논문 2005-42CI-2-2

안정적인 화상회의 시스템을 위한 알고리즘

(An Algorithm for Stable Video Conference System)

이 문 구*

(Lee, Moon-Ku)

요 약

기존의 화상회의 시스템은 화상회의 참석자 수가 n으로 증가함에 따라 대역폭과 메모리가 n^2 요구된다. 그리고 또한, 음성 데이터의 전송도 참석자 수가 증가하면 그에 따른 트래픽 증가와 참석자의 발언권에 대한 문제를 갖는다. 본 논문에서는 화상 데이터를 위한 서버 측 버퍼링 기법과 발언권자의 증가에 따른 트래픽 증가 등의 문제를 해결하기위한 침묵탐지기법을 이용한 원격화상회의 알고리즘을 제안한다. 화상 버퍼링 알고리즘은 서버에서 다른 클라이언트로 브로드 캐스팅하는 기법이 아니라, 클라이언트로부터 압축된 비디오데이터를 할당된 버퍼에 전송 받는 서버측 버퍼링 기법과 클라이언트에서 다른 참석자들의 비디오데이터를 자신의 대역폭과 네트워크전송속도에 맞게 얻어가는 클라이언트 인덱스처리방법을 이용한 알고리즘이다. 침묵탐지기법을 이용한 음성전송 알고리즘은 다수의 참석자 중에서 말하지 않는 것으로 탐지된 음성데이터는 서버로 전송하지 않도록 하는 방법이며, 채널관리 알고리즘은 우선권이 있는 참석자에게 발언권을 할당하는 방법이다. 제안한 알고리즘을 이용한 원격 화상회의 시스템은 참석자의 수와 관계없이 제한된 메모리와 대역폭 그리고 네트워크전송속도에서 20프레임 이상, 평균 30ms의 안정적인 화상데이터와 음성데이터전송이 이루어졌다.

Abstract

In previous video conference system, when the number of participants in video conference increases by n, the bandwidth and memory of n2 is required. And also, it brings about increase in traffic and problem of a say during a conference in aspect of transmission of voice data. In this paper, we propose an algorithm of remote video conference using silence detection algorithm to resolve the questions such as buffering method of video data in server and heavy traffic detection algorithm to the increase in participants. Video data buffering algorithm is not a method of broadcasting to other client in the server, but this algorithm uses two other methods; the buffering method of receiving compressed video data from clients and the indexing method for acquiring the video data of other participants in clients according to clients' bandwidth and network transmission speed. We apply a voice transmission algorithm and a channel management algorithm to the remote video conference system. The method used in the voice transmission algorithm is a silence detection algorithm which does not send silent participants' voice data to the server. The channel management algorithm is a method allocating a say to the participants who have priority. In consideration of average 20 frames and 30ms regardless of a number of participants, we can safely conclude that the transmission of video and voice data is stable.

Keyword : voice transmission algorithms: 화상전송알고리즘, video data buffering algorithm: 화상 버퍼링 알고리즘 silence detection algorithm : 침묵 탐지 알고리즘, channel management algorithm : 채널관리 알고리즘

I. 서 론

초고속 네트워크와 대역폭의 발전에 따라 웹 컨퍼런스의 도입, 화상회의 시스템의 도입 그리고 음성 회의 시스템의 도입 등이 가속화되고 있다. 그러나 기존의 원격화상회의 시스템에서는 화상처리를 하기 위해서 다음과 같은 문제를 고민해야 한다. 예를 들면, 네트워크 대역폭 절감 문제, 네트워크 속도와 다른 참석자들 간의 프레임 속도 처리 문제, 참석자 수 증가에 따른 기하급수적인 대역폭 문제와 메모리 사용량 문제 등이다[8].

* 정회원, 김포대학 IT 학부 인터넷정보과
(Div. of IT, Dept. of Internet Information , Kimpo College)
접수일자: 2004년11월23일, 수정완료일: 2005년2월27일

본 논문에서는 화상 버퍼링과 침묵탐지기법을 이용한 원격화상회의 알고리즘을 제안한다[1]. 화상 버퍼링 알고리즘은 서버에서 다른 클라이언트로 브로드 캐스팅하는 기존의 비디오 데이터 전송 기법과는 정반대로, 서버 측에서 클라이언트로부터 압축한 비디오데이터를 할당된 버퍼에 전송 받고 클라이언트에서 다른 참석자들의 비디오데이터를 자신의 대역폭과 네트워크전송속도에 맞게 얻어가는 개념이다. 이러한 개념을 이용한 화상 버퍼링 알고리즘은 사용자수가 아무리 증가하여도 서버 측에서는 각 클라이언트를 위하여 할당된 버퍼를 관리하면 되므로 메모리 증가의 문제가 해결된다. 그리고 서버에서 동시에 모든 클라이언트들에게 같은 고속의 처리속도로 비디오 데이터를 전송하기 위하여 트래픽의 증가와 네트워크 전송속도 향상의 문제를 고려하여야 하지만, 클라이언트가 서버 측의 버퍼로부터 자신의 네트워크 전송속도에 맞추어 비디오 데이터를 얻어가므로 트래픽과 네트워크처리속도 등은 문제시 되지 않는다. 그리고 침묵탐지기법을 이용한 음성전송 알고리즘은 다수의 참석자 중에서 말하지 않는 것으로 탐지된 무언의 음성데이터는 서버로 전송하지 않도록 하는 방법이므로, 참가자의 수가 증가함에 따른 트래픽 증가의 문제가 해결되며, 채널관리 알고리즘은 우선권이 있는 참석자에게 발언권을 할당하는 방법으로 여러 참석자에게 발언권을 이리저리 돌려야 하는 문제점을 해결한다. 제안한 알고리즘을 이용한 원격 화상회의 시스템은 참석자의 수가 증가해도 메모리와 대역폭 문제가 해결되었다. 제안한 알고리즘을 이용한 원격 화상회의 시스템은 제한된 메모리와 대역폭 그리고 네트워크전송속도에서도 20프레임 이상, 평균 30ms의 안정적인 전송이 이루어졌다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 기존의 화상회의 시스템의 문제점을 설명하고, Ⅲ장 에서는 본 논문에서 제안한 화상전송 알고리즘과 음성전송 및 채널관리 알고리즘을 소개한다. Ⅳ장에서는 제안하는 알고리즘의 구현을 위한 제반 기술에 대하여 설명하였으며, Ⅴ장에서는 안정적인 프레임전송에 대한 실험분석 내용을 기술하고, 마지막으로 Ⅵ장에서는 결론을 기술한다.

