다자간 협업 환경을 위한 UDP 터널링 기반의 멀티캐스트 연결성 솔루션의 구현
(Implementation of UDP-Tunneling Based Multicast Connectivity Solution for Multi-Party Collaborative Environments)

김 남 곤[†]　김 종 원[††]
(Namgon Kim)　(Jongwon Kim)

요 약 다자간 분산 협업 시스템인 Access Grid (AG)는 원격지에 존재하는 다수의 사용자들 사이에 멀티미디어 정보를 효율적으로 주고받기 위해 IP 멀티캐스트를 활용한다. 하지만 아직까지 많은 네트워크들이 IP 멀티캐스트를 지원하지 않고 있어, AG 기반의 원격 협업 환경을 실제 활용하는 데 어려움을 겪고 있다. 이러한 IP 멀티캐스트 연결성에 대한 해결책으로 AG 커뮤니티에서는 IP 멀티캐스트 네트워크 내에 릴레이 서버를 두고 이 서버와 UDP 연결을 통해 멀티캐스트 데이타를 전달받는 형태의 멀티캐스트 브리지를 활용하고 있다. 하지만 멀티캐스트 브리지는 각각의 유니캐스트 피어에게 동일한 데이타를 중복해서 전송하기 때문에, 시스템과 네트워크 활용 측면에서 문제를 가지고 있다. 본 논문에서는 AG의 IP 멀티캐스트 연결성 문제에 대한 대안으로 UMTP(UDP multicast tunneling protocol)에 기반한 멀티캐스트 연결성 솔루션을 제안한다. UMTP는 응용 계층의 멀티캐스트 터널링 프로토콜로, 멀티캐스트 네트워크에 존재하는 노드와 멀티캐스트가 불가능한 네트워크에 존재하는 노드를 UDP 터널을 이용해 연결하고, 멀티캐스트 데이타를 캡슐화 하여 전달, 상호 간에 멀티캐스트 연결성을 제공한다. UMTP의 장점들을 취하여 제안된 솔루션은 시스템과 네트워크 활용에 효율성을 높이고, 또한 방화벽 환경에 적용 가능하도록 설계되었다. 본 논문에서는 이러한 설계를 기반으로 AG에 활용 가능한 멀티캐스트 연결성 솔루션인 AG Connector를 구현, 검증한 결과를 보인다.

키워드 : Access Grid, 협업 환경, 멀티캐스트 브리지, 오버레이 멀티캐스트, UDP 멀티캐스트 터널링 프로토콜

Abstract The Access Grid (AG) provides collaboration environments over the IP multicast networks by enabling efficient exchange of multimedia contents among remote users; however, since lots of current networks are still multicast-disabled, it is not easy to deploy this multicast-based multi-party AG. For this problem, the AG provides multicast bridges as a solution by putting a relay server into the multicast networks. Multicast-disabled clients make UDP connections with this relay server and receive forwarded multicast traffics in unicast UDP packets. This solution is facing several limitations since it requires duplicate forwarding of the same packet for each unicast peer. Thus, in this paper, we propose an alternate solution for the multicast connectivity problem of the AG based on the UMTP (UDP multicast tunneling protocol). By taking advantage of flexibilities of UMTP, the proposed solution is designed to improve the efficiency of network and system utilization, to allow reuse of multicast-based AG applications without modification, and to partially address the NAT/firewall traversal issues. To verify the feasibility of proposed solution, we have implemented a prototype AG connectivity tool based on the UMTP, named as the AG Connector.

Key words : Access Grid, advanced collaboration environment, multicast bridging, overlay multicast, and UDP multicast tunneling protocol

· This research is supported in part by the Ubiquitous Computing and Network (UCN) Project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in Korea.

† 학생회원 : 광주과학기술원 정보기전공학부
ngkim@nm.gist.ac.kr
†† 종신회원 : 광주과학기술원 정보기전공학부 교수
jongwon@nm.gist.ac.kr

논문접수 : 2006년 12월 4일
심사완료 : 2007년 5월 25일

1. 서 론

The Access Grid (AG) [1] is a multi-party distributed collaboration system. The AG is developed to share video, audio, and data among number of users over the IP multicast networks. The native IP multicast is known as an efficient way of doing one-to-many and many-to-many communications, since we can distribute data to multiple users by just sending out a copy of data from a sender. However, due to the complexity and difficulty of involved configuration and operation, most of network providers are reluctant to support native IP multicast in their networks.

With the AG, the availability of native IP multicast is very important to the deployment of this multi-party collaboration environment. For this, the AG community has been using a multicast connectivity solution called as QuickBridge [2]. The QuickBridge server is located in multicast-enabled network and does data relay between multicast-enabled and multicast-disabled networks. The AG users in multicast-disabled networks can receive the multicast traffic from the QuickBridge server in UDP unicast packets. Although its operation is rather simple and straightforward, it is subject to potential waste of network bandwidth. Moreover, due to one-to-one mapping between multicast and unicast addresses, the QuickBridge needs to open ports according to the number of connected users. It is thus very difficult to apply this solution for users under firewall environment.

