IOCP와 Multi-Thread를 이용한 RTSP Media Server 개발

김수진*, 김익형*, 권장우**
*팜캐스트 기술, **동명정보대학교 컴퓨터공학과

Development of RTSP Media Server Using IOCP & Multi-Thread

SuJin Kim*, IgHyeong Kim*, Jang-Woo Kwon**
*PalmCast Technology
**Dept. of Computer Engineering, Tongmyong Univ. of Information Technology

요 약

본 논문에서는 RTSP 프로토콜을 제어하기 위한 서버 시스템을 IOCP 기반의 Multi-Thread 기법을 이용하여 구현하는 방법을 소개한다. 다수의 클라이언트에 대한 응답을 Thread로 구성하는 부분에서 Multi-Threading을 이용함으로써 수행 속도를 높이고 Winsock2에서 제공하는 IOCP(I/O Completion Port)를 이용하여 견고하고 확장이 용이한 RTSP(Real Time Streaming Protocol) 스트리밍 서버를 개발하였다.

1. 서론

최근의 네트워크 시스템의 급속한 발전은 다양하고 질 좋은 네트워크 응용 제품을 내놓기에 충분한 환경을 제공하고 있다. 그 중에서도 유무선 온라인 멀티미디어 서비스는 이제 생활의 일부분으로 자리를 잡고 있다. 특히 온라인 게임, 인터넷 방송, 채팅 서비스, 다자간 화상회의 등의 시스템은 이미 다양한 기술 및 서비스 전략을 통하여 일반 사용자들에게 소개되고 있다. 이와 같은 네트워크 서비스에는 운용 시스템 및 네트워크의 부하 문제, 그리고 자원의 효과적인 관리 문제가 뒤따르게 된다. 특히 온라인 게임이나 인터넷 방송과 같은 멀티미디어 서비스는 동시에 수많은 사용자들이 접속하여 이용하기 때문에 네트워크의 부하 및 시스템의 수행속도에 커다란 영향을 준다. 따라서 본 연구에서는 RTSP(Real Time Streaming Protocol) 프로토콜을 이용한 스트리밍 서버를 구현할 때 다수의 클라이언트에 대한 응답을 위한 Thread를 구성해야 하는 부분에서 Thread의 개수를 줄여 수행속도를 높이고, 견고하고 확장 가능한 Daemon Process를 구현한다. Multi-Thread를 이용한 Socket 기반의 서비스를 수행하는 시스템을 개발할 경우, 견고하고(Robust) 확장 가능한(Scalable) 서버 소프트웨어를 구성하기 위한 방법으로 Winsock2에서 제공하는 IOCP(I/O Completion Port)를 이용하여 시스템을 설계할 수 있다. 일반적인 Multi-Thread를 이용한 Socket기반의 서버 소프트웨어는 해당 클라이언트 접속 개수 만큼 서버측에서 Thread가 생성되어야 하는

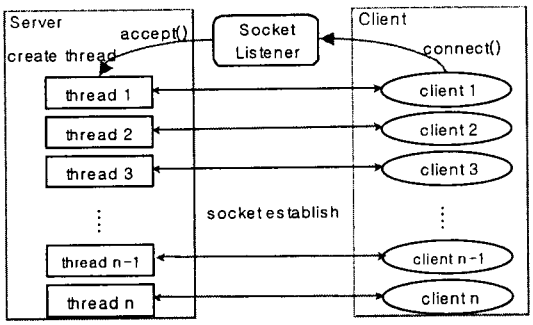
반면, IOCP기반의 Socket을 이용한 서버 소프트웨어를 구성할 경우 서버측의 Thread 수를 하나의 CPU당 64개로 한정시켜 Thread 수를 급격히 줄이고 수행속도를 높이는 형태의 시스템을 구현할 수 있다.

본 논문에서는 RTSP 프로토콜을 처리하기 위한 서버 시스템을 구성하기 위해서 IOCP 기반의 multi-Thread 기법을 이용하여 구현하는 방법을 소개한다.