Ⅱ. 화상회의 시스템

1. 화상 데이터

MPEG4 화상 데이터는 다음과 같이 두 종류의 데이터로 구분한다[2].

1) I 프레임 : 상호 관계에 따른 이미지 정보를 가지지 않은 데이터이므로 JPEG 데이터와 같이 하나의 이미지만으로 복원이 가능한 데이터이다. 그러나 JPEG과 마찬가지로 데이터 량이 크다.

2) P 프레임 : 바로 전의 이미지를 바탕으로 하여 변화된 이미지의 정보만을 가진 데이터이다. 그렇기 때문에 상대적으로 I 프레임에 비하여 크기가 작다. 만약에, 변화된 상태가 크다면, 데이터 량도 마찬가지로 커지게 된다.

2. 화상회의 시스템의 문제점

기존의 원격화상회의 시스템에서는 화상처리를 하기 위해서 다음과 같은 문제점을 갖는다. 네트워크 속도가 틀린 참석자들 간의 프레임 속도 처리 문제, 참석자 수 증가에 따른 기하급수적인 대역폭 문제 그리고 참석자 수의 증가에 따른 기하급수적인 메모리 사용량 문제 등이다. 이러한 문제점이 생성되는 이유는 일반적인 화상회의 시스템에서는 화상 데이터를 보낼 때 일정한 프레임 율에 따라 화상 데이터를 전송하고 다른 참석자들 모두에게 전송하는 브로드 케스팅(broadcasting) 방식을 사용하고 있기 때문이다. 이러한 방식은 클라이언트(회의 참석자)의 요청에 따라 서버 측에서 모든 클라이언트에게 응답하는 방식으로 알고리즘이 간단하고, 구현이 용이하다. 그러나 이러한 방식을 사용하게 되면, 참석자들의 수가 많아질수록 기하급수적으로 데이터 량이 급증하게 된다. 참석자들의 수를 n 이라고 표현한다면, 한명의 참석자의 데이터를 다른 모든 참석자들에게 전송하게 되므로 n * (n-1) 이 된다. 그리고 자신의 데이터를 서버로 전송하게 되므로 실제로는, n ^ 2 이 된다. n 이 10 이라고 한다면, 100명분의 데이터가 돌아다닌다는 결론에 도달한다. n 이 100이면 10000명분의 엄청난 트래픽이 발생한다. 또한, 어떤 참석자가 자신의 화상 데이터를 전송했으면, 다른 참석자들에게 전송해야 하므로, 버퍼링을 해야 하는데, 문제는, 모든 참석자들의 네트워크 회선 속도가 틀리므로, 참석자들마다 별도의 버퍼를 마련해야 한다는 것이다. 그러므로 메모리 문제도 n ^ 2가 되어 네트워크 트래픽과 마찬가지로 엄청난 메모리 량을 소비하게 된다. 이와 같은 알고리즘을 채택한 화상회의 시스템은 대량의 참석자들을 감당하기 힘든 구조를 갖고 있다.

음성 데이터도 참석자들 모두에게 전송하는 방식을 채택하고 있으므로 화상 데이터와 마찬가지로, n ^ 2

에 대한 트래픽을 가진다. 그러므로 특수한 알고리즘을 가지지 않으면, 엄청난 트래픽으로 많은 사용자들을 수용할 수가 없다. 이 때문에 많은 기존의 화상회의 시스템은 발언권을 3 ~ 4개로 제한시켜 이에 대한 문제를 해결하려고 했다. 왜냐하면 4사람 이상이 발언권을 같고 동시에 대화를 한다면 대화내용을 바르게 인식할 수 없기 때문이다. 그리고 발언권을 3~4개로 제한하더라도 이러한 방법은 상당히 큰 불편함을 초래한다. 만약, 10명의 참석자가 있다면, 발언권을 이리 저리 돌리는 사태가 발생하여, 화상회의의 장점을 무색케 만든다.

Ⅲ. 화상 및 음성 데이터 전송기술

1. 화상전송 알고리즘

본 논문에서는 기존의 화상회의 시스템처럼 클라이언트에서 비디오 데이터를 서버로 전송(push)하고, 서버에서 다른 클라이언트로 브로드 캐스팅하는 개념과는 정반대로, 서버 측에서 클라이언트로부터 비디오 데이터를 전송받고, 클라이언트에서 다른 참석자들의 비디오 데이터를 자신의 대역폭과 네트워크 전송속도에 맞게 얻어가는 개념의 알고리즘을 채택하고 있다[3],[4].