For the ubiquitous deployment of the AG, we also need to solve other connectivity problems than the multicast connectivity problem. The ultimate target of connectivity solution for the AG is depicted in Fig. 1. We need to provide solutions for three kinds of connectivity problems: 1) connectivity to AG nodes under unicast-only networks, 2) connectivity from/to AG nodes under NAT/firewall environments, and 3) connectivity to AG sessions from low bandwidth/performance systems. Within the AG community, the first multicast-connectivity problem is partially addressed by the QuickBridge solution. However, for the rest two, no clear solution has been provided yet. Under the NAT/firewall environments, network managers are strictly restricting the number of open ports [3]. Also, the access is asymmetrically controlled by limiting access from outside. Although solutions are being proposed for unicast-based communication scenario, it is not easy to address together with multicast connectivity. Also, the AG nodes with low-bandwidth network or low-performance systems can not join the AG sessions since the bandwidth and/or system power (to receive and to decode the video streams) cannot be met. To support these poorly equipped AG nodes, the connectivity solution should provide either media-side scalability or transcoding capability.

In this paper, as a first step toward these objectives, we introduce an alternative multicast connec-

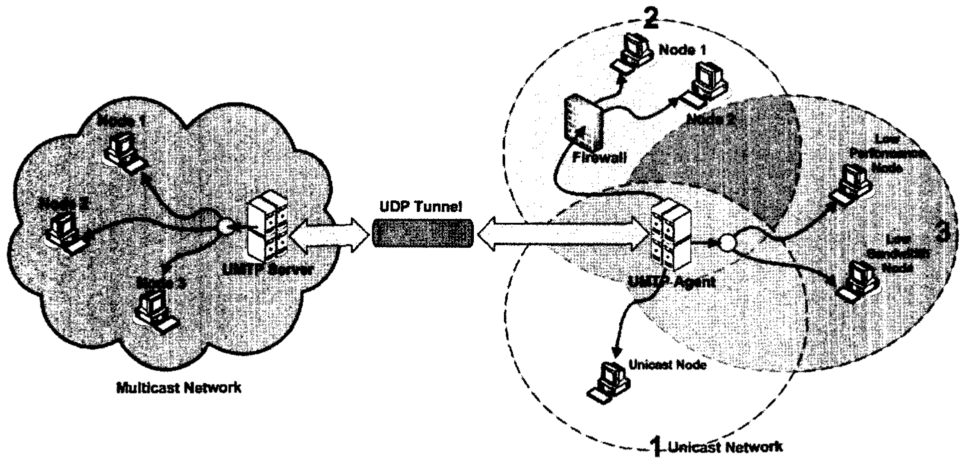


Fig. 1 The objectives of the proposed AG connectivity solution

tivity solution for the AG that can replace existing QuickBridge-based solution. It is based on an application-level tunneling protocol for the last mile multicast, the UMTP (UDP multicast tunneling protocol) [4]. The UMTP creates a UDP tunnel between two tunnel endpoints (TEs). One TE is located in a multicast network while the other exists in a network that is not multicast-reachable to the other endpoint. Each TE receives multicast data in its network. Received multicast data is encapsulated to a predefined format and delivered to its matching endpoint. The UMTP thus communicates with end-user application by multicast instead of unicast. This gives a lot of advantages for network, application, and UMTP-based connectivity solution itself. With the proposed solution, we can improve the network efficiency by maximizing the multicast usage, enable multicast-dependent AG applications to run without modification, and partially address the NAT/firewall traversal problem.

The rest of this paper is organized as follows. In Section 2, the AG and existing bridging solutions are explained. In Section 3, we introduce AG connectivity solution based on the UMTP and explain how the UMTP should be combined with AG. In Section 4, we extend the concept of the UMTP. The implementation and verification follow in Section 5. After that, we conclude the paper in Section 6.

2. The AG and Existing Bridging Solutions

The AG is a multi-party collaboration system supporting group-to-group interaction. The AG provides various functions for sharing information to construct collaboration environment. The AG users can share these function only if they are in the same space for collaboration, called as a venue. When they are ready to join the collaboration session, they enter into the venue by contacting a venue server with a venue client tool. They will receive information to configure their node services and to launch required services including media producer/consumer services. They are also informed of other participants (i.e., nodes) from the venue server.

Multicast enable makes the AG possible to send data to all users with just one copy of data. The AG reduces bandwidth requirement for multimedia communication among participating nodes with native IP multicast. Also, since the AG nodes are sharing a multicast address for the media producer/consumer services (i.e., video send/receive), they do not need location information (i.e., IP address) of other nodes to communicate with.

However, as discussed already, multicast support is not yet enabled for all the networks. As of now, most of the routers support multicast protocols. But, multicast routers are more expensive and hard to configure than unicast only routers. What is more important is that the Internet service providers (ISPs) are not motivated enough to take the operational overhead. This issue is related with the lack of reasonable pricing policy for the multicast service. As a result, the wide deployment of native IP multicast has been slowed down.

