using a minimum spanning tree that connects all clients. Line width indicates the required bandwidth if "one-to-one" transmissions are used for multiple clients.



(b) Peer-to-Group Unicast: A single set of packets is transmitted to each subgroup, in which clients directly and exclusively receive the packets. Routes to subgroups are fixed or provided by conventional routers.

Figure 1: IP Multicast and Peer-to-Group Unicast

2.2.1 Multicast in Local Network

When a host group is configured in a collision domain such as a wireless network, then multicast in the local network [2] may be adequate for exchanging data packets within the group. Because the data exchange does not go beyond any routers, no router support or globally-assigned IP address is required.

When a client receives source packets from the server, it copies and sends them to a local multicast address. There is *one* set of source packets as unicasts coming in from the server, and *one* set of copied packets relayed from the clients. Each client receives *one* set of data packets and transmits $1/n$ set of packets on average, where n is the number of clients in the group.

Broadcast in the local network also works for sharing data packets within group members, but broadcast consumes the computing resources of hosts who aren't participating in the service, and so it may not be appropriate to employ broadcast for sharing data packets.

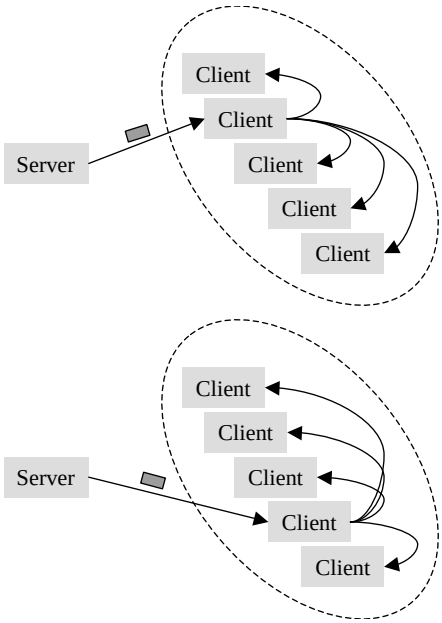


Figure 2: Packet dissemination based on evenly distributions

It should also be noted that the exchange of data packets here may not be reliable and may lead to data loss since the multicast or broadcast is connectionless.

2.2.2 Peer-to-Peer Unicast

When host subgroup members are directly connected to a single switching hub, connection-oriented peer-to-peer unicast becomes much better suited than the multicast, because it is guaranteed that they can communicate with each other and no collision or congestion will occur. In other words, all of the clients under a single switching hub can simultaneously exchange data with no interference from the other client pairs. Thus data packets can be smoothly and reliably exchanged via peer-to-peer unicast for a single switching hub. For example, a 100-Mbps switching hub would be capable of handling 4-Mbps of video packets, which is near TV-quality.

A single group should not extend over two or more large-scale switching hubs (e.g., with 196 ports), because, in that case, many packets may go through the uplinks of the hubs, which may lead to network congestion in uplinks since the hubs are shared by all the down clients. When a group of clients is limited within a single switching hub, then the congestion case described above can be avoided. When small switching hubs (e.g., with 5-8 ports) are attached to the large-scale switching hub so that more clients can be connected, which is common in actual network configurations, though

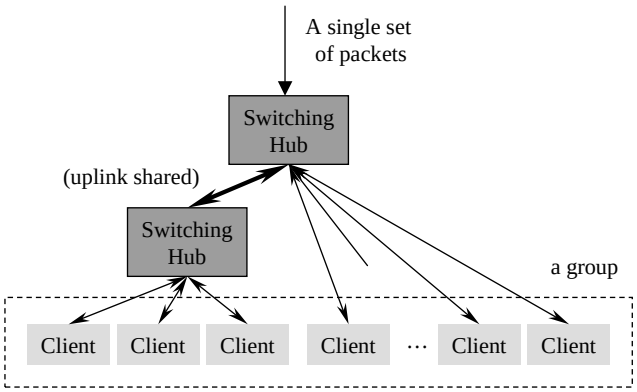


Figure 3: Peer-to-peer unicast under a large-scale switching hub

the uplinks of the small hubs, which are also shared, congestion should not occur when exchanging data packets among the clients as long as traffic does not exceed the capacity of their uplinks, as shown in Figure 3. Each client receives *one* set of packets and it transmits about *one* set of copied packets to his/her peers. However, within a single switching hub, only n sets of data packets go through, which is the same as in the Edge Architecture and IP Multicast, where n is the number of clients in a group, because the outgoing packets from the clients are identical to the packets coming in to them. Therefore, each client in a single switching hub, receives *one* set of source packets and makes $n - 1$ sets of copied packets.

