실시간 무선 영상 감시시스템을 위한 Streamer의 설계 및 구현

이진영* · 김홍준** · 이광석***

A Design and Implementation of Streamer for Real-Time Wireless Video Surveillance System

Jin-young Lee* · Heung-jun Kim** · Kwang-Seok Lee***

요 약

네트워크 인프라의 성장과 디지털 영상압축 기술의 발달로 네트워크 카메라 서버를 이용한 실시간 영상감시 시스템의 수요가 증가함에 따라 가정이나 소규모 사무실에 적합한 실용화 수준의 보안 시스템에 대한 요구도 함께 증대되고 있다. 기존의 CCTV를 이용한 실시간 영상감시에 비해 네트워크 카메라 서버를 이용한 영상감시는 많은 이점이 있다.

본 논문에선 실시간 영상감시 시스템의 핵심 모듈로서 JPEG 영상의 수집과 전달을 담당하는 JPEG Streamer의 모델을 설계, 구현하며 JPEG Streamer의 안정성과 효율성을 위해서 쓰레드 풀과 공유메모리를 사용하고 실시간 영상의 품질을 높이기 위해서 더블버퍼링의 개념도 도입하였다. 또한 제시된 Streamer의 모델을 이용하여 개인 휴대단말기(PDA)로 무선 인터넷을 통해 실시간 영상을 전송하는 무선 영상 감시시스템을 제시한다.

ABSTRACT

Recently, the network Infrastructure grows rapidly and the digital image compression technique has made remarkable progress. Therefore, the demand of the real-time image surveillance system which uses a network camera server has been increasing. Network Camera Server has emerged as an attractive alternative to the CCTV for the wireless video surveillance.

In this article, the model of JPEG Streamer for collecting and delivering JPEG image is designed and realized as a key module for the wireless video surveillance system. The thread pool and shared memory have been used to improve the stability and efficiency of the JPEG Streamer. In addition, the concept of double buffering is of much benefit to improve the quality of real-time image. In this article, the wireless video surveillance system by using JPEG Streamer is suggested to send the real-time image through the wireless internet with the personal digital assistance (PDA).

키워드

JPEG Streamer, wireless video surveillance system

* 강남대학교 교양학부

** 진주산업대학교 컴퓨터공학부

*** 진주산업대학교 전자공학과

접수일자 : 2006. 12. 7

Ⅰ. 서 론

1.1. 연구배경

최근 네트워크 인프라의 급성장과 디지털 영상압축 기술의 발달로 기존 실시간 영상감시 솔루션의 중심을 이루던 CCTV 제품군들이 아날로그 영상을 JPEG 포맷으로 압축해서 전송해주는 NCS(Network Camera Server) 제품들로 대체되고 있다[1][2].

더욱이 실시간 영상감시 솔루션에도 애플리케이션들에서처럼 다양한 소프트웨어적인 접근을 통하여 기존 하드웨어 중심의 단순하고 수동적인 감시활동이 아닌 보다 능동적이고 지능적인 감시활동을 요구하고 있는 실정이다. 다중사용자가 다수의 채널을 웹을 통해 감시할 수 있는 기능, 모니터링 중에 소프트웨어로 모션디텍션을 구현하는 기능, 센서 연동에 의한 특정영상의 자동녹화 기능 등이 요구가 대표적인 예이다.

실시간 영상감시의 응용분야는 매우 다양하며 시스템을 구축하기 위해서는 네트워크를 통하여 실시간으로 사용자들에게 JPEG 영상을 전송해주는 영상전송 모듈이 반드시 필요하다. 하지만 아쉽게도 기존 NCS 벤더들이 제공하는 컴포넌트들은 그 성능이나 운용에 많은 제약이 있다. 우선 다수 채널을 감시하는 솔루션을 구성하기가 어렵다는 것이 큰 제약이 된다. 일반적으로 NCS는 최대 4개 채널까지의 영상을 지원한다. 하지만 실시간 영상감시 시스템은 16개 채널 또는 그 이상을 감시의 대상으로 삼는 경우가 많다. 결국 실시간 영상감시 시스템을 구성하는 데는 2개 이상의 NCS를 병렬로 구성하고 이들을 제어하기 위한 새로운 모듈을 개발해야 한다. 또 다른 제약은 Embedded OS를 탑재한 NCS들은 한정된 자원으로 인해 지원하는 동시사용자의 수가 5명을 넘지 못한다는 것이다. 이 또한 실시간 영상감시 시스템이 요구하는 수준의 동시사용자 수를 서비스하기 위해선 NCS의 능력(자원)을 어떤 식으로든 증폭시켜주어야 한다. 그러므로 실시간 영상감시 시스템에 대한 최근의 요구사항들을 만족시키기 위해서는 대안이 필요하다.