Thus, for the AG, which is currently operational with numbers of users around the world, a multicast connectivity solution named QuickBridge is being utilized. The users in multicast-disabled networks can join the AG session through the QuickBridge servers, located in multicast-enabled networks. The QuickBridge servers are usually tied with (i.e., registered to) specific venue servers and they retrieve multicast group address list from the venue. Each QuickBridge server joins these (retrieved) multicast group addresses and relays the multicast data to multicast-disabled AG nodes. Each multicast-disabled node can receive multicast traffic from the QuickBridge server through a dedicated unicast UDP connection. The QuickBridge server individually manages this unicast connection with each AG node and sends/receives data to/from it. This makes multicast-disabled users can communicate with users in the venue. Fig. 2 shows the operations of QuickBridge.

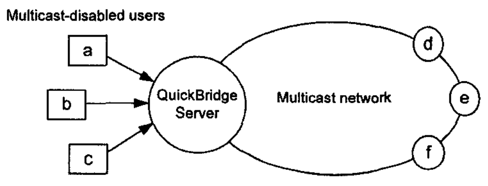


Fig. 2 QuickBridge server for the AG

As mentioned above, the QuickBridge uses unicast UDP datagrams to provide multicast connectivity. Although the use of unicast UDP may provide required connectivity simply, it brings in several disadvantages. It consumes more bandwidth than using multicast, since the QuickBridge server should send out separate packets to each user. This requires additional network bandwidth for each multicast-disabled user. Also, this iterative data delivery requires more system resources. Even if the QuickBridge server has just one multicast address to relay, as the number of unicast users are increasing, it consumes more network bandwidth and system performance. Due to these problems, most of the multicast connectivity schemes avoid using UDP unicast. Another multicast connectivity tools, such as liveGate and mTunnel, use multicast packets as much as it can when it delivers multicast data to multicast-disabled users. Finally, a tool named as RCBridge [5], even though it shares several features with QuickBridge, also provides 'rcb-forward', which uses multicast with end-user applications. In addition, the QuickBridge server creates unicast addresses as many as the number of multicast addresses in the venue server. To enable the QuickBridge server, the firewall should open the ports needed to communicate with the server. However, opening ports under the firewall environment is extremely restricted by the network operators.

These disadvantages of QuickBridge are justifying that we need a better, yet efficient, multicast connectivity solution for the AG. It should not induce any additional burden to the network and multicast connectivity system itself. If possible, it should reuse existing open ports so that it can be easily applicable to the firewall environment. In the following section, we introduce a new multicast connectivity solution for the AG with these features. The proposed solution leverages data encapsulation to decrease the number of open ports for data relay. It also uses multicast as much as possible in delivering data to the end-user applications, thus lowering the minimum requirements on the network bandwidth and system performance.

3. AG Connectivity Solution based on the UMTP

3.1 UMTP

The proposed multicast connectivity solution is based on UMTP. As discussed, the UMTP is an application-layer tunneling protocol that provides multicast connectivity. The UMTP operates using unicast UDP datagrams (i.e., packets) between pairs of nodes. Each node forms a TE and each datagram is either a 'command' or 'data'. With the commands, a TE can create a UDP tunnel with the other TE. After constructing the tunnel, they exchanges data with each other through this tunnel.

The TE can act either as a "master" or "slave". It is defined that the TE trying to access the multicast network is the tunnel master and the other already in the multicast network is the tunnel slave. The tunnel master periodically sends join command for a particular (group, port) to its tunnel slave. This instructs that the announced (group, port) is still of interest and should be tunneled. The tunnel slave starts tunneling for selected (group, port) when it receives join command from its tunnel master. It also stops tunneling for selected (group, port) when it receives leave command from its tunnel master or it has not received any join command from the master for some period of time.

Each TE exchanges UDP payload with a form of UMTP descriptor. Fig. 3 shows the UMTP descriptor. The 'command' field of the descriptor decides the action of TE. The UMTP defines several commands and exchanges information according to these commands. The 'multicast address' and 'port' field consists (group, port) that need to be tunneled or currently being tunneled. For commands other than "DATA", this descriptor makes up the entire UMTP payload. In the case of "DATA" command, the UMTP descriptor is preceded by the data from the (group, port) being tunneled.

With this UMTP descriptor, the UMTP enables data encapsulation in relaying multicast data between TEs. Each TE of a tunnel receives multicast data from (group, port) being tunneled. Before

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  (IPv4) source address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         source cookie         |      destination cookie       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  (IPv4) multicast address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             port              |      TTL      |S|vers.|command|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 3 16-octet UMTP trailer descriptor

delivering the data to the other TE, the TE constructs a UMTP payload by adding UMTP descriptor to the received multicast data. The TE that receives this UMTP payload can know the destination (group, port) of the multicast data. This enables TEs to share just one UDP port for the tunnel exchange of multicast data destined to various (group, port) pairs.

A TE retrieves multicast data from the UMTP payload received from the other TE. It re-multicast retrieved multicast data to its network locally. Multicast makes one copy of data reach all users in same multicast network. Multicast removes the need to repeatedly send data to each receiver. It thus can save network bandwidth and system resource of the TE. The UMTP is good for utilizing multicast infrastructure. Sometimes, users can see other participants in a same multicast group after certain amount of time. This is because it takes some time for setting up new multicast path among participants. It can take longer than we expect. Applying QuickBridge for this causes the router to stop multicast path setup; however, with the UMTP, by using multicast delivery, end user application can request multicast path setup to the multicast router. After setting up the path, participants can communicate in native IP multicast.