2.3 Dynamic Configuration to Group Membership

A server is responsible for transmitting each source packet to one of clients, and so it has to know which clients are currently active in the service. When using peer-to-peer unicast for sharing packets, each client also has to know which peer clients have recently joined the group because it is responsible for transmitting copied packets to them.

When a new client joins, the protocol to establish new connections is as follows:

1. The new client requests to join in the service,
2. The server determines which group the new client should belong to,
3. The server notifies the clients that have already joined at the group about the new client,
4. The server and the group's clients establish connections with the new client and also update their own peer list to include it,

5. The new client accepts the connections, and builds its own peer list about them.

The peer lists track two types of connections, the source-packet provider, and the copied-packet receivers. When clients receive a packet from the source-packet provider, they copy and send it to the copied-packet receivers and return an acknowledgment packet to the source-packet provider, which is used for measuring the round-trip time, as described in Section 3.1.

For peer-to-peer unicast for sharing packets, when a client stops or leaves the service, the other running clients and server will notice the discontinuation of the connection. This is a trigger for each client to update their peer list and to remove the disconnected peer. Note that clients and the server are able to sense the disconnection even when the peer has abnormally terminated (e.g., by access violation) because they are connection-oriented. In this design, disconnection between any two clients implies that one or both of them have stopped, and so the server is supposed to sense one or two disconnections so that the connection graph should remain complete. We assume that disruptions of connections that make the graph incomplete will not occur or will be so rare that error recovery can handle them.

When sharing packets via local multicast or broadcast, each client does not have to know the peer's state but just sends copied packets into the multicast or broadcast address even when there are no peers in the group. On the other hand, the server is still responsible for maintaining connections to all of the clients.

Thus, simple peer lists and connections make it easy to manage a configuration while allowing clients to enter and exit at any time.

3 Basic Performance of P2G Dissemination

We present how to schedule and optimize the flow of packets in P2G dissemination, which makes the system stable and reliable. We also discuss performance factors regarding scalability and error recovery.

3.1 Dynamic Optimization of Packet Flow

Ideally, in P2G dissemination, clients have an evenly distributed workload, however, the situation may change due to different client performance levels — slower and faster. Some clients might be not able to process a full set of data packets in time because of their limited CPU and memory computing resources. Some clients might temporarily not be able to receive any packets because other processes need more computing resources in the same machine. Thus, the system should be able to adapt not only to clients but also to temporary changes, P2G dissemination optimizes the destinations of each packet using per-packet optimization.

A client has no choice of packet destinations because it is supposed to send copied packets to all peers in the same group as soon as it receives source packets. On the other hand, the server is able to select destinations for each source packet, which would control the reliability and the stability of the packet flow in the system, because some packets might be blocked if slow clients receive a lot of source packets from the server and they are upstream of the others.

Thus, the server's main role is to select destination clients for each packet, according to the following steps:

1. The server chooses clients which are ready to receive packets,
2. Assigns priorities to the chosen clients,
3. Probabilistically selects one of them according to their priorities.

More precisely, the readiness of a client means that the server is able to send the next packet to the client, or the server's kernel that handles networking tasks is ready to accept the next packet for transmitting it to the client even though it might be blocked in the kernel. We denote the set of ready clients as S. The priority for the i-th client is calculated by using the round-trip time (RTT) to it and the copy rate from it to the others, or

$$p_i = \frac{C_i/R_i}{\sum_{i \in S} C_i/R_i}, \tag{1}$$

where p_i is the priority ($0 \leq p_i \leq 1$), C_i is the success ratio of copying ($0 \leq C_i \leq 1$) from the client to its downstream clients, and R_i is the RTT including not only the time for a packet to be transmitted to the client and an acknowledgment to be returned to the server, but also the client's response time that is controlled by the client's computing resources. Note that C_i/R_i represents the effective throughput, that is, the number of packets which have successfully reached downstream clients through the i-th client in a unit of time. The copy rate is periodically sent to the server from each client. When sharing packets via local multicast or broadcast, $C_i = 1$ is used for all clients. Finally, the server sends the next packet to the selected client, and then repeats the steps for new packets.