1.2. 연구 목적

본 논문에서는 실시간 영상감시 시스템을 구성하는 핵심 컴포넌트 중 하나인 JPEG Streamer의 모델을 제시하고, 설계·구현하고자 한다. 무선망을 이용한 실시간 영상감시 시스템의 개발과정에 JPEG Streamer를 사용함으로써 개발자는 사용자 인터페이스에 집중할 수 있고, 그 결과 전체 시스템 개발을 단순화하고, 생산성을 향상시키며 실시간 영상감시 시스템의 신뢰성을 높이고 기존의 NCS가 갖는 몇몇 제약사항들을 극복할 수 있다[3]. 또한 본 논문에선 무선망을 이용한 실시간 영상감시 시스템을 설계하는 과정에서 시스템이 운영될 서버의 용량계획에 도움을 주고자 네트워크 사용량 추정 모델과 메모리 사용량 추정 모델을 제안한다. 이 자원사용량 추정모델과 JPEG Streamer 모델을 통해 다양한 환경의 실시간 영상감시 시스템을 모의실험해 볼 수 있을 것이며 본 논문에서는 POSIX thread의 API들을 이용하였다[4].

마지막으로 실시간 영상감시 솔루션을 보다 효율적이고 대중적으로 적용하기 위해 구현된 JPEG Streamer을 무선망을 통해 개인 휴대 단말기(PDA)로 수집된 정보를 전송하는 무선 영상 감시 시스템을 제안하고 구현한다. 제안된 실시간 무선 영상 감시 시스템은 기존 시스템이 고정된 장소에서만 화상 정보를 수신받는 단점을 보안하여 무선 인터넷을 통해 이동하면서도 영상 감시를 할 수 있다.

Ⅱ. JPEG Streamer의 설계

2.1. 요구분석

JPEG Streamer의 모델에 대한 요구사항은 그림 1을 보면 쉽게 알 수 있는데 실시간 무선 영상감시 시스템은 영상을 감시하는 카메라, 실시간 영상을 수집하고 분배하는 JPEG Streamer, 그리고 사용자와 JPEG Streamer 사이에서 인터페이스를 제공하는 서버로 구성된다.

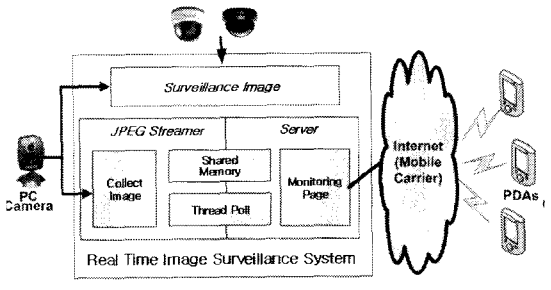

그림 1. 실시간 무선 영상 감시시스템의 구성도
Fig. 1 Main Modules of the Real-Time Wireless Video Surveillance System

JPEG Streamer는 실시간 무선 영상감시 시스템에서 핵심 엔진부로 실시간으로 영상을 수집하고 영상을 요청하는 사용자에게 영상을 제공하는 역할을 수행한다. 따라서, JPEG Streamer는 이를 위해 사용자와의 인터페이스를 제공해야 한다. 또한, 다수의 NCS로부터 수집된 영상들을 다수의 동시 사용자에게 실시간으로 제공해야 하므로 많은 수의 쓰레드가 병렬처리 되어야 한다. 영상을 수집하는 쓰레드와 영상을 가져가는 쓰레드는 공유메모리를 이용하여 영상을 주고받는데 본 논문의 JPEG Streamer는 안정적인 영상감시와 효율적인 운영을 위해 쓰레드 풀을 이용한다.

2.2. JPEG Streamer의 기능

실시간 영상감시 시스템의 핵심 엔진에 해당하는 JPEG Streamer의 기능은 JPEG 영상수집 기능, 수집된 영상을 공유메모리에 저장 기능, 요청시에 공유메모리로부터 영상을 추출하여 네트워크로 전송하는 기능 그리고 안정적인 쓰레드 풀 관리 기능으로 나누어지면 그림 2는 JPEG Streamer를 구성하는 클래스들과 각 클래스들이 수행해야하는 메쏘드들에 대한 Class 다이어그램이다.