3.2 UMTP-based AG Connectivity Solution

The UMTP has a proper architecture to resolve the connectivity problems of the AG. It is a tunneling protocol which uses multicast in delivering data to end-user application. It can share a single port in exchanging data between networks by using encapsulation. This means that the UMTP can be easily applicable to networks behind NAT/firewall. Using multicast removes the need for restarting media. The architecture of exchanging data through the TEs will make stream control easy for low bandwidth/performance users.

The AG provides media supports for video and audio, respectively. The video service for the AG uses VIC (video conference tool). The audio service uses RAT (Robust Audio Tool). Both VIC and RAT has originated from MBone (multicast backbone) tools. They are using RTP (real-time transport protocol) above UDP/IP as its transport protocol, and conforming to the RTP profile for audio and video conference with minimal control. The RTP uses RTCP (RTP control protocol) to monitor the quality of service and to convey information about the participants in an on-going session. RTP and RTCP are operating on different open ports. This means that to use both video service and audio services, we needs to open four ports to enable communication with other participants for the AG session. If there are other users who are joining other AG sessions in the same network, then the network should open ports with multiple of four. This is not an easy decision for a network administrator of a company or institution, where NAT/firewall is running for their network security, because opening many number of ports may cause security problems. One may allow the video and audio services to select the ports dynamically. Since each venue uses different ports (i.e., port numbers), most of the cases, these port numbers are allocated randomly. This dynamic port allocation feature of the AG actually makes the network administrator more reluctant to accommodate the request of open ports for the AG.

The UMTP can be a solution for this problem of the AG. The UMTP operates on a single static port. The UMTP creates a UDP tunnel between UMTP master and slave with predetermined port number and exchanges every data only through this UDP tunnel. As discussed, the data encapsulation feature of the UMTP makes it possible. By using multicast, the UMTP removes the need to restart the video and audio services of AG. When a venue client starts, it runs video and audio services with multicast address for these services. To utilize the QuickBridge, the venue client should stop the video and audio services and restart these with QuickBridge's address. Only after restart, these applications can receive data from the QuickBridge server. However, with the UMTP, it is not required because it utilizes multicast locally. Therefore, providing multicast connectivity with the UMTP will enable persistent use of AG sessions even when the network experiences temporary multicast break. In addition, developing an application that works for both multicast and unicast environment is not easy. Partly due to this reason, AG media applications sometimes crash when we try to switch to the QuickBridge service. However, the UMTP-based connectivity solution will remove these concerns and enable more stable AG experience from any types of networking environment.

To utilize the UMTP for the AG, a tool called 'AG Connector' is developed in this paper. It is composed of three components: UMTP Server, UMTP Agent, and AG Connector shared application. The UMTP Server and UMTP Agent are implementation of UMTP slave and UMTP master, respectively. The UMTP Server waits for the connection request from the UMTP Agents. The AG Connector shared application [6] is a user interface. It runs the UMTP Agent and retrieves multicast group address of video and audio services for current AG session from the venue. The AG Connector shared application requests the UMTP Agent to join the multicast group. The UMTP Agent then joins the multicat group and redirects join request to the UMTP Server. Fig. 4 shows this inter-operation among AG Connector components. This relay of join request makes the UMTP Server and UMTP Agent share the same multicast group and makes data relay possible.

4. Extensions to the UMTP

In this section, two extensions of the existing UMTP protocol are discussed. First, we need to address the possible multicast looping problem. As discussed already, at both TEs, the UMTP uses multicast to deliver data to end-user applications. This makes the possibility of two TEs being connected by native multicast or via another UMTP tunnel elsewhere. Thus, after reviewing existing schemes to detect the erroneous looping for the UMTP, we introduce an effective method to detect and remove the looping. The second extension is adding the notion of multicast group to the UMTP, since the UMTP specification does not address the way of knowing multicast group address that needs to be tunneled.

4.1 Avoiding Multicast Looping via MPROBE Protocol

For the looping problem, the UMTP addresses it as follows. First, when multicasting a payload extracted from a DATA packet, a UMTP implementation should choose an appropriate TTL (time to live) value. It must also verify whether this packet is re-received or not. Second, if the UMTP implementation receives a multicast packet whose source address is the same with that of TE, it must immediately shut down this tunnel. Third, if looping seems possible under current tunneling topology, each TE should periodically send a short 'status' packet - containing its unicast address - to a common multicast address. Each TE then should listen to this address and check the possible looping by examining received status packets.