The situation may change rapidly in each client, because a client process shares computing resources with other processes in the same machine, which might temporarily and unexpectedly require more computing resources. In order to be adaptive and robust to such temporal changes, the RTT and copy rate should be time-sensitive. They should reflect more of the recent situation than the past. A time-average with a forgetting factor might be better for representing both short and long term states, for example, as

$$\overline{R_i(k)} = \frac{1}{2}\left(R_i(k) + \overline{R_i(k-1)}\right), \tag{2}$$

where $R_i(k)$ indicates the k-th (or at time k) RTT measured for the i-th client, and $\overline{R_i(k)}$ indicates the time average in which the recent measurements are weighted more heavily.

The dynamic per-packet optimization described so far is for maximizing the total throughput of data moving towards clients. Accordingly, it maximizes the error tolerance and stability of the system. It is also reasonable to assume that a system that acts on a per-packet basis can be robust enough to group membership dynamics. Clients may can join and leave rapidly, because per-packet operations are of smaller granularity than per-client operations.

3.2 Scalability

As described so far, the server sends a single set of data packets to each group, even when the group contains two or more clients. So, the amount of packet transmitted from the server does not depend on the number of clients but rather on the number of groups, which can be determined prior to starting the service.

At the same time, the server must connect to all of the joined clients and select one of them as a destination for the next source packet for each group. The number of connections and optimization steps depends linearly on the number of clients, which may be a limitation of P2G dissemination.

For sharing packets via peer-to-peer unicast, each client must connect to the others and build a complete graph of all of them. Since the number of clients under a single switching hub can be limited to at most a few hundreds, it is reasonable to say that a server can remain stable while handling such numbers of clients.

Assuming that a server transmits m source packets in a time unit and they are equally delivered to n clients, then each client receives m/n source packets and relays them to $n-1$ peer clients (thus, $m-m/n$ copied packets) when using peer-to-peer unicast. The copied output packets are identical to some of the copied input packets of the peer clients. Thus, m packets go through each client, which means that the total number of packets depends linearly on the number of clients for each group, which is the same as in IP Multicast and the Edge Architecture. Even if packet delivery is not equally divided among clients, the linear relationship between number of packets and clients still holds. This is different from the case of local multicast or broadcast, where there are m source packets and m copied packets regardless of the number of clients.

Distributed evenly, the number of packets that each client has to process in a unit time includes: m/n source packets, $m - m/n$ copied packets for input, and $m - m/n$ copied packets for output, amounting to a total of $2m - m/n$ packets. Since the range spans from m (for *one* client) up to $2m$ (for an infinite number of clients), it is obvious that total number of packets

processed per client does not depend on the number of clients. Thus, the even packet distribution feature of P2G dissemination also makes the system scalable.

In actual situations, according to the per-packet flow optimization, packets are not always delivered equally to each client. Differences among client computing resources leads to uneven distribution for the sake of maximizing total throughput, as described in the previous section. Under extreme cases where all of the packets are delivered to a single client, forcing it to send mn copied packets to peers, scalability is violated. In that case, the client should drop some of the copied packets. However, in all purpose normal cases, the per-packet optimization tends to balance out and distribute evenly within the system, and thus satisfying scalability.

3.3 Reliability

For reliable delivery of the original data, every client should receive enough packets to recover the data. If the throughput to the i-th client is smaller than the number of the packets provided in a time unit, or

$$C_i/R_i < m, \tag{3}$$

then some of the packets will be lost due to transmission timeouts and the client may not be able to receive all of the original data. Because this is due to limitations of the client's computing resources, there are no countermeasures for the client to get the complete set of data. However, our concern should be how to avoid spreading or propagating the effects of packet losses to the other clients. Note that the situations mentioned above might temporarily happen to any clients unless an appropriate amount of their computing resources are reserved for client processes, but such reservation can not be relied on in actual situations. A solution, therefore, is the per-packet optimization scheme, which probabilistically places slower clients downstream, thus decreasing the probability that packets are lost upstream. However, the packet loss still may be above zero.