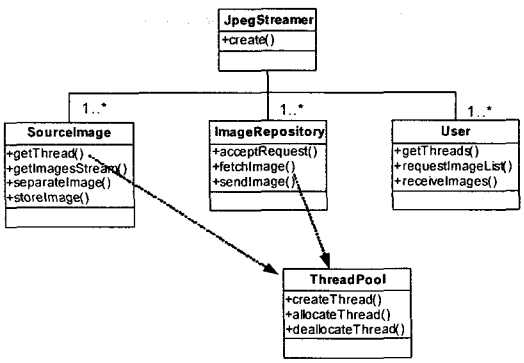

그림 5. JPEG Streamer의 클래스 다이어그램
Fig. 5 Class Diagram of the JPEG Streamer

1) JPEG 영상의 수집

JPEG Streamer는 NCS들로 부터 원시영상을 JPEG 포맷으로 수집한다. 각각의 영상마다 하나의 소켓을 개설하고, 이들 영상들은 동시에 실시간으로 전달되어야 하므로 각 영상의 처리를 위한 소켓들은 멀티쓰레딩으로 이루어진다.

JPEG Streamer의 영상 수집과정은 구체적으로 다음과 같은 순서로 진행된다. 먼저 쓰레드 풀에서 메쏘드 getThread()를 이용하여 수집할 카메라 영상 수만큼의 가용한 쓰레드를 받아야 한다. 만약 가용한 쓰레드가 쓰레드 풀에 없다면 사용자의 요구는 거절된다.

각각의 쓰레드들은 모두 동일한 메쏘드에 의해서 동작한다. 그러므로 한 개의 쓰레드가 어떻게 동작해야 하는지만 정의하면 된다. 이 후는 멀티쓰레딩 API를 호출하면 시스템이 알아서 해줄 것이다[4]. 쓰레드는 제일 먼저 소켓을 생성하고, 이 소켓에 자신이 요청한 영상을 제공할 NCS의 IP Address와 포트번호를 바인딩하여 NCS에 접속한다[9]. 접속이 성공하면 NCS에 영상을 요청하는 CGI를 보내고, NCS는 이 소켓에 HTTP 프로토콜을 이용하여 Server Push 방법으로 JPEG 영상의 스트림을 제공한다. SourceImage 클래스의 getImageStream() 메쏘드가 이 과정을 수행한다.

2) 수집된 영상을 공유메모리에 저장

Server Push 방식은 클라이언트의 한 번의 요청만으로 서버가 지속적으로 동일한 형태의 복수 데이터를 제공하기 위해서 사용되는 방법으로 boundary string을 구분자로 해서 JPEG 이미지들을 분리하고 검증한다. SourceImage 클래스의 seperateImage() 메쏘드가 이일은 수행한다.

storeImage() 메쏘드는 잘려진 JPEG 영상을 공유메모리의 각 영상채널마다 지정된 주소에 저장되며 지속적으로 반복되어진다. 따라서, 공유메모리의 지정된 저장소는 해당 카메라 영상의 가장 최근 스냅샷을 갖게된다. 그림 3의 JPEG Streamer의 영상수집 / 저장 Sequence Diagram은 영상의 수집에서부터 쪼개고 저장하는 과정을 설명하는 순차 다이어그램이다.

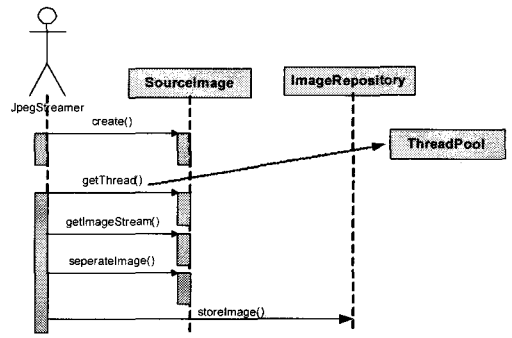

그림 6. JPEG Streamer의 영상수집/저장 Sequence Diagram
Fig. 6 Sequence Diagram of the JPEG Streamer Image Collection/Store

3) JPEG 영상의 전송

사용자는 requestImageList() 메쏘드를 통해 자신이 원하는 영상채널의 목록을 JPEG Streamer에게 요청한다. 한편 JPEG Streamer는 사용자들의 요청을 처리하기 위해 서버소켓을 개설하고 사용자들의 요청을 기다린다[5]. 서버소켓이 사용자의 요청을 받게 되면 별도의 사용자용 소켓을 만들어 사용자가 처음 영상채널의 목록을 요청할 때 할당 받은 쓰레드에 넘겨준다. 사용자가 요청한 영상채널들은 각기 쓰레드 하나씩을 차지하여 멀티쓰레딩으로 동작하게 된다.

requestImageList() 메쏘드는 제일 먼저 JPEG Streamer에 소켓접속을 시도한다. JPEG Streamer는 이를 위해 서버소켓을 개설하고 기다리고 있으며 JPEG Streamer는 연결을 수락하고 사용자에게 클라이언트 소켓을 하나 만들어 준다. 그리고는 getThread() 메쏘드를 이용해 쓰레드 풀에서 가용한 쓰레드 한 개를 얻어와서 이 쓰레드에 방금 만든 클라이언트 소켓과 사용자가 requestImageList()를 통해서 보낸 정보, 즉 원하는 영상채널의 번호를 넘겨준다.