However, these schemes have several limitations. The first scheme is facing implementation difficulty. The packet that causes data looping can be removed from the network if its TTL value is set to 0. To do this, the UMTP implementation should check

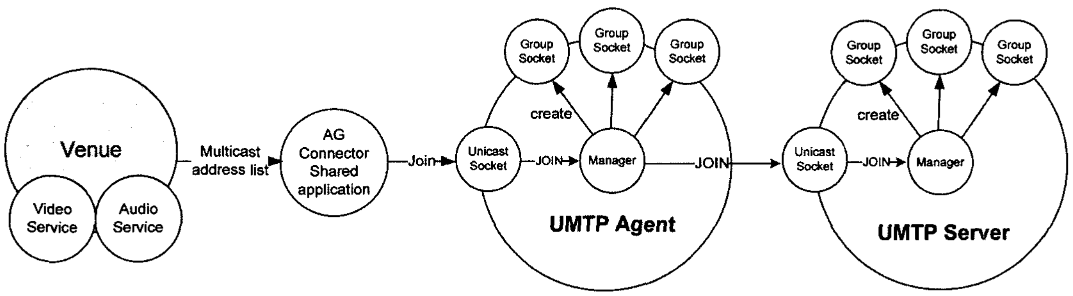


Fig. 4 Interoperation among the AG Connector components

the TTL value of received multicast packet and relay this packet to the other TE with decreased TTL value. However, in the standard version of socket interface under Unix and Windows, there is no way for a user application to know TTL of an incoming packet. Therefore, as an application layer protocol, UMTP can not know the TTL of received multicast packet. This makes implementing the first scheme impossible. The limitation with the second scheme is that each TE should check the source address of every packet. Since it should be verified all the time, it can be an exhaustive and resource-consuming job for all TEs. The third status packet scheme that shares a common multicast address for every TE seems the most suitable for detecting multicast loop. However, current UMTP specification does not provide detailed procedure on this. In this paper, we work out a way for doing this under the name of MPROBE protocol. The MPROBE protocol is designed to remove possible looping by detecting the existence of TEs in same multicast network. A TE emits MPROBE packets to its network to notify other TEs of its existence. A TE can detect the existence of the other TE, as well as the TE that constitutes other tunnel elsewhere, in same multicast network. There is several possibility of looping with the UMTP. Both endpoints that construct a tunnel can be in same multicast network. Also, TEs that belong to different tunnel can be in same multicast network. We can divide them into following cases according to the type of TE: 'master' or 'slave'.

STEP 1. Let us imagine the case of master TE. If it receives MPROBE packets from the slave of its tunnel, it means these two TEs are in same multicast network and each can receive multicast data from each other. It is the same when the slave TE receives MPROBE packets from the master. In both cases, the master TE should disconnect from the slave and leave the multicast group. This is because the slave TE can construct another tunnel with other TEs.

STEP 2. The slave TE can receive MPROBE packet from another slave TE. In this case, the loop may be created or not depending on the relationship of master TEs for these slave TEs. If two master TEs are also in the same multicast network, it can cause the multicast data looping. Multicast packets from a tunnel will be relayed by other tunnel and they continue until one of these tunnels close. However, if the master TE of each slave exists in different multicast network, it will not cause looping.

STEP 3. The master TE can receive MPROBE packets from another master TE. Two master TEs in a multicast network can cause looping, if they tunnel for the same (group, port). If they are connected to the same slave TE, it definitely creates a loop. When they are not connected to the same slave TE, looping depends on the relation between the slave TEs. If the slave TEs belongs to the same multicast network, it causes looping. Thus, these two master TEs should not have tunnel for the same (group, port). One more implication with this case is that it is not easy for the master TE to know the relationship between their slave TEs.

In summary, in order to detect the possible multicast looping and to react properly, we should know the type of other TE. By exchanging the information about the slave TEs, we can ease the loop detection process. In Fig. 5, the MPROBE packet format to convey the required information is defined. The 'command' field contains the type of MPROBE packet. Each MPROBE packet has four types: MPROBE, MPROBE_ACK, MLEAVE, and MTEAR_DOWN. The first type, MPROBE, is used to detect other TE in the .same multicast network. MPROBE_ACK stands for the acknowledgement for MPROBE packet. This MPROBE_ACK packet consists of (group, port) list of the involved TE. It is sent to the TE that sends the MPROBE packet. The MLEAVE packet contains the list of group addresses that conflict with (group, port) list of the TE that sends MPROBE_ACK. This MLEAVE packet commands to the TE (that sent MPROBE_

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            command            |             flag              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 (IPv4) destination address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 (IPv4) slave address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 5 MPROBE packet format

ACK) to leave the group that is in the list of MLEAVE packet. The last MTEAR_DOWN packet handles the case of master TEs having the same slave TE. In this case, one of them should be shut down and the decision is made based on the IP address of each TE. The TE with bigger IP address will send MTEAR_DOWN to the other and the one that receives MTEAR_DOWN will shut down its operation. Fig. 6 and Fig. 7 show the detailed operation after receiving MPROBE and MPROBE_ACK of MPROBE protocol, respectively.

4.2 Group Management Extension for the UMTP

The UMTP specification does not address how to obtain multicast group addresses that need to be tunneled. It assumes that any application that needs multicast connection can request tunnel for (group, port) that it wants to use.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     (IPv4) group address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            port               |            command            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 8 Group management protocol extension for the UMTP

The designed group management extension for the UMTP consists of 'join' and 'leave' command for target (group, port). User application should send 'join' command periodically to show its interest for that group address. If the user application sends 'leave' command or master TE does not receive 'join' command for some period of time, the master TE will leave the corresponding (group, port). Fig. 8 shows the protocol format for the group management.