2. IOCP 기반의 Multi-Thread 설계

2.1 일반적인 Multi-Thread의 구현 방법

쓰레드(Thread)는 다수의 사용자들을 동시에 처리할 수 있는 프로그램이 각각의 사용과 관련하여 가지고 있는 정보들을 말한다[1]. 프로그램의 관점에서 볼 때 쓰레드는 한 명의 개별 사용자 또는 특정한 서비스 요청을 서비스하는데 필요한 정보를 의미한다. 다수의 사용자들이 해당 서비스를 사용하고 있거나 다른 프로그램들로부터 동시에 서비스 요청이 발생할 경우 각각의 사용자나 프로그램들을 위해 쓰레드가 만들어진다. 쓰레드는 프로그램에게 현재 어떤 사용자가 서비스를 받고있는지를 파악하게 함으로써 다른 사용자들을 위하여 재진입 해야 할 것인지의 선택을 할 수 있도록 한다.

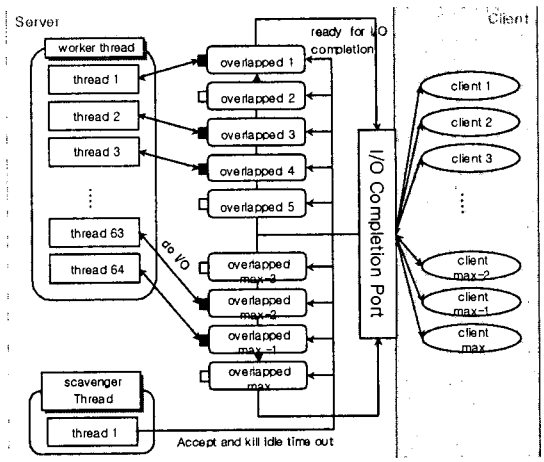


[그림 1] 일반적인 Multi-Thread 구조

그림 1은 일반적인 Socket을 이용한 Multi-Thread의 구조를 나타낸다. 접속하는 클라이언트의 수만큼 서버측에서 Thread를 생성하고 각각의 Thread가 직접 해당 클라이언트의 데이터를 주고 받는 형태를 취한다.

2.2 IOCP 기반의 Multi-Thread 설계

IOCP(I/O Completion Port)는 열려진 File의 I/O에 관련된 event 처리를 수행하기 위한 용도로 개발된 port로 한번에 64개의 서로 다른 비동기 I/O 결과를 처리할 수 있으며, Queue형태로 완료된 I/O 데이터 결과의 신호에 생성하여 주며 Socket과 연동되어 IOCP Socket 서버를 구성한다[2]. 많은 수의 쓰레드 생성으로 인한 오버헤드 문제로 발생하는 확장의 제한성을 해결하기 위한 방안으로 IOCP(I/O Completion Port)를 이용한다. 그림 2는 IOCP 기반의 Multi-Thread 서버 구조를 나타낸다.



[그림 2] IOCP 기반의 Multi-Thread 서버 구조

IOCP를 이용한 Multi-thread 서버의 구성은 Overlapped Plus라는 IOCP를 위한 데이터를 접속할 클라이언트 개수 만큼 미리 서버에서 생성하고 I/O Completion Port의 대기열에 삽입하여 클라이언트가 접속할 때 동시에 64개의 클라이언트 I/O를 획득, 64개의 Worker Thread에서 처리하는 방식이다.
하나의 Worker Thread에서는 하나의 I/O를 처리하며 Overlapped Plus 개체를 통해서 클라이언트와 데이터를 교환한다.

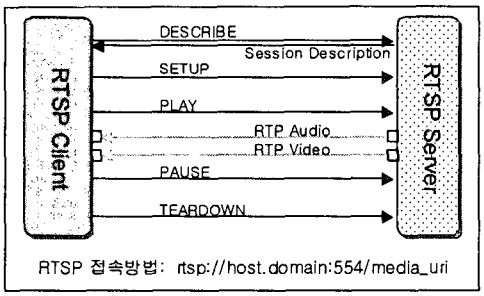
그림 2에서 Overlapped는 비동기형 Input/Output의 정보를 저장하는 공간으로 실제 Socket에서 데이터를 주고 받기 위하여 Socket의 데이터 교환 용도에 맞게 Overlapped Plus라는 확장 형태로 사용하고 실제 Thread에서 이용하기 위하여 IOCPClient Class를 구성하였다. Overlapped는 서버가 생성할 때 최대 접속할 수 있는 클라이언트 개수 만큼 생성하여 이용하며 I/O Buffer의 크기 및 시스템에 따라 생성 개수가 달라진다. Worker Thread는 64개의 입출력을 처리하여 Socket과의 데이터를 생성하고 처리하는 Thread를 말하며, Scavenger Thread는 클라이언트로부터 처음 접속되는 accept()부분에 대한 처리를 담당하고, 접속 후

Idle Time Out이 발생할 경우 Overlapped를 초기화 하는 부분으로 1개의 독립된 Thread로 동작한다.
Client는 네트워크 상에서 접속하는 실제 클라이언트를 가리킨다.