메쏘드 fetchImage()는 넘겨받은 영상채널 번호를 이용해 JPEG Streamer의 공유메모리로부터 해당 영상채널의 최근 JPEG 영상을 버퍼메모리에 복사하고, 이를 sendImage() 메쏘드를 이용해 소켓에 실어 내보낸다. 사용자측 클라이언트는 이를 받아서 화면에 그려주고, 다시 최근 영상을 요청하게 된다. 물론 이 때 TCP 소켓연결은 끊어지지 않은 상태이다. 즉 Client Pull 방식으로 JPEG 영상을 지속적으로 가져온다[6]. 이상의 과정을 순차 다이어그램으로 나타낸 것이 그림 4이다.

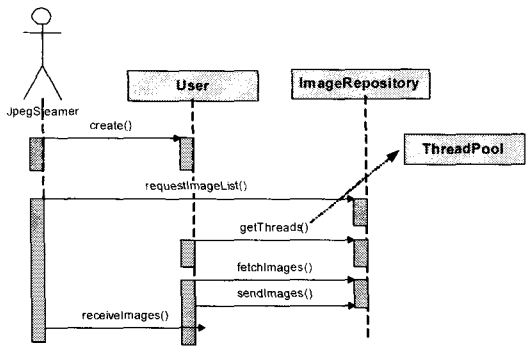

그림 7. JPEG Streamer의 영상요청/수신Sequence Diagram
Fig. 7 Sequence Diagram of the JPEG Streamer Image Request/Send

2.3. 안정적 서비스를 위한 쓰레드 풀 관리

JEPG Streamer는 원시영상을 수집하기 위해 Client Socket을 수집할 영상채널 수만큼 생성하고 관리하며 사용자들의 요청을 처리하기 위해서 Server Socket을 개설하고 대기하고 있다가 사용자의 카메라 영상 요청마다 애 소켓 한 개씩을 accept해 주어야 한다. 또한, 실시간 영상감시 시스템에서는 모든 영상들을 실시간으로 처리하기 위해 소켓들의 입출력이 병렬로 처리되어야 하며, 특정 소켓의 입출력이 다른 소켓의 입출력에 영향을 미치지 않아야 하기 때문에 안정적인 병렬처리를 위해 멀티쓰레딩 모델이 필요하다.

멀티쓰레딩을 이용하여 병렬처리를 구현하기 위해서는 한정된 시스템 자원에 대한 멀티쓰레딩의 정도를 결정해야 한다[4][7].

JPEG Streamer에 의해서 생성되고 실행될 쓰레드의 최대개수를 미리 지정해 주는 것이 안정적인 프로그램의 수행에 도움을 줄 수 있다. 따라서, JPEG Streamer는 최초 시스템으로부터 자신에게 필요한 수만큼의 쓰레드를 위한 일정량의 자원을 미리 할당받아서 각각의 쓰레드들에게 미리 이 자원을 배정하고 상황에 따라 이 쓰레드들을 할당하고 회수하는 방식으로 동작하도록 설계하였다. 그러므로 JPEG Streamer는 영상을 수집하거나 사용자의 요청에 새로운 소켓을 배정할 때마다 쓰레드 풀에서 가용한 쓰레드가 있는지를 먼저 확인하고, 가용한 쓰레드가 있는 경우에만 쓰레드를 배정 받아서 이 쓰레드 안에서 소켓 통신을 하게 된다. 그림 5는 이벤트에 따라 쓰레드의 상태의 변화를 나타내는 상태 다이어그램이다.