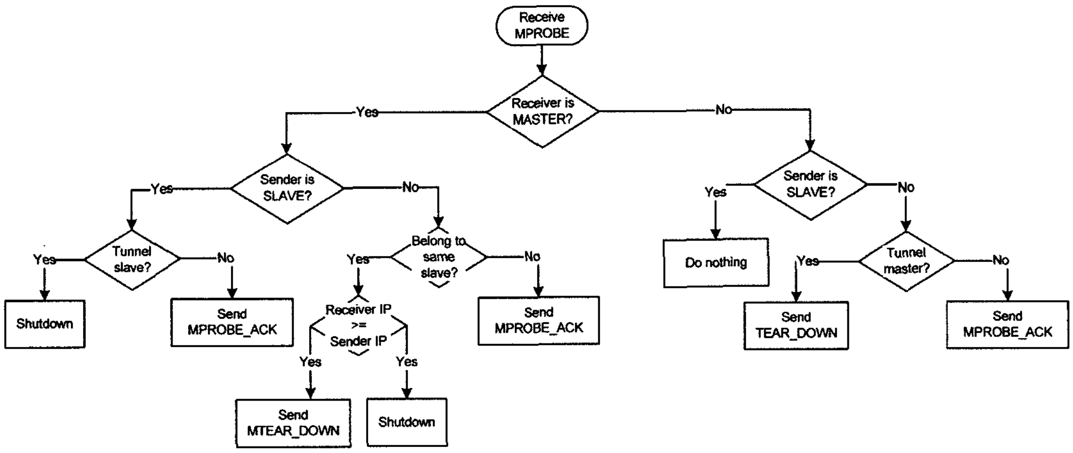


Fig. 6 Flow diagram for receiving MPROBE

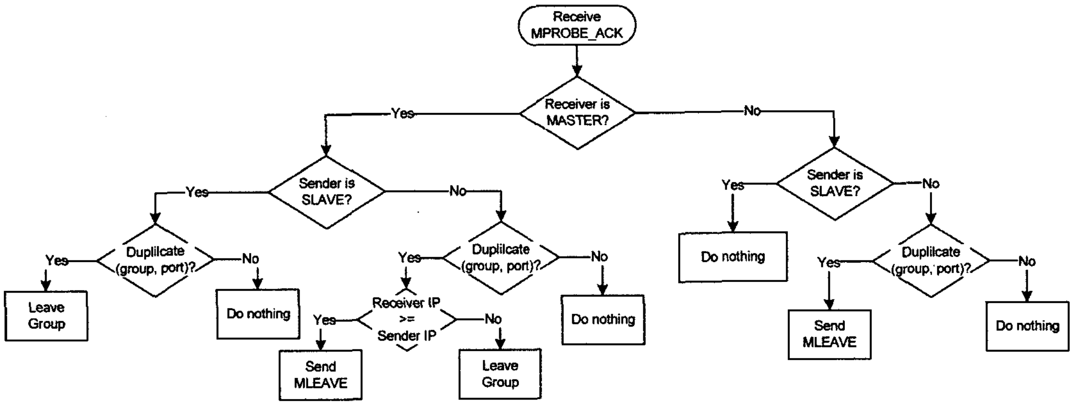


Fig. 7 Flow diagram for receiving MPROBE_ACK

5. Implementation and Verification

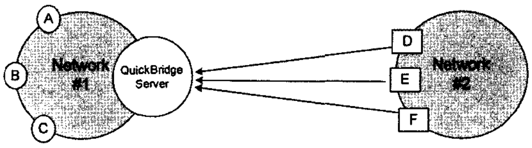
5.1 Implementation

We implement a multicast connectivity solution for the AG, AG Connector. The AG Connector is an interface for the AG nodes to utilize the UMTP. By implementing AG Connector as a shared application, we can easily make the AG nodes to utilize AG Connector. This AG Connector shared application implements group management protocol to communicate with the UMTP master. The prototype of AG Connector is implemented with C language. To make it run on both Unix-based system and windows systems, we use the UCL common multimedia library and pthread. It operates by managing several threads. The 'unicast receiver thread' communicates with the AG Connector shared applications. The 'MPROBE thread' implements the MPROBE protocol. There are number of 'multicast group thread' that receives data from each multicast group. More information about latest status of the AG Connector can be found at [7].

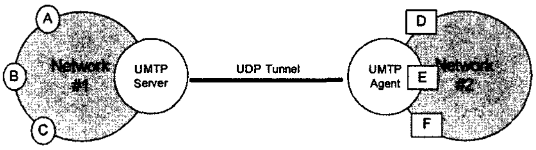
5.2 Verification

We implement the AG Connector so that it can substitute the existing QuickBridge connectivity solution. By implementing the UMTP, we aim a system that achieves multicast connectivity with less bandwidth/performance consumption than the QuickBridge. Thus, we will compare the network bandwidth consumption of AG Connector with QuickBridge and show the effectiveness of AG Connector.