이 알고리즘에서는 참석자별로 자신의 비디오 데이터 큐만을 가지고 있다. 그러므로 메모리 증가는 순차적으로 증가하므로, n ^ 2 이 아닌 n 형태로 변형되어 메모리 문제가 해결된다.

```
// 클라이언트로부터 비디오 데이터를 수신하기 위한 스케쥴러
void VideoRequestScheduler(VCSConfClassInfo *pClass)
{
    VCSConfVideoData            *pVideoData;
    VCSConfUserInfo             *pUserInfo;

// 회의에 참석한 모든 참석자들을 체크한다.
    foreach (pUserInfo from pClass)
    {
        // 비디오요청이 셋팅되어 있는가 ?
        if (pUserInfo->IsVideoRequested())
        {
        // 사용자로부터 갱신된 비디오 데이타를 얻는다.
            pVideoData =
GetUpdatedVideoDataFromUser(pUserInfo,
 pUserInfo->IsIFrameRequested());

// IFrame 비디오 데이터이면 기존의 버퍼의 내용을 모두 소각한다.
            if (pVideoData->IsIFrame())
            pUserInfo->RemoveVideoDataAll();

            // 비디오 큐에 쌓는다.
            pUserInfo->AddVideoData(*pVideoData);
        }
    }
}
```

그림 1. 화상전송 알고리즘
Fig. 1. Voice data transmission algorithm.

```
// 클라이언트의 비디오 데이터 수신에 대한 서버 측의 구현
void        OnGetVideoDataList(Array<VCSConfVideoDataInfo>
&dataList)
{
    VCSConfVideoDataInfo        *pVideoData;
    VCSConfClassInfo            *pClass;
    VCSConfUserInfo  *pUserInfo, *pSelectedUser;

// 현재 세션의 회의 정보를 얻는다.
    pClass  = GetCurrentSessionClass();

// 현재 세션의 사용자 정보를 얻는다.
    pUserInfo= GetCurrentSessionUserInfo();

// 내가 선택한 사용자들로부터 갱신된 비디오 데이타를 얻는다.
foreach                (pSelectedUser                from
pUserInfo->m_mapSelectedVideo)
    {
        // 다음 비디오 데이타
        pVideoData    =
pSelectedUser->FindNextVideoData();

        // 갱신된 데이터가 있으면..
        if (pVideoData != NULL)
        {
            // 갱신된 비디오 데이터 목록에 추가한다.
            dataList.Add(*pVideoData);
        }
        // 갱신된 데이터가 없으면 비디오를 요청한다.
        else
        {
    if (pSelectedUser->GetVideoDataSize() <
 g_pEnvInfo->m_nDefaultVideoIFramePeriod)
            {
// 비디오 큐의 크기가 IFrame 주기에 도달하지 못했으면 PFrame
을 요청한다.
 pSelectedUser->SetVideoRequest(PFrame);
            }
            else
            {
// 비디오 큐의 크기가 IFrame 주기에 도달했으므로 IFrame을 요청
한다.
    pSelectedUser->SetVideoRequest(IFrame);
            }
        }
    }
}
```

그림 2. 화상데이터 버퍼링 알고리즘
Fig. 2. Voice data buffering algorithm.

다른 참석자들 별로 버퍼를 가지는 대신에 자신의 큐에서 마지막으로 가져간 비디오 데이터에 대한 인덱스를 가지게 된다. 버퍼 큐가 10 이라고 하고, 마지막으로 가져간 비디오 데이터의 인덱스가 5라고 한다면, 다음에 비디오 데이터를 요청하면, 인덱스6의 비디오 데이터를 리턴하게 된다. 만약에, 다음 인덱스의 비디오 데이터가 존재하지 않는다면, 그때, 서버가 해당 참석자의 비디오 데이터를 얻어오는 개념이다[4].

제안하는 화상전송 알고리즘에서는 비디오 전송을 서버가 제어함으로써 n ^ 2 에 대한 트래픽이 절대로 발생할 수 없도록 하면서, 불필요한 데이터 전송이 일어나지 않도록 하는 특징을 가지고 있다. 버퍼 소각 시기는 I Frame이 발생했을 때이다. 그러므로 버퍼가 모두 찼을 경우에 클라이언트에게 요청하는 프레임은 반드시 I Frame이어야 한다.

2. 음성전송 알고리즘

음성전송 알고리즘은 무제한의 발언권 할당이 가능하도록 침묵탐지기법과 채널관리기법으로 설계하였으며, 발언권은 기본 디폴트로 4명으로 되어 있으며, 조절이 가능하도록 하였다[5].

침묵탐지기술이란 다수의 참석자가 있지만 이중에서 참석자가 말하지 않는 것으로 판단된 음성 데이터는 서버로 전송되지 않는 방식으로, 이에 따라 엄청난 트래픽이 줄어 들 수 있게 된다. 채널관리 알고리즘은 많은 사람이 동시에 발언을 하지만, 우선권이 있는 사람에게

```
// 클라이언트의 오디오 데이터 전송에 대한 서버 측의 구현
void OnSendAudioData(VCSConfAudioDataInfo &dataInfo)
{
        int                     nChannel;
        BOOL                    bCanSpeak;
        VCSConfClassInfo        *pClass;
        VCSConfUserInfo         *pUserInfo;
// 현재 세션의 회의 정보를 얻는다.
        pClass          = GetCurrentSessionClass();

// 현재 세션의 사용자 정보를 얻는다.
        pUserInfo       = GetCurrentSessionUserInfo();

// 오디오 채널 점유를 시도한다.
        NChannel        =       OccupyAudioChannel(pClass,
pUserInfo);

        // 채널 점유에 성공했으면..
        if (nChannel >= 0)
        {
           // 오디오 데이터를 회의 참석자들에게 브로드캐스팅 한
다.
           pClass->BroadcastData(dataInfo);
        }
           }
```