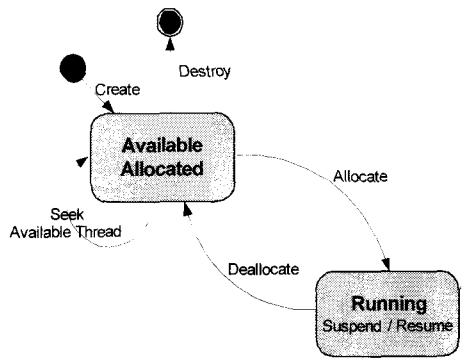

그림 8. 쓰레드 풀의 Statechart Diagram
Fig. 8 Statechart Diagram of the Thread Pool

쓰레드 풀에 있는 쓰레드들은 할당 가능한 상태와 할당된 상태 중 한가지 상태를 갖으며 할당된 상태를 갖는 쓰레드는 다시 실행중인 상태와 할당된 후 정지된 상태로 나누어진다. JPEG Streamer는 기동될 때 환경정보에서 최대 쓰레드 개수를 읽어 와서 그 개수만큼 쓰레드 풀을 조성한다. 초기 상태의 쓰레드들은 모두 할당 가능한 상태이며 getThread()에 의해 할당된 상태로 바뀐다. getThread()가 호출될 때마다 쓰레드 풀에서 가용한 쓰레드는 감소되며 최종적으로 모든 쓰레드가 할당되면 쓰레드 풀은 더 이상의 요청을 받아들이지 않게 된다. 잠시 후 쓰레드들이 작업을 마치면 할당 가능한 상태로 전환된다. 이와 같은 과정에 의해 JPEG Streamer는 시스템의 자원을 안정적으로 사용하고 시스템에 무리를 주지 않으면서 동시 사용자수를 보장할 수 있다.

2.4. 공유 메모리의 사용과 관리

공유 메모리의 사용과 관리에서 고려해야할 중요한 사항은 dirty read 문제와 문제 해결시 성능 저하이다. 따라서, JPEG Streamer에서 사용된 공유 메모리에서도 동일한 문제를 해결하여야 한다.

JPEG Streamer를 구성하는 객체중 하나인 공유 메모리는 SoureceImage 객체의 storeImage() 메쏘드에 의해서 그 내용이 갱신되고, User 객체의 fetchImage() 메쏘드에 의해서 그 내용이 읽혀진다. 공유 메모리를 이용하는 두 객체는 각각 읽기 작업만 하는 객체와 쓰기 작업만을 수행하는 객체이며 storeImage()와 공유 메모리 그리고 fetchImage()는 1:1:N의 관계를 갖는다. 이때 N은 최대 동시 사용자수를 의미하기도 한다.

공유 메모리는 JPEG 영상들을 저장하는 저장소로 JPEG 영상 한 장의 최대크기를 단위 메모리의 크기로 하며 단위 메모리는 각각의 영상 채널에 매핑된다.

SourceImage 객체는 매핑 번호를 이용해서 매핑되는 영상채널의 가장 최근 스냅샷을 지정된 번호의 공유 메모리에 저장하게 된다. 마찬가지로 User 객체가 요청한 카메라 영상들은 각각의 영상채널의 고유번호가 된다. JPEG Streamer는 이 영상채널의 번호에 해당되는 공유 메모리를 찾아서 그 안에 저장되어 있는 JPEG 영상을 사용자 쓰레드의 버퍼에 복사해준다. 쓰레드는 이 버퍼의 내용을 자신에게 배정된 소켓을 통하여 User 객체에게 전송하게 된다.

공유 메모리에 JPEG 영상을 저장하는 쓰레드와 이 공유 메모리에 있는 JPEG 영상을 읽어가는 쓰레드는 병렬로 수행된다. 따라서 최악의 경우 SourceImage 객체가 특정 번호의 공유 메모리에 쓰기 작업을 시작하는 순간에 다수의 User 객체들이 같은 번호의 공유 메모리를 읽으려고 시도할 수도 있다. 이 경우 각각의 User 객체들은 완성되지 않은 JPEG 영상을 가져가게 되거나 메모리 보호 오류가 발생하게 된다.

공유 메모리객체 ImageRepository는 이러한 문제, 즉 쓰레드 간의 메모리에 대한 읽기, 쓰기 작업 중에 발생하는 dirty read문제의 해결을 위해 상호배제를 구현해야 한다[10]. 각각의 공유 메모리는 하나의 쓰기작업을 하는 쓰레드에 의해서 Locked 상태가 되고, 이 동안에는 어떠한 읽기 작업도 허용되지 않는다. 쓰기작업이 완료되면 쓰레드는 공유 메모리의 상태를 UnLocked으로 변경한다. 만약 이 공유 메모리에 대한 Lock이 해제되기를 기다리고 있는 읽기작업 쓰레드가 있었다면 바로 작업을 진행한다. 쓰레드는 읽기작업을 시작하면서 역시 이 공유 메모리를 Locked상태로 변경한다. 그러나, 쓰기작업을 위한 Lock과는 달리 이 공유 메모리에 대해서 같은 읽기작업을 하는 또 다른 쓰레드들은 자유롭게 공유 메모리의 JPEG 영상을 가져갈 수 있다. 그림 6은 상호배제에 대한 상태 다이어그램이다.