We prepare two networks without multicast connection from each other. Fig. 9 shows the network configuration considered. Each network has three AG users in it. Only the users in network #1 send video and audio stream and the users in network #2 only try to receive that. To make the result clear, we make total stream bigger by sending five video streams (A: 3 videos, B: 1 video, C: 1 video). For the first experiment, we run a QuickBridge server in network #1 and users in the network #2 try to receive streams from the server. The users connect the server one by one with 3 minutes' time interval from each other to see the network usage variation clearly. Fig. 10(a) clearly shows the bandwidth variation according to the connection of each user.

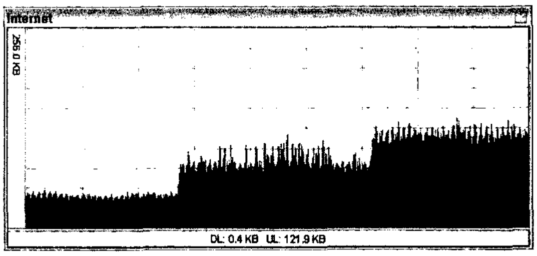


(a) QuickBridge

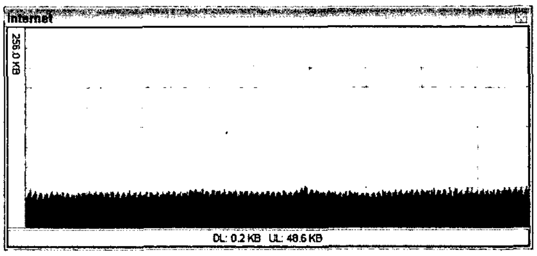


(b) UMTP

Fig. 9 Experiment environments



(a) QuickBridge



(b) UMTP

Fig. 10 Network bandwidth usage

Next, we check the network bandwidth consumption of the AG Connector. We use the machine that we ran QuickBridge as the UMTP server to make this experiment fair. In network #2, user E runs the UMTP agent and create tunnel with the UMTP server. Users in network #2 connect venue one by one with 3 minutes' time interval like the first experiment. Fig. 10(b) shows no additional network bandwidth consumption for additional user. With this experiment, we can verify the effectiveness of AG Connector.

To verify the operation of MPROBE protocol, we setup two nodes in a multicast network and connect these nodes with AG Connector. As a first experiment, we measure the network bandwidth usage of UMTP Agent with MPROBE protocol operating. And for second experiment we measure network bandwidth usage without MPROBE protocol. Fig. 11 shows the result.

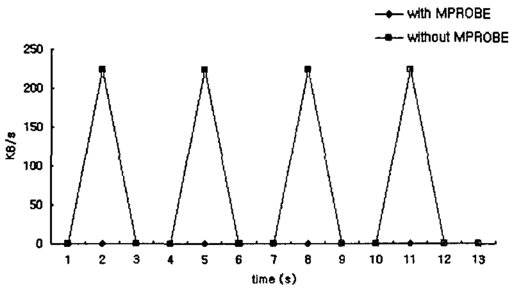


Fig. 11 Effect of MPROBE protocol

We send only 10 bytes packet to a shared multicast address. However, without MPROBE protocol, UMTP Agent receives 224.2 KB per second. On the other hand, with MPROBE, AG Connector detects the multicast reachability to UMTP Server in 2 seconds and stops operation without receiving any data traffic from UMTP Server. We send same size of packet every 3 seconds, and we can observe the same result for every execution.

6. Related work

Multicast is realized through the creation and maintenance of forwarding trees connecting sources and receivers in a multicast group. Participating multicast routers exchange group membership information and use multicast routing protocols (e.g., DVMRP, MOSPF, CBT, PIM-DM, PIM-SM, and PIM-SSM) to establish multicast trees to deliver data. By using the tree delivery structure, IP multicast improves network efficiency and scales to large group size. Despite its bandwidth efficiency, IP multicast usually suffers from a number of deployment issues.

Application layer multicast (ALM) has recently gained increasing popularity in the multicast community. In this type of multicast architecture, group membership, multicast delivery structure construction, and data forwarding are solely controlled by participating end hosts, thus it does not require the support of intermediate nodes. ALMs either organize end users into a mesh and establish multicast trees on top of this mesh (e.g., End System Multicast [8], NICE, ALMI [9]), or directly construct a multicast tree (e.g., Yoid). However, the lack of knowledge about underlying network topology usually results in performance penalty compared with IP multicast. Furthermore, it typically requires a large amount of control overhead to maintain group membership and multicast delivery structures among the end users and to monitor network conditions.

Alternatively, in overlay multicast (OM), multicast functionalities are supported by specially deployed intermediate proxies, which cooperatively construct a 'backbone overlay' infrastructure and establish multicast trees among themselves for data delivery. Outside the backbone overlay, the proxies can deliver data packets to end hosts via multicast or unicast. Recently, many works have been conducted, such as Scattercast [10], Overcast [11], RMX [12], AMCast, and OMNI [13]. It gains knowledge about the network topology by using proxies. End users can form clusters around closest proxies without complicated measurement techniques and clustering algorithms. Therefore, it can easily improve multicast efficiency and reduce resource usage. Unlike application layer multicast, where each overlay only supports one group and/or one application, backbone overlay can support multiple groups and applications simultaneously, and the control overhead for probing and maintaining the backbone is not affected by the number of co-existing groups supported. Consequently, when there are a large number of groups, OM is expected to out-perform ALM in terms of control overhead. Nevertheless, these benefits are achieved at the cost of deploying and maintaining overlay proxies.