Another possibility for packet loss is that one or more clients leave without flushing the copied packets, which can happen by accident. Auto Repeat Request (ARQ) is one of the solutions for recovering packet losses, but, ARQ is likely to cause network congestion and overload a server, while delaying packet arrival. Consequently, P2G dissemination should employ Forward Error Correction (FEC) or erasure code technique [8] for recovering from losses, which bypasses the need to communicate with the server for recovering erasures.

Note that FEC does not necessarily recover all erasures when the erasure rate exceeds the capacity of the code. In this case, the clients may have to find and obtain the lost packets by themselves for recovering binary data such as software packages. For continuous digital media such as MPEG2

and MPEG4 standards, they should temporarily stop playing when unrecoverable erasures occur, or allow an error-resilient property of the media format to overcome the erasures.

Packets do not necessarily arrive at each client in sequential order, rather they may be temporally swapped because the unevenness of the peers potentially produces variable delay. So, for receiving continuous digital media, each client should buffer for the variance of delay so that it can decrease the number of pending packets before reading. The length of the buffer (or pre-buffer) should be long enough to hold most of the jittered delay so that the FEC is able to recover the packets that have not arrived in time, or

$$\text{Prob}\,[d > L] < P_{\text{FEC}}, \tag{4}$$

where L is the length of the pre-buffer, d indicates the random variable of delay, and P_{FEC} is the maximum recovery rate of the erasure code. As mentioned above, the FEC is also expected to recover erasures caused by low computing resources and unflushed copied packets, and so an appropriate margin should be used in the left hand side of the above inequality.

4 Implementation Examples and Experimental Results

We now show some implementation examples of P2G dissemination, and also some experimental results and analysis. This concrete discussion is based on an actual prototype implementation.

4.1 Large-Scale Switching Hubs and Shared Backbone

Consider a network where some large-scale (e.g., with 192 ports) 100-Mbps switching hubs are connected to a 1-Gbps backbone that consists of a few routers, as shown in Figure 4, which may be a typical configuration of the network infrastructure for many commercial buildings. Each personal computer (PC) connects to one of the switching hubs. In addition, some small ($\sim$ 8 ports) switching hubs with a few ports may be inserted between the large-scale hub and the end PCs.

Even though the 1-Gbps backbone has high bandwidth, it is shared by many end PCs, and so packet duplicates should be avoided on the backbone. For example, it is not possible to serve even 250 streams of 4-Mbps bandwidth at the same time. On the other hand, the PCs are connected to each other with 100-Mbps bandwidth under a single large switching hub, but they cause no network congestion with 4-Mbps communication since the communication is limited within the group of PCs under the switching hub, even if the number of PCs exceeds 250.

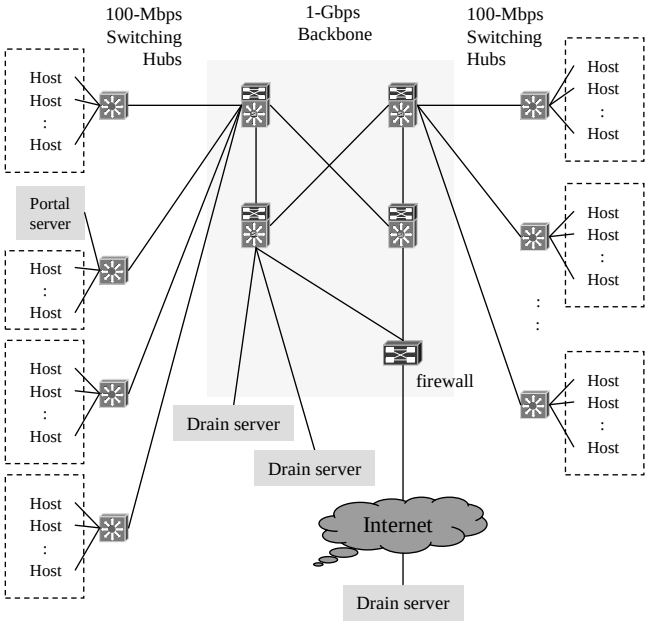


Figure 4: Implementation Example

In this configuration, one or more spreading servers are located in the backbone and they may transmit as many sets of streaming packets as the number of the switching hubs, because each switching hub forms a single host group, where each spreading server serves one or more groups. When there are N large switching hubs in the infrastructure, then at a maximum, N sets of streaming packets go through the backbone, even when the number of clients increases to be much larger than N. Thus, streaming a 4-Mbps (i.e., TV quality) or an 8-Mbps (i.e., DVD quality) bandwidth (e.g., 720×480-resolution video with 48-KHz audio) would be acceptable in this configuration, where N is typically 20 to 50.