그림 3. 음선 전송 알고리즘
Fig. 3. Voice Transmission Algorithm.

```
// 채널 관리 함수 (위의 음성 전송 알고리즘 모듈에서 사용한다.)
int     OccupyAudioChannel(VCSConfClassInfo     *pClass,
VCSConfUserInfo *pUserInfo)
{
   int                  i, nSize;
   int                  nMaxDiffTime, nDiffTime;
   int                  nMaxDiffChannel;
   VCSConfAudioChannel          *pChannel;
   nSize  = pClass->m_arrChannel.GetSize();
// 점유할 수 있는 채널이 존재하는지 찾는다.
   foreach (i=0; i<nSize; i++)
   {
      pChannel      = pClass->m_arrChannel.GetAt(i);
// 이미 채널을 소유하고 있다.
   if (pChannel->m_pUserInfo == pUserInfo)
{
// 최종 소유한 시간을 갱신한다.
   pChannel->m_dwLastOwnedTime              = dwCurrentTime;
   return (i);
   }
// 현재 시간과 최종적으로 소유한 시간과의 차이를 구한다.
   nDiffTime=abs(GetCurrentTime()-
 pChannel->m_dwLastOwnedTime);
// 가장 차이가 큰 참석자를 구한다.
   if (nMaxDiffTime < nDiffTime)
   {
   pMaxDiffChannel= i;nMaxDiffTime= nDiffTime;
      }
   }
// 현재 채널수가 최대수보다 작으면 추가한다.
      if (nSize < MAX_CHANNEL_NUM)
      {
     VCSConfAudioChannel          channelInfo;
     channelInfo.m_pUserInfo      = pUserInfo;
     channelInfo.m_dwLastOwnedTime          =
GetCurrentTime();
// 채널을 추가하고 추가된 채널 인덱스를 리턴한다.
      return (m_arrChannel.Add(channelInfo));
      }
// 최소 점유 시간을 넘으면 채널을 뺏는다.
      if (nMaxDiffTime >= MIN_CHANNEL_OWNED_TIME)
      {
         pChannel    =
pChannel->m_arrChannel.GetAt(nMaxDiffChannel);
   pChannel->m_pUserInfo= pUserInfo;
   pChannel->m_dwLastOwnedTime = GetCurrentTime();
      return (nMaxDiffChannel);
      }
      return (-1);
}
```

그림 4. 채널관리 알고리즘
Fig. 4. Channel Management Algorithm.

채널을 할당하여, 할당되지 않은 사용자의 음성은 무시하는 방법이다[6],[8].

IV. 구현을 위한 기반 기술

1. 통신기반 기술

1) 네트워크 상에서 어떤 대상과 메시지를 주고받기 위해서 상호간에 통신에 대한 규약으로 통신 플랫폼(Communication Platform : 이하 CP)을 구성하였다. CP는 실제 네트워크 통신에 기반 한 모든 애플리케이션에서 사용될 수 있는 하위 기반 플랫폼으로써 작게는 통신 라이브러리이지만 크게는 CPToolkit에서 지향하는 개발플랫폼의 핵심인 미들웨어로서 사용된다. CP는 실제적으로 TCP/IP를 사용하지만, 다른 통신 프로토콜에서도 마찬가지로 적용될 수 있는 인터페이스를 제공한다[7].

2) 특정화된 통신 방법은 통신 프로그래밍에서 상당한 제약을 가져올 수 있다. 그러한 단점을 극복하기 위하여 핸들러 호출뿐만 아니라 구조화된 스트림 통신으로 상호 메시지 통신자체는 복잡하지만 구현은 매우 쉽게 처리가 가능하다. 상대방에게 호출되는 메시지는 두 가지 형태가 있을 수 있다. 하나는 핸들러라고 하는 프로시저를 호출하기 위한 핸들러 메시지가 있고, 또 하나는 일반적인 스트림 데이터 통신을 위한 스트림 메시지가 있다. 두 형태의 호출 차이점은 핸들러 메시지는 핸들러 프로시저로 호출되지만 스트림 메시지는 이벤트 프로시저에서 처리되는 차이점이 있다

3) 단일 세션 및 단일 포트를 이용하여 멀티 통신이 가능한 구조로 설계하였다. 세션(session)과 채널(channel)이라는 개념을 도입하여 하나의 세션을 이용하여 멀티 통신이 가능한 개념이다. 여러 포트를 이용하여 오픈 한다고 하더라도 애플리케이션에 관계없이 프로그래밍이 가능하도록 설계하였다.

4) 이 기종간과의 데이터 통신의 투명성을 제공하도록 설계하였다. CPER(CP Encoding Rule)의 도입으로 이 기종 간에 Big Endian, Little Endian 의 타입과는 관계없이 상호간에 데이터 교환이 변형 없이 전달될 수 있도록 처리가 가능하도록 하였다[5].

5) 최적화된 통신 구조로 고속의 데이터 통신 지원

멀티쓰레드 구조 및 WINSOCK2로 이루어진 통신 메커니즘은 최고의 통신을 수행하도록 한다. 낮은 CPU 점유율로 대량의 데이터를 전송 가능케 하는 것이다.

Connection Listener, Data Read, Data Write, Channel Processing 등이 별도의 쓰레딩으로 이루어져 수행시간의 빈틈없이 최고의 성능을 발휘하고, 복수개의 CPU가 탑재된 멀티프로세싱 환경에서는 보다 높은 수행 성능을 보장할 수 있도록 한다[2],[3].

2. 클라이언트 서버 동작

CP는 전형적인 클라이언트 서버 (C/S) 프로그래밍 방식을 지원한다. 모든 C/S 시스템의 골격은 서비스 받고자 하는 클라이언트가 서버에게 요청(request)를 하면 서버는 그에 대한 처리를 하고 클라이언트에게 응답(response)를 하는 것을 말한다. 또한 , CP는 멀티프로세싱을 매우 강력하게 지원한다. 이것은 다중 쓰레드를 매우 효율적으로 최대한 이용한다는 것이며, 다중 CPU 시스템에서 더욱더 빠르게 동작될 수 있다. 멀티쓰레딩 프로그래밍은 일반적으로 싱글쓰레딩 프로그래밍에 비하여 상대적으로 훨씬 더 까다롭기 때문이다. (예: 동기화 문제 및 교착상태 문제)

CP는 클라이언트로부터 받은 호출을 서버는 쓰레드로 생성하여 병렬적으로 수행될 수 있기 때문에 서버는 핸들러 루틴에 대하여 복수개의 요청에 대한 처리(Mutex, Semaphore)에 대하여 대비해야 한다. 예를 들어, 어떤 핸들러가 현재 수행되고 있고, 메모리 등의 자원에 대하여 처리하고 있는 시점에서 다른 핸들러가 실행되어 동시에 자원을 접근하는 과정에서 발생할 수 있는 에러를 막기 위하여 동기화 처리를 수행해야만 하는 것이다.