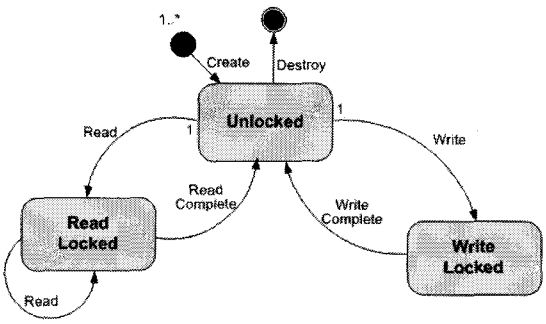

그림 9. ImageRepository의 상태챠트 다이어그램
Fig. 9 Statechart Diagram of the ImageRepository

2.5. 운영환경에 따른 용량계획

JPEG Streamer는 다수의 JPEG 영상을 실시간으로 네트워크를 통해 전송해야하므로 네트워크에 소통될 패킷량이 매우 중요한 요소가 된다.

본 절에서는 JPEG Streamer 모델을 통하여 실시간 영상감시 시스템을 위한 자원의 용량계획의 지표가 되는 두 가지 자원 사용량 추정 모델을 제시한다.

① 네트워크 대역폭 추정 모델

자원들의 용량계획에 영향을 미치는 구성요소는 표 1과 같으며 실시간 무선 영상 감시시스템을 위한 JPEG Streamer의 네트워크 대역폭(NBW : Network Band Width)은 영상채널들의 원시영상을 수집하기 위해서 필요한 대역폭과 사용자들의 요청에 따라 영상을 내보내기 위해 필요한 대역폭의 합으로 다음과 같다.

표 1. 실시간 영상 감시 시스템의 운영에 필요한 매개변수들

Table. 1 Parameters of the Real-Time Wireless Video Surveillance System

항 목	매개변수	비 고
카메라 영상의 수	MaxCamera	감시 대상 영상의 수
JPEG Image의 크기	ImgSize	각 카메라 영상의 한 프레임이 갖는 평균 Image의 크기
매 초당 전송되는 JPEG 프레임 수	FPS	Frame Per Second
동시 사용자 수	MaxUser	
한 화면 최대 분할 수	MaxDiv	보통 16분할

$$NBW = (MaxCamer \times ImgSize \times FPS) + (MaxDiv \times FPS \times ImgSize \times MaxUser)$$

위 수식은 순수 JPEG 영상의 대역폭만을 계산한 것이므로 TCP/IP 통신규약에 따라 추가 발생되는 헤더에 필요한 대역폭을 추가해야 한다. 추가되는 정보는 네트워크의 MTU에 따라 달라진다[9].

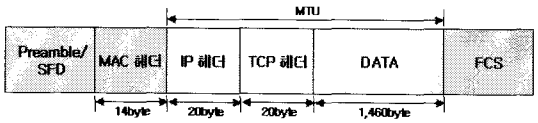

그림 10. JPEG 영상을 전달하기 위한 패킷

Fig. 10 JPEG Image transmission Packet

TCP/IP는 메시지를 통째로 보내지 않고 MTU 단위로 쪼개서 보낸다. 통상 MTU는 1,500byte이고, 이 중에 40byte가 TCP와 IP 헤더로 사용된다. 그러므로 원본 JPEG 영상은 1,460byte 단위로 쪼개지고 각각의 조각에 TCP/IP 헤더가 추가되어 하나의 MTU, 즉 패킷으로 만들어진다. MTU는 다시 14byte의 MAC 헤더가 추가되고, 여기에 이더넷 컨트롤러는 프리앰블과 SOF, FSC 등 12byte의 부가적인 정보를 MTU의 앞뒤에 덧붙여서 네트워크에 내보내게 된다. 이 패킷을 전달받은 상대편 TCP/IP는 이 패킷에 대한 응답패킷으로 66byte를 되돌려 준다[9]. 그러므로 JPEG 영상의 매 1,460byte 마다 132byte의 부가적인 트래픽이 발생한다고 볼 수 있다. 이를 계산하면 다음 수식에서처럼 원래 순수메시지의 약 9%에 해당하는 부가정보가 발생하게 된다.

$$\begin{aligned} \text{부가정보} &= \frac{\text{실제메시지의 전체 바이트수}}{1{,}460} \times 132 \\ &= \frac{132}{1{,}460} \times \text{실제메시지의 전체 바이트수} \\ &= 0.09 \times \text{실제메시지의 전체 바이트수} \end{aligned}$$

원래 메시지 크기의 9%가 부가정보로 추가되므로 부가정보 계수를 1.09로 하면,

$$NBW = 1.09 \times [(MaxCamer \times ImgSize \times FPS) + (MaxDiv \times FPS \times ImgSize \times MaxUser)]$$

이 된다. 수식을 보다 간단히 하기 위해 표 2와 같이 상수로 정의한다. NCS 장비의 특성상 각 장비가 4개 영상채널을 모두 서비스 할 때의 각 채널당 최대 FPS는 5이다. 그리고 사용자가 모니터를 통해서 한 화면에 감시하기에 좋은 최대 분할 수는 16이 적당하다. 그리고 352×240 크기의 리솔루션에서 중간정도의 압축률을 갖는 JPEG 영상 한 장의 평균 크기는 13KB 정도이다.