Recently, there are some works (e.g., HMTP [14], RMCP [15], XOM [16]) to provide end-to-end service by combining ALM and IP multicast. This hybrid multicast approach dynamically maps ALM path to underlying IP multicast path where IP multicast is available to optimize the performance. The AG Connector also takes the hybrid multicast approach and takes advantages of it. By using IP multicast, this approach utilizes existing multicast infrastructure support to improve both end-user performance and bandwidth efficiency. On the other hand, hybrid multicast approach inevitably induces possible multicast looping problem. HMTP gives a solution for this problem by simply assuming multicast islands. In the AG Connector, as a solution for real Internet environment, we try to solve the problem without any assumption and propose MPROBE protocol to detect and resolve multicast loops.

7. Conclusion

This paper presents a new multicast connectivity solution for the AG multi-party collaboration environments. The proposed UMTP-based solution effectively takes advantage of several UMTP features: UDP tunneling, data encapsulation and re-multicast to local network. By extending the UMTP, we give a solution for multicast data loop problem of the UMTP. However, current prototype implementation only demonstrates the preliminary result of the proposed solution. We should provide a way for the master TE find the location of slave TE easily. By connecting UMTP servers, we can broaden the multicast-reachable area. Also, we should address the one-directional multicast problem. With IPv6 support, AG Connector can bridge IPv4 multicast with IPv6 multicast during the IPv4 to IPv6 transition period. Especially for firewall problem, it can be a good traversal solution for both IPv4 and IPv6 networks.

References

[1] ANL Futures Laboratory, Access Grid Toolkit (version 3.0.2), http://www.accessgrid.org/.
[2] MCS Futures Lab. Argonne National Laboratory, "Bridge server design," Oct. 2003.
[3] A. Ganjam, "Design and experience with supporting NAT and firewall in end system multicast," Master Thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh PA, May 2003.
[4] R. Finlayson, "The UDP multicast tunneling protocol," Internet Draft, IETF, November 2003.
[5] RCBridge, http://if.anu.edu.au/SW/rcbridge.html.
[6] MCS Futures Lab., Argonne National Laboratory, "Programmer's Manual-Shared Applications," Jan. 2004.
[7] Networked Media Lab., "AG Connector," http://ace.nm.gist.ac.kr/AG Connector/.
[8] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in Proc. ACM SIGMETRICS, June 2000.
[9] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An application level multicast infrastructure," in Proc. 3rd Usenix Symposium on Internet Technologies & Systems, March 2001.
[10] Y. Chawathe, "Scattercast: An architecture for internet broadcast distribution as an infrastructure service," Ph.D. Thesis, University of California, Berkeley, CA, Dec. 2000.
[11] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. OToole, "Overcast: Reliable multicasting with an overlay network," in Proc. the 4th Symposium on Operating Systems Design and Implementation, Oct. 2000.
[12] Y. Chawathe, S. McCanne, and E. A. Brewer, "RMX: Reliable multicast for heterogeneous networks," in Proc. IEEE Conference on Computer Communications (INFOCOM'2000), Mar. 2000.
[13] S. Banerjee, "Construction of an efficient overlay multicast infrastructure for real-time applications," in Proc. IEEE Conference on Computer Communications (INFOCOM'2003), Apr. 2003.
[14] B. Zhang, S. Jamin, and L. Zhang, "Universal IP multicast delivery," Elsevier Computer Networks, vol. 50, Issue 6, Apr. 2006.
[15] ITU-T Recommendation X.603. "Information technology-Relayed Multicast Protocol: Framework," Apr. 2004.
[16] D. M. Moen, J. M. Pullen, and F. Zhao, "Implementation of Host-based Overlay Multicast to Support of Web Based Services for RT-DVS," in Proc. 8th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT 2004), Oct. 2004.

김 남 곤
2004년 2월 전남대학교 정보통신공학부 학사 졸업. 2006년 2월 광주과학기술원 정보통신공학과 석사 졸업. 2006년 3월~현재 광주과학기술원 정보기전공학부 박사과정. 관심분야는 Collaborative environment, Overlay multicast

김 종 원

1987년 서울대학교 제어계측공학과 학사
1989년 서울대학교 제어계측공학과 석사
1994년 서울대학교 제어계측공학과 박사
1994년 3월~1999년 7월 공주대학교 전자공학과 조교수. 1997년 8월~2001년 7월 University of Southern California 연구 조교수. 1999년 12월~2000년 7월 Technology Consultant for VProtect Systems Inc. 2000년 7월~2001년 6월 Technology Consultant for Southern California Division of InterVideo Inc. 2001년 9월~현재 광주과학기술원 정보기전공학부 부교수. 관심분야는 Networked Media Systems and Protocols focusing "Reliable and Flexible Delivery for Integrated Media over Wired/Wireless Networks" (네트워크미디어: http://nm.gist.ac.kr)