클라이언트 측에서 CP 프로그래밍은 대부분은 동기호출을 사용하기 때문에 비동기 호출을 사용하지 않는 한 멀티쓰레딩에 대해서 고민할 필요는 없다. 그러나 그렇다하더라도 CP에서 지원하는 고급 기능들의 경우에는 비동기 호출을 사용해야만 가능한 것들이므로 동기화 객체를 사용하는 방법에 대해서 확실하게 아는 것이 중요하다고 할 수 있다.

CP를 사용하기 위해서는 우선적으로 SERVICE를 생성(CreateService)시켜야 한다. 그리고 상대방으로부터 연결을 받기 위해서는 SERVICE를 시작(StartService) 시켜야한다. 이것은 SERVICE를 백그라운드(쓰레드)로 동작시키며, 상대방으로부터의 연결을 받기 위해서 포트(TCP, UDP)를 열고 수신대기(listen) 한다는 것을 말한다. 물론, 클라이언트들 중에 대부분의 경우에는 포트를 개방할 필요가 없을 것이다. 그러므로 SERVICE를 시작시키지 않아도 서버에 호출을 할 수 있다는 것이다. 서비스의 시작은 연결을 수신 받을 필요가 있을 때

에만 작동시키는 것이다[5],[6].

클라이언트로부터 연결이 되면 새로운 Worker Thread를 생성하여 세션이 맺어지게 되고 인증 과정을 거치게 된다. 세션은 물리적인 데이터 송수신 및 프로토콜을 처리한다. 각각의 세션에서는 프로토콜 해석기가 있어서 수신 받은 데이터를 디코딩하고 프로토콜을 해석한다. 세션이 성립되고 사용이 완료된 후 세션을 닫게, 그러면 상대방도 마찬가지로 닫히게 된다. 사용자가 자신의 세션을 닫을 때는 DESTROYED 이벤트만 발생하는 반면에 상대방은 CLOSED이벤트가 먼저 발생하여 상대방으로 연결이 해제되었다는 것을 알려 준다. CP에서의 세션 관리는 신뢰성 있는 연결을 지향하기 때문에 외부로부터의 무작위 데이터 호출에 대한 견고성과 좀비 세션이 발생하지 않도록 자체적인 일정한 스케쥴링에 의해 제거될 수 있도록 한다.

실제의 데이터 통신은 물리적인 세션을 가상적인 채널이라는 개념을 통하여 나누어서 수행되는데 이것은 하나의 세션을 다시 무수히 많은 별도의 세션과 같이 이용할 수 있게 함으로써 매우 융통성 있게 사용할 수 있도록 하기 위함이다. 채널은 서비스 및 세션과 마찬가지로 별도의 쓰레드에서 동작되며 순차적인 호출을 보장한다. 그러므로 여러 개의 오퍼레이션을 순서대로 호출해서 서버에서도 마찬가지로 클라이언트에서 호출된 순서대로 서비스되고 반환되기 위해서는 반드시 특정한 한 개의 채널에만 호출해야 한다. 비록 같은 세션이라 할지라도 다른 채널에 호출하게 되면 병렬적으로 처리되어 어느 채널이 먼저 수행될 지 전혀 알 수가 없게 된다. 이렇게 사용이 완료된 채널은 닫아주어야 한다. 채널도하나의 가상 연결이므로 세션과 마찬가지로 한쪽에서 닫으면 상대방도 마찬가지로 닫히며, 연결을 해제하는 쪽에서는 채널 DESTROYED 이벤트만 발생하지만 상대방은 CLOSED 이벤트가 먼저 발생되어 상대방으로부터 연결이 해제된 것을 알 수 있다.

[그림 5]는 제안하는 웹 기반의 화상회의 시스템의 모듈 구성도 이다. VCS(Video-Conference System는 본 논문에서 제안하는 웹 기반의 원격화상회의 시스템의 약칭으로 이하 VCS로 표기) 클라이언트 모듈에서 VCSWeb는 웹기반서비스 제공을 위한 모듈로 확장자는 *.ocx 이고, VCSConf는 클라이언트 실행파일로 확장자는 *.exe이다. 그리고 VCSAgent 역시 실행파일(*.exe)이며 인증절차와 플러그 앤 서버(Plug & Server)로 연동하며, 확장이 가능하도록 설계하였다. VCS서버 모듈에서 VCSServerMain은 플러그 앤 클라이언트(Plug & Client)로 자동 착탈식 방식으로 설계하였고, 클러스터링 기술이 아닌 로드 벨런싱((load balancing) 기능이 내장되도록 하였다.

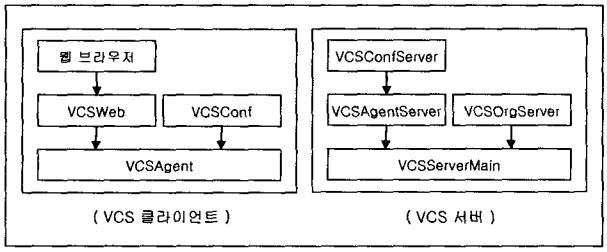

그림 5. 화상회의 시스템 모듈 구성도
Fig. 5. Video Conference System Module.

3. 제반 설정사항

본 절은 웹기반 화상회의 시스템 구현을 위한 알고리즘을 작성하기에 필요한 각종 설정사항들을 기술한다.

○ DefaultAudioFormat

(타입: DWORD, 최소: 0, 최대: 1, 디폴트: 1)
화상회의를 진행할 때의 음성 포맷을 설정할 수 있다. 0으로 설정하면 데이터 량이 절반으로 줄어들지만, 그만큼 음성의 질이 떨어지므로 디폴트를 1로 설정하였다.