Establishment of the service is as follows: First, a new client accesses the portal server shown in Figure 4, and the portal determines the group to which the client should belong, according to its IP address, and it assigns the client to the corresponding spreading server. The spreading server notifies the clients currently running in the group that a new client has joined, so that they can establish new connections with the new client.

Spreading servers and sink clients are individually probing for disconnections from each other to maintain a complete graph, and if necessary, they update their own peer lists for later transmission of copied packets.

4.2 Wireless network

P2G dissemination is also applicable to wireless networks such as "host spots" in public spaces and a company intranet that consists of wireless-networking PCs, as mentioned in Section 2.2.1. In this implementation example, each client locally multicasts copied packets for its peers when it receives source packets from a spreading server in order to share and recover the original data. Because spreading servers establish connection-oriented communications with clients, they are able to sense the end of the connections and stop transmitting packets, when some of the clients get out of the "hot spot" area. The clients are also able to sense the disconnection. This is one of the advantages over using an edge server that multicasts or broadcasts data packets to possible clients through unreliable communications that may not be able to detect the end of a connection.

The portal server and spreading servers are exactly the same as in the previous implementation example, because they don't need to know how to share packets within peers in a group. Each client just ignores a new member notification from the server. Thus, this configuration is able to coexist or be integrated into the previous implementation example on typical company networks.

4.3 Experimental Results

We conducted several medium-scale experiments which served 4-Mbps streams of TV-quality sports programs to many end user PCs. The data source was MPEG2-encoded in real-time and fed to a spreading server. The network configuration was almost the same as in Figure 4, except that we had no portal server but only one spreading server that served eight groups, each of which corresponded to a single switching hub that maintains about 200 PCs.

The original MPEG2 was segmented into 4-KB packets with serial numbers, and then the packets were FEC-encoded so that 116 packets were extended to 128 packets, which means that up to 12 erasures can be recovered from in every 128-packet block. The internal buffer for receiving packets is 2048 packets long, which is 16 seconds for 4 Mbps. The initial pointer for playing the MPEG2 was set to the middle of the buffer, and so the length of the pre-buffer was about half of the internal buffer (e.g., 8 seconds for 4 Mbps). Each client sent the copied packets to its peers on a last-in-first-out basis, that is, each client sent the latest source packet first so that most of the clients would receive the latest or almost-latest packets to synchronize their internal buffers.

In total, 122 clients had joined the experiment from the eight pre-defined groups, for 7544 seconds on average, with 50 clients served in the largest group, and 25 clients in the second largest group at the same time.

Overall, the spreading server, which was equipped with an Intel Pentium-Ⅲ/500 MHz and 128 MBs of memory, needed no more than 50% of its CPU resource to serve 102 clients simultaneously at maximum. The participating clients were equipped with Pentium-Ⅲ/500 MHz to Pentium 4/2.2 GHz CPUs. Most of the clients, except for some PCs with internal problems (see below) played the streaming data smoothly, and most of the CPU resources were spent on decoding and rendering the MPEG2 video and audio.

The reliability and stability of the system was measured in terms of dropped and blocked packet, as follows:

4.3.1 Packet Dropped at Server

If the computing resources assigned to a spreading server or client was not enough to transmit or receive packets, then the spreading server was not be able to transmit source packets to some clients, but might drop some of them, because the source data was a live video source with audio, and a timeout was necessary. However, in our experiments, we didn't observe this situation. No packets were dropped at the server, and at least one client was always ready to receive packets throughout our experiments.

4.3.2 Packet Dropped by Peers Upstream

If the computing resources assigned to a client were permanently or temporarily insufficient to transmit or receive the copied packets, then some packets would be dropped at clients upstream, because the internal buffers have a limited size which causes timeouts when sending copied packets. Denoting as c_i the total copy rate of the i-th client for the time duration t_i, then the time-weighted copy rate or $\sum_{i=1}^{i=n} c_i t_i \Big/ \sum_{i=1}^{i=n} t_i$ was measured to be 0.9987, where n is the number of clients. That indicates that 0.13% of the copied packets didn't flow to peers downstream, but this was small enough to recover the dropped packets by using the 128/116-FEC decoding (i.e., up to 9.375% losses are recoverable).