표 2. 실시간 영상 감시 시스템의 상수값

Table. 2 Constants of the Real-Time Wireless Video Surveillance System

항목	상수 이름	상 수 값
JPEG Image의 크기	ImgSize	13KB
매 초당 JPEG 프레임 수	FPS	5
한 화면 최대 분할 수	MaxDiv	16

그러므로 이들 상수들을 NBW에 적용하면 다음과 같다.

$$\begin{aligned} NBW &= 1.09 \times (MaxCamera \times 13 \times 5 \\ &\quad + 16 \times 5 \times 13 \times MaxUser)KB \\ &= 1.09 \times (65 \times MaxCamera + 1040 \times MaxUser)KB \\ &= (70.85 \times MaxCamera + 1133.6 \times MaxUser)KB \\ &= [(566.8 \times 16) + (9068.8 \times 10)]Kbps \\ &= 99756.8Kbps = 97.41Mbps \end{aligned}$$

계산된 값은 실시간 영상감시 환경에 따라 상수값의 변화로 달라질 수 있으며 패킷충돌에 따른 재전송 등의 부가적인 트래픽은 고려하지 않았다. 그러나, JPEG Streamer의 설계과정에서 도출된 네트워크 대역폭 추정 모델을 통해서 다양한 운영환경의 실시간 영상감시를 위한 시스템의 설계과정에서 이 시스템에 필요한 네트워크 대역폭을 간단하게 구할 수 있으며 구성된 네트워크 환경에서 운영될 수 있는 최대 영상채널의 수(MaxCamera)와 동시사용자 수(MaxUser)를 구할 수 있다.

② 메모리 사용량 추정 모델

메모리 사용량 추정을 위해 사용할 JPEG Streamer의 매개변수는 표 3과 같다.

표 3. 공유 메모리 사용량 추정을 위한 매개변수들
Table. 3 Parameters for the Shared Memory

항 목	매개변수 이름	비고
쓰레드풀의 쓰레드 개수	MaxThreadNum	
공유 메모리 개수	MaxMemNum	
JPEG Image Size	ImgSize	13KB, Max 26KB

쓰레드 풀에서 사용하는 쓰레드의 총개수는 다음과 같다.

$$MaxThreadNum = MaxCamera + MaxUser \times MaxDiv$$

JPEG Streamer는 NCS로부터 카메라 영상을 수집하기 위해 최대 카메라 영상채널 수만큼의 Client Socket을 생성하게 된다. 여기에 사용자들을 위한 소켓들을 더해주면 JPEG Streamer가 운영해야 할 전체 소켓의 개수를 구할 수 있고, 이 소켓 수가 쓰레드 풀을 구성하는 쓰레드의 개수가 된다. 이 때 사용자들을 위한 소켓의 최대값은 [최대사용자수 × 한 화면 최대분할 수]로 구할 수 있다.

메모리 사용량은 쓰레드들이 소켓에 입출력을 하기 위해 사용되는 버퍼메모리와 공유메모리 크기의 합이다.

공유메모리는 JPEG 영상의 평균 크기 11KB 또는 최대 크기 26KB에 해당하는 단위메모리를 최대 카메라 영상수만큼 곱한 크기로 ImageRepository의 크기이다. 따라서, 공유메모리의 크기(SM, Shared Memory)는 다음과 같다.

$$SM = MaxCamera \times 26KB$$

쓰레드 풀이 사용할 버퍼메모리는 상호배제를 구현하는 과정에서 발생하는 영상의 지연 현상이나 끊김 현상을 해결하기 위해 더블버퍼링(Double Buffering)을 적용함으로써 요구되는 메모리이다. 따라서 전체 메모리 사용량은 공유메모리와 버퍼메모리의 합이 되며 표 1과 표 2 그리고 표 3의 매개변수들과 네트워크 대역폭 추정 모델을 구할 때 사용한 상수값을 이용하여 구한 메모리 사용량 추정 모델의 수식은 다음과 같다.