0 = 16비트 8KHz (초당 약 6KB)
1 = 16비트16KHz (초당 약 12KB)

○ DefaultAudioBufferSize

(타입:DWORD, 최소:1, 최대:16,디폴트:4, 단위:20ms)
클라이언트 수신 측에서의 오디오 재생 버퍼 크기를 설정한다. 1씩 증가마다 120ms의 음성지연이 추가적으로 발생하므로, 숫자가 작을수록 음성 지연이 줄어든다. 그러므로 디폴트 음성 지연 시간(송신자의 음성이 서버를 거쳐 수신측에 도달한 후 재생되는 시간)은 120ms * 4 = 0.48초로 계산된다.

○ DefaultVideoQuality

(타입: DWORD, 최소: 1, 최대: 31, 디폴트: 4)
화상 데이타 MPEG 압축률을 지정한다. 숫자가 높을수록 높은 압축률을 의미한다. 압축률이 높게 되면 압축된 데이터의 크기를 줄어들게 해주므로 데이터 전송량이 줄어들게 되지만, 그만큼 압축되지 않은 화면보다 질적으로 많이 떨어지게 된다.

○ DefaultVideoFrameRate

(타입:DWORD, 최소:33, 최대:10000, 디폴트:66,단

위:ms)
초당 화상프레임율을 지정한다. 디폴트 66은 (1000ms / 66ms) = 15 프레임이며, 최소 33은 (1000ms / 33ms) = 30 프레임이다. 최대 10000은 (1,000ms / 10,000ms) = 0.1 프레임 이므로 초당 10 프레임을 가리킨다. 프레임이 증가하면 화상 처리에 따른 PC부하 및 네트워크 대역폭도 그만큼 요구되어 지므로, 사이트의 네트워크 및 고객의 요구 사항에 따라 적절한 최대 프레임 수로 지정하면 된다.

○ DefaultVideoIFramePeriod
(타입: DWORD, 최소: 1, 최대: 30, 디폴트: 10)
화상데이타의 I프레임과 P프레임 간격을 지정한다. I프레임보다 P프레임 데이터 크기가 작으므로 I프레임 발생 횟수를 줄이면 그만큼 데이터 전송량을 줄이는 효과가 있다. 그러나 이 간격이 커지면, 빠른 대역폭에 있는 사용자와 느린 대역폭에 있는 사용자와의 프레임 지연 갭이 생길 수 있다. 빠른 대역폭의 사용자는 거의 최신의 프레임을 받을 확률이 높지만, 느린 대역폭의 사용자는 다음 I프레임이 발생하기 전까지는 빠른 대역폭의 사용자가 이미 받은 P프레임을 받을 확률이 높을 것이기 때문이다.

○ StreamContentLength
(타입: DWORD, 최소: 128, 최대: 16384, 디폴트: 8192)
데이터 전송시의 IDLE 데이터(음성을 제외한 화상, 판서 등의 데이터)의 현재 전송 크기를 지정한다. 이 값이 커지면 그만큼 한번에 많은 데이터를 전송하는 것이므로 빨라지게 된다. 그러나 그만큼 음성 데이터 전송 간격이 넓어질 수 있게 되어 음성 데이터의 품질에 문제가 생길 수 있다. 이 값은 음성 전송 상태(지연 유무, 끊김 유무 등)에 따라 자동적으로 최소 및 최대 크기의 범위 내에서 변할 수 있다. 이것은 음성 품질이 나빠지면 값이 작아지고, 일정 시간동안 음성 품질이 좋으면 값이 다시 커지게 된다.

○ StreamMinContentLength
(타입: DWORD, 최소: 128, 최대:16384, 디폴트: 128)
데이터 전송시의 IDLE 데이터의 최소 전송 크기를 지정한다. StreamContentLength는 이 값보다 작아질 수 없다.

○ StreamMaxContentLength
(타입: DWORD, 최소: 128, 최대: 16384, 디폴트: 16384)
데이터 전송시의 IDLE 데이터의 최대 전송 크기를 지정한다. StreamContentLength는 이 값보다 커질 수 없다.

○ WriteIntervalTime
(타입: DWORD, 최소:0, 최대:100, 디폴트:10, 단위:ms)
데이터 전송시의 IDLE 데이터의 전송 간격을 지정한다. 이 값이 작으면 IDLE 데이터를 많이 전송할 것이므로 음성 데이터의 전송 기회가 그만큼 줄어들게 되어 음성 데이터의 품질에 영향을 미칠 수 있다.
초당 최대IDLE 데이타의 전송량
=StreamContentLength*(1000ms/WriteIntervalTime)

○ StreamContentAccelRate
(타입:STRING, 최소:1.0, 최대:5.0, 디폴트:1.5, 단위: 배)
데이터 전송시의 IDLE 데이터 가속율을 지정한다. 이것은 일정시간(StreamAccelCheckTime) 동안 IDLE의 지속(음성 데이터의 비전송)시 IDLE 데이터량을 전송시키는 것이다. 이것은 음성 품질에 최대한 영향을 주지 않으면서 IDLE 데이터 전송 량을 늘리기 위한 방법이다.

○ StreamAccelCheckTime
(타입: DWORD, 최소: 0, 최대: 제한없음, 디폴트: 360, 단위: ms)
데이터 전송시의 StreamContentAccelRate를 적용하기 위한 IDLE 상태 지속시간을 지정한다. 지정된 시간동안 실시간(음성) 데이터가 전송되지 않고 IDLE 상태가 지속되는 시간을 가리키는 것이다. 그러므로 디폴트 360은 (StreamContentAccelRate가 1.5일때) 360ms 동안 IDLE 지속시 IDLE 전송 데이터양을 1.5배로 늘릴 수 있다는 뜻이다[9].

V. 실험 분석

제안하는 화상전송 알고리즘은 서버가 클라이언트로부터 비디오데이터를 얻고, 다른 클라이언트들이 비디오데이터를 서버 측 버퍼로부터 각자의 처리속도에 따라 자료를 가져가면서 가져간 비디오데이터의 인덱스 기록을 남기는 방식이다. 이는 웹 화상회의 시스템이 활성화 되어도 안정적으로 운영이 되도록 구현하였으며, 이러한 알고리즘은 네트워크 전송문제, 사용자수의

증가에 따른 트래픽 증가문제, 사용자 수의 증가에 따른 메모리 사용량과 대역폭증가 문제 등을 해결하였다. 그리고 버퍼 소각 시기는 I Frame이 발생했을 때로 설정하여 버퍼 오버플로우 현상을 사전에 방지 하였다.