Since the packet dropping rate at the server and upstream peers represents the basic performance of P2G dissemination, the results show that P2G dissemination operates well as a reliable one-to-many multicast distribution system.

4.3.3 Packet Blocking

Final packet losses at the end clients may be brought by not only packets dropped at upstreams mentioned in the previous sections, but also packets blocked at the clients themselves.

Denoting as e_i the total unrecoverable erasure rate occurred at the i-th client for duration t_i, the overall time-weighted average of the erasure

rate was 0.42%, but most clients experienced near zero rate. Therefore, most of the erasures occurred in a limited set of problematic clients. in our experiment, eight clients suffered from constant erasures. The erasure rate for these eight PCs was 5.00%, but only 0.14% for the rest of the PCs. Note that 0.14% erasures would result in one frame drop for every 24 minutes if they are burst.

Out of the eight problematic clients, one client solved the problem by adjusting networking parameters such as the Maximum Transmission Unit (MTU) to appropriate values. It was also found that four clients were equipped with improper network interface cards (NIC), for example, 100-Mbps Ethernet PC-card with no CardBus support, which may not be sufficient for double 4-Mbps streaming — 4 Mbps for input and almost 4 Mbps for output of copied packets. The problems of the remaining three clients were unknown, but they are most likely the result of the the PC's own problems.

Also note that some other clients temporarily showed unrecoverable erasures when other processes in the same PC consumed more computing resources (e.g., in starting and reading from a mail program), but this is unavoidable unless some of the computing resources are reserved for the P2G dissemination client process. Countermeasures on the player side might be necessary to avoid this type of stream discontinuation.

5 Conclusions

We have presented a new reliable multicast scheme, Peer-to-Group Dissemination, in which each client participates on an equal basis with no leaders or secondary servers as found in the Edge Architecture. Furthermore, P2G dissemination requires no network infrastructure support as opposed to IP Multicast.

P2G dissemination is composed of two sub-schemes, Peer-to-Group Unicast and Group Packet Sharing. The Peer-to-Group Unicast allows a single set of packet transmissions from a server to a group of clients to avoid network congestion and server overload. The Group Packet Sharing exploits the peer-to-peer unicast capability guaranteed for clients connecting to a single switching hub, so that they can smoothly and reliably exchange data packets. For collision domains such as wireless networks, it utilizes multicast or broadcast in a local network.

A server optimizes and determines destinations for each packet and then sends it to one of the clients as a spreading source, while each client collects source packets from the server and copied packets from peers as a sink destination. The per-packet optimization by the server and the equal-basis clients allow for dynamic group memberships, such as the entering and exiting of several clients to occur without affecting the overall stability of the

system. In addition, connection-oriented communication and Forward Error Correction (FEC) increase the system's reliability and stability.

The equal-basis clients potentially cause variation in packet arrivals, and so each client needs to pre-buffer packets to deal with delivery problems, but this might make P2G dissemination unsuitable to real-time streaming for such uses as video conferences.

From the experimental results performed on an actual network environment in which a single small server simultaneously served more than a hundred clients, we have observed no problems with the basic performance of P2G dissemination.

References

[1] S. Deering, "Routing in internetworks and extended lans," Tech. Rep. STAN-CS-88-1214, Stanford University, Department of Computer Science, July 1988.

[2] "RFC-3171: IANA guidelines for IPv4 multicast address assignments," 2001.

[3] Dimitrios Pendarakis, Sherlia Shi, Dinesh Verma, and Marcel Waldvogel, "ALMI: An application level multicast infrastructure," in *3rd USENIX Symposium on Internet Technologies and Systems (USITS)*, 2001, pp. 49–60.

[4] Yang-Hua Chu, Sanjay G. Rao, and Hui Zhang, "A case for end system multicast," in *ACM SIGMETRICS 2000*, Santa Clara, CA, June 2000, ACM, pp. 1–12.

[5] W3C, "Edge architecture specification," W3C Note 04 August 2001, http://www.w3.org/TR/edge-arch.

[6] "Akamai," http://www.akamai.com.

[7] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RFC 1889: RTP: A transport protocol for real-time applications," Jan. 1996.

[8] J. Blomer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman, "An xor-based erasure-resilient coding scheme," Tech. Rep., International Computer Science Institute, Berkeley, California, 1995.