2.3 RTSP (Real Time Streaming Protocol)

RTSP는 인터넷 상에서 멀티미디어 스트리밍 데이터를 제어하는 방법에 대한 표준안이다[3]. RTSP는 H.323과 마찬가지로 멀티미디어 컨텐츠 패킷 포맷을 지정하기 위해서 RTP 프로토콜[4]을 사용한다. 하지만 H.323이 적당한 크기의 그룹간 화상회의를 위해 설계된 데 반해서 RTSP는 대규모 그룹들에게 오디오 및 비디오 데이터를 효과적으로 Broadcast하기 위한 목적으로 설계되었다[3].

그림 3에서 RTSP 프로토콜은 TCP로 접속할 경우 포트번호 554번으로 접속을 하게 되며, 서버측의 IOCP Socket Listener는 554번으로 접속하는 클라이언트의 Socket 세션을 연결하고 명령어를 전달하게 된다.



[그림 3] RTSP(Real Time Streaming Protocol)

RTSP 프로토콜은 Audio 및 Video의 RTP, 멀티캐스트, 유니캐스트 등의 전송 세션을 제어하기 위한 프로토콜로 Options, Describe, Setup, Play, Pause, Record, Teardown, Set_Parameter, Get_Parameter 등의 명령어 Set을 통해서 클라이언트와 서버간 교신한다.

2.4 RTSP Media Server

2.4.1 RTSP 명령어 SET과 IOCP와 관계

클라이언트와 서버간의 명령어 형태는 Request와 Response로 분류할 수 있으며, Request와 Response는 다양한 포맷으로 구성되어 클라이언트와 서버간 교신하게 된다.

그림 4는 ITEF RFC-2326[3]에서 정의한 Request와 Response의 BNF 규약과 실제 구현한 클라이언트와 서버간의 프로토콜 전송 예제이다.

```
Request  = Request-Line          ; Section 6.1
         *(  general-header      ; Section 5
         |   request-header      ; Section 6.2
         |   entity-header )     ; Section 8.1
           CRLF
           [ message-body ]      ; Section 4.3

Response = Status-Line           ; Section 7.1
         *(  general-header      ; Section 5
         |   response-header     ; Section 7.1.2
         |   entity-header )     ; Section 8.1
           CRLF
           [ message-body ]      ; Section 4.3
```

[*] Request Sending:
PLAY rtsp://localhost/video.mov RTSP/1.0
Range: npt=0.0000-
Cseq: 5
User-Agent: PalmCast Network Inspector 1.0
Date: Thu, 17 Oct 2002 01:45:18 GMT
Content-Length: 0
Session: 1034814440

[*] Response Received:
RTSP/1.0 200 OK
Range: npt=0.0000-
Cseq: 5
Server: PalmCast RTP/RTSP Media Server 1.0
Date: Thu, 17 Oct 2002 01:45:18 GMT

[그림 4] 클라이언트와 서버간의 프로토콜 전송 예제

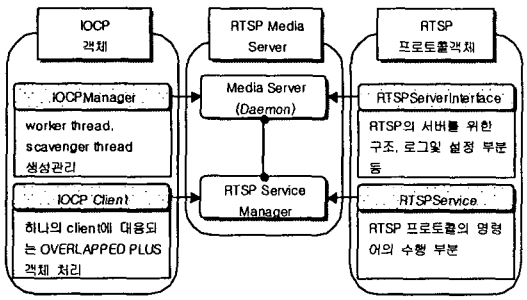
RTSP 클라이언트에서 IOCP기반의 RTSP 서버로 전송되는 Request는 Overlapped Plus 객체와 64개의 Worker Thread를 통해 I/O Completion Port에서 선택되어진 하나의 Overlapped buffer를 하나의 Thread가 처리하게 된다. IOCP의 특성상 클라이언트에 많은 부하가 더해지더라도 64개의 Thread는 균등하게 부하 분산하여 수행된다. IOCP는 부하 분산의 특성을 가지는 반면에, buffer의 크기가 1024Kbytes와 같은 형태로 정해져 있기 때문에 buffer 크기보다 많은 데이터가 입력될 경우에는 처음 처리한 Thread 이후에 다음 Thread에서 남은 자료를 재처리해야 하는 구현상의 어려움을 가진다.