또한 제안하는 음성전송 알고리즘은 침묵탐지기술로서 이는 다수의 참석자 중에서 말하지 않는 것으로 판단된 음성 데이터는 서버로 전송되지 않는 방식으로, 이에 따라 엄청난 트래픽이 줄어 들 수 있게 되었으며, 채널 관리 알고리즘은 많은 사람이 동시에 발언을 하지만, 우선권이 있는 사람에게 채널을 할당하여, 할당되지 않은 사용자의 음성은 무시하는 것이다. 이러한 음성전송 알고리즘의 구현으로 무제한 참석자가 있는 경우 발생하는 트래픽 문제를 해결하였다. [그림 6]과 [그림 7]은 제안하는 화상전송 알고리즘과 음성전송알고리즘을 이용하여, 원격에서 통신하도록 구현하고 테스트한 결과이다. 회의 참석자가 4명 과 6명의 경우 모두 프레임수가 평균 10~13 프레임정도이고, 전송속도는 평균 142ms이다. 실험결과에서 참석자수의 증가에 따른 메모리사용량의 문제, 대역폭문제 그리고 네트워크 전송속도문제는 해결되었다. 그러나 트래픽 지연과 프레임

그림 6. 참석자 : 4 평균 10~13 프레임전송
Fig. 6. client : 4 average 10~13 frame trans.

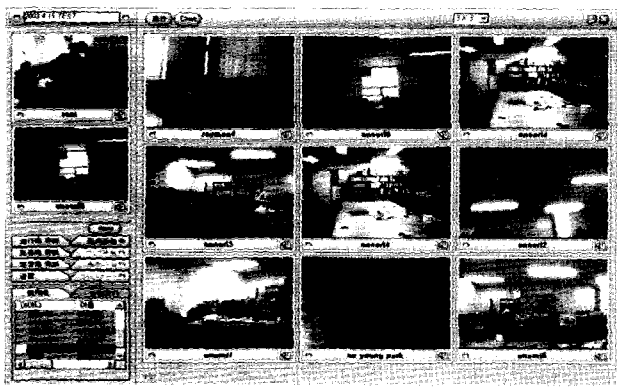

그림 7. 참석자 : 6 평균 10~13 프레임전송
Fig. 7. client : 6 average 10~13 frame trans.

수가 평균 15프레임 이하정도이고, 전송속도 역시 평균 140ms이다.

[그림 8]은 제안하는 화상전송알고리즘과 음성전송알고리즘을 이용한 경우 메모리, 대역폭, 클라이언트와 서버간의 네트워크 전송속도 등의 문제는 해결되었으나, 프레임 수와 관련하여 해상도문제 그리고 전송속도의 향상 등을 위하여 수정 및 보완하였다. 화상전송을 위한 비디오데이터의 압축시점을 클라이언트가 서버에 전송할 때 하는 것이 아니라 사전이 백그라운드로 압축을 해놓고 요청이 들어오면 바로 전송할 수 있도록 함으로써 안정적인 프레임전송이 가능하도록 하였으며, 원격의 참석자와의 프레임 전송 속도를 향상하기 위하여 I 프레임의 압축률을 향상시킨 결과 지속적으로 평균 프레임수가 20이상, 30ms로서 전송속도가 4~5배 이상 향상되었으며, 선명한 해상도를 보여주었다.

[그림 9]는 전송속도와 프레임수와의 관계를 실험한

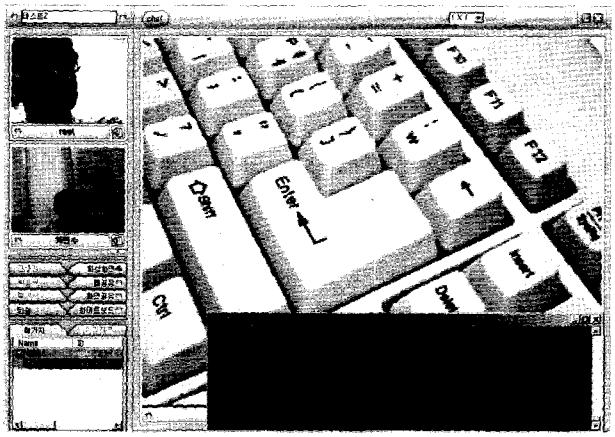

그림 8. 참석자 : 2 평균 20 프레임전송
Fig. 8. participants : 2 average 20 frame trans.

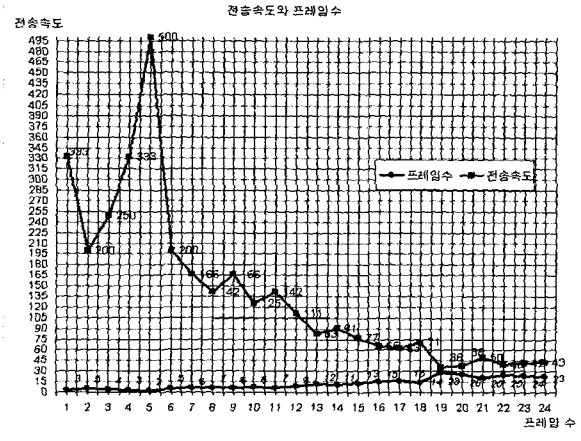

그림 9. 전송속도와 프레임수와의 관계
Fig. 9. Transmission speed and frame number.