본 연구에서는 위와 같은 문제를 해결하기 위해서 Overlapped Plus에게 상태 전이도와 파서기(Parser)를 장착하여 명령어가 완료되는 지점까지 하나의

RTSP Request 명령어 셋을 추출할 수 있는 구조로 설계, 개발하였다.

상태 전이도는 *Overlapped* Plus가 *I/O Completion port*를 통과할 때의 전이 상태를 기록하는 곳으로 Accepting, Reading, Writing, SendingFile 등으로 분류되며, Reading 부분이 I/O Completion port를 통해서 데이터를 읽었다는 상태를 나타낸다. 파서기는 IOCPClient 클래스를 통해서 하나의 라인이 완성될 경우 해당되는 자료를 RTSP 프로토콜 파서기에 통과시켜 Request의 마지막 라인까지 파싱이 완료되면 실제 RTSP Command에 대응하는 명령어를 서버에서 수행하도록 구성하였다.

2.4.1 IOCP와 RTSP 프로토콜의 결합



[그림 5] RTSP Media Server 구조

그림 5에서 보는 것과 같이 IOCP와 RTSP 프로토콜을 결합하기 위해서 상위에서 확장한 새로운 RTSP Media Server와 RTSP Service Manager를 구성하여 Thread관리와 실제 RTSP 명령 수행을 해결하게 된다. IOCP의 Daemon Process를 위한 구조는 Overlapped를 위한 IOCP Client와 Thread를 관리하기 위한 IOCP Manager로 구분할 수 있다.

RTSP의 Daemon Process를 위한 구조는 RTSP Command 처리를 위한 RTSP Service 부분과, IOCP Manager로부터 확장하여 실제 Daemon Process를 구성하기 위한 RTSP Server Interface로 나눌 수 있다.

3. 구현 및 고찰

IOCP 기반의 Multi-Thread를 이용한 RTSP Media Server의 개발은 Windows XP 환경 하에서 펜티엄-III 700MHz, 메모리 256MB, VC++을 이용하여 구현하였다.

본 연구를 통해 네트워크 서비스상의 시스템 부하 및 수행 속도를 개선하는 효과를 보았으며, IOCP를 이용한 Multi-Thread는 약 3000 Client가 접속할 경우에도 무난한 수행 능력을 보였다.

RTSP 서버를 구성할 경우 RTSP 클라이언트로부터 RTSP 프로토콜을 주고 받는 Thread와 함께 RTP Audio/Video 패킷을 전송하는 부분의 Thread가 동시에 필요하게 되며, 본 연구에서는 RTSP 클라이언트로부터의 접속에서 64개 이상의 RTSP 클라이언트가 접속할 경우 불필요한 Thread 개수를 급격히 줄이는 결과를 얻을 수 있었다.

4. 결론

본 논문에서는 RTSP 프로토콜을 제어하기 위한 미디어 전송 시스템을 IOCP 기반의 Multi-Thread 기법을 이용하여 구현하였으며, Socket기반의 서버 소프트웨어를 개발 할 경우 효과적인 시스템 개발이 가능하다는 것을 보였다. 본 연구는 온라인 게임 및 멀티미디어 컨텐츠 서비스의 서버 소프트웨어를 개발할 경우에 다양하게 접목될 것으로 보인다.

향후의 연구 방향은 RTSP 프로토콜의 완벽한 지원과 POST-PC 계열 시스템을 위한 멀티미디어 전송에 중점 투자할 계획이며, RTSP 프로토콜의 파싱 부분에서 XML 기술을 접목하여 보다 진보된 RTSP서버를 개발할 예정이다.

본 연구는 2002년도 중소기업 기술혁신개발사업의 지원으로 이루어졌습니다.

[참고문헌]

[1] Silberschatz Galvin, "Operating System Concepts", Addison-Wesley Publishing, pp. 111-116, 1994

[2] Writing Windows NT Server Applications in MFC Using I/O Completion Ports, http://msdn.microsoft.com/library/

[3] IETF RFC 2326 RTSP: "Real Time Streaming Protocol", April 1998

[4] IETF RFC 1889 RTP: "A Transport Protocol for Real-Time Application", January 1998

[5] Sockets Over IOCPs Sample, http://mvps.org/win32/network/sockhim.html

[6] Dominik Filip, CThread – A worker Thread Wrapper class, http://www.codeproject.com/threads/cthread.asp