결과이다. 표에서 프레임수가 5프레임 미만인 경우는 일반적인 화상회의방식으로 전송한 경우로서 전송속도가 평균 400~500ms 정도이다. 이 경우는 프레임수가 로컬(local)에서 5미만이며, 원격인 경우와 참가자의 수가 증가한다면, 그에 따른 대역폭을 확장하거나 하드웨어를 업그레이드하거나 혹은 화상회의용 카메라의 성능을 최상으로 보상하여야 하는 등의 방법으로 개선하지 않는다면 원격화상회의 시스템으로서의 여러 가지 문제에 직면한다. 프레임수가 15프레임 미만인 경우는 ([그림 6]참조) 제안하는 알고리즘의 인덱스 버퍼링 방식을 적용하여 최종 수정하기 이전단계로서 해상도문제와, 참가자수의 증가에 따른 메모리문제, 대역폭문제 등은 해결되었으나 전송속도가 142ms로서 아직 상당히 느린 결과였다. [그림 7]의 경우 역시 제안하는 알고리즘방식으로 구현하였으나 원격의 참가자수가 6명으로 증가된 경우로서, 대역폭과 메모리문제는 없었으나 프레임 수가 평균 15프레임미만으로 나타났다. 제안하는 알고리즘을 이용한 경우 대역폭, 메모리문제 등은 참가자수와는 무관하였다. [그림 8]은 제안하는 알고리즘에서 클라이언트가 서버에 전송할 때 압축하는 것이 아니라 클라이언트에서 사전에 백그라운드로 압축을 해놓고 전송하도록 하였으며, 데이터 량이 많은 I 프레임의 압축률을 높였더니 속도가 30ms 정도이고 평균 20 프레임의 안정된 전송이 이루어졌다.

VI. 결 론

기존의 원격화상회의 시스템에서는 회의 참석자 모두에게 데이터를 전송하는 브로드 케스팅 방식을 사용하므로 참석자 수의 증가에 따른 기하급수적인 대역폭 문제 와 메모리 사용량 문제에 접하게 된다. 뿐만 아니라 네트워크 속도가 틀린 참석자들 간의 프레임 속도 처리 문제와 네트워크전송속도 문제 등을 갖게 된다.

본 논문에서는 이러한 문제를 해결하기 위하여 화상 버퍼링 기법과 침묵탐지 기법을 이용한 화상전송 및 음성전송 알고리즘을 제안하였다. 화상 버퍼링 알고리즘은 서버에서 다른 클라이언트로 브로드 캐스팅하는 기법과는 반대로, 서버 측에서 클라이언트로부터 압축한 비디오데이터를 할당된 버퍼에 전송 받고 클라이언트에서 다른 참석자들의 비디오데이터를 자신의 대역폭과 네트워크전송속도에 맞게 얻어가는 개념이다. 이러한 개념을 이용한 화상전송알고리즘은 기존의 화상전송방법으로 인하여 발생할 수 있는 대역폭의 확장과 클라이언트의 증가에 따른 메모리 사용량의 증가, 하드웨어의 지속적인 업그레이드 등의 문제가 서버 측의 버퍼링 기법과 클라이언트의 요청과 응답에 대한 인덱스 처리 방법의 알고리즘을 이용하여 해결되었다. 침묵탐지기법을 이용한 음성전송 알고리즘은 다수의 참석자 중에서 말하지 않는 것으로 탐지된 음성데이터는 서버로 전송하지 않도록 하는 방법이며, 채널관리 알고리즘은 우선권이 있는 참석자에게 발언권을 할당하는 방법이다. 제안한 알고리즘을 이용한 원격 화상회의 시스템은 클라이언트의 수가 증가되는 것과는 상관없이 20 프레임 이상 전송되고 평균 30ms의 속도를 보여 안정적인 전송이 이루어졌다고 말할 수 있다. 무언의 참석자를 탐지하는 침묵탐지알고리즘에서 지연되는 시간을 줄이는 방법에 대한 연구를 지속 하고자 한다.

참 고 문 헌

[1] Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman "The Design and Analysis of Computer Algorithms (Addison-Wesley Series in Computer Science and Information Processing)"Addison Wesley, 01 January, 1974.

[2] David S. Taubman, Michael W. Marcelin, JPEG2000", Book News, Inc. 2000.

[3] Khalid Sayood "Data Compression" Academic Press, 2000.

[4] Keith Jack, "Video Demystified" LLH Technology publishing, 2001.

[5] Jesus Pinto, Kenneth J. Christensen, University of South Florida, "An Algorithm for Playout of Packet Voice Based on Adaptive Adjustment of Talkspurt Silence Periods" 24th Conference Computer Networks, Lowell, Massachusetts, IEEE Computer Society, p.224, October 17 - 20, 1999.

[6] G.Y. Hong, A.C.M. Fong, Massey University, B. Fong, Lucent Technologies in Singapore"QoS Control for Internet Delivery of Video Data" IEEE Computer Society p.0458, April 08 - 10, 2002.

[7] Y. Shibata, N. Seta, S. Shimizu, Dept. of Comput. Sci., Toyo Univ., Kawagoe, Japan "Media synchronization protocols for packet audio-video system on multimedia information networks" IEEE Computer Society HICSS'95,

Hawaii, USA, p.594, January 04 - 07, 1995.

[8] http://www.shimson.com

[9] 김회린, 한국정보통신대학원 공학부, "음성인식 기술 개요 및 향후 과제" 대한전자공학회지,2001,5 v.028, n.005, pp.74-80

[10] 이정철, 울산대학교 컴퓨터·정보통신공학부, "음성합성 기술 개요 및 향후 과제", 대한전자공학회지, 2002,12 v.029, n.012, pp.1491-1497

저 자 소 개

이 문 구(정회원)
1984년 숭실대학교 전자계산학 (학사)
1993년 이화여자대학교 대학원 전산교육학 (석사)
2000년 숭실대학교 대학원 컴퓨터시스템 (공학 박사)
2000년 3월~현재 김포대학 컴퓨터계열 조교수
<주관심분야 : 인터넷 보안, 암호화 알고리즘, 전자상거래 보안, 침입차단시스템, 위험분석 및 관리>