$$\begin{aligned} MaxMem &= \\ &(MaxThreadNum \times MaxCamera) \times 26KB \\ &= (MaxCamera + MaxDiv \times MaxUser \\ &\quad + MaxCamera) \times 26KB \\ &= (2MaxCamera + 16MaxUser) \times 26KB \end{aligned}$$

실제 네트워크 대역폭을 구할 때 사용했던 것처럼 최대 영상채널 수를 16으로, 최대 동시사용자 수를 10명으로 가정하여 메모리 사용량을 추정하면 다음과 같다.

$$\begin{aligned} MaxMem &= (2 \times 16 + 16 \times 10) \times 26KB \\ &= 192 \times 26KB = 4.875MB \end{aligned}$$

III. 제안 모델 구현

JPEG Streamer는 독립적으로 동작하는 완성된 애플리케이션이 아니라 실시간 영상감시 시스템을 구성하는데 필요한 모듈로 실시간 영상감시 시스템을 구축하기 위한 컴포넌트이다. 따라서, JPEG Streamer 컴포넌트를 검증하기 위해 개인 휴대 단말기를 이용한 실시간 영상감시 시스템을 모델로 한다.

3.1. 개인 휴대 단말기 영상감시의 구현

본 논문에서 구현한 JPEG Streamer를 이용해서 개인

휴대 단말기(PDA)를 통해 실시간으로 원격지의 영상을 감시하는 시스템을 구성해 보았다.

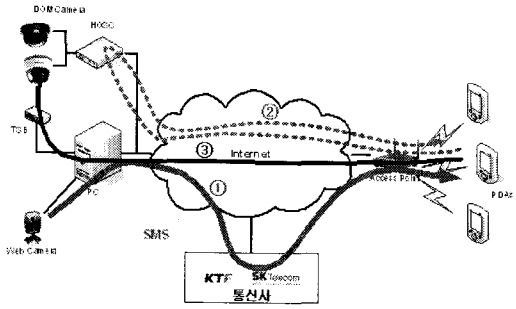

그림 11. 실시간 무선 영상 감시 시스템의 프로세스 구성도
Fig. 11 Main Modules of the Real-Time Wireless Video Surveillance Process

개발 시스템은 보안을 적용한 장소에 설치된 PC에서 Detect된 영상을 문자 메시지와 함께 개인 휴대 단말기로 전송한다.

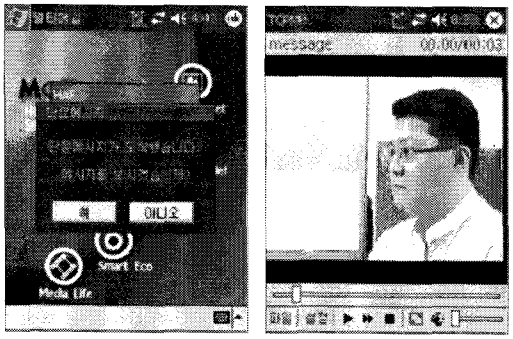

그림 8. PDA에 전달된 메시지와 감시영상 화면
Fig. 8 The transferred message and video image by the Video Surveillance system

다음은 그림 7에서 ③에 해당하는 화면으로 Mobile 장비를 이용하여 원격지에 DOM 카메라를 제어하여 원격지의 내부를 확인한다.

개발된 PDA 소프트웨어는 현재 4개 채널을 동시에 확인할 수 있으며 다양한 보기 기능과 카메라 좌우상하, 확대/축소가 가능하다.

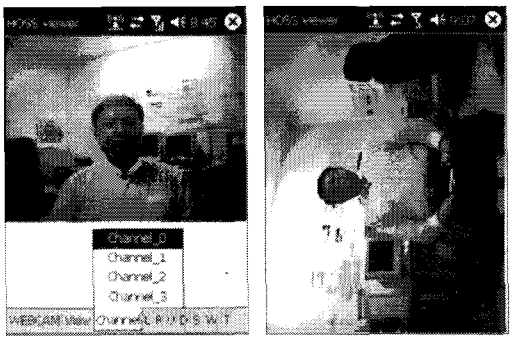

그림 9. PDA를 이용한 다양한 감시영상 화면뷰
Fig. 9 variance views on a PDA

3.2. 구현 시스템의 테스팅

본 구현과정에서 JPEG Streamer의 기능, 안정성, 성능 등을 평가하기 위해 실제 PDA의 화면에서 영상의 지연 또는 끊김 현상이 없는지를 확인하였으며 네트워크 연결의 안정성을 위해서는 소켓을 지속적으로 감시하여 끊겨졌을 경우 일정간격으로 재접속을 시도하여 영상감시를 수행할 수 있도록 구현하여 안정성을 확보하였다.

또한, JPEG Streamer 컴포넌트를 탑재한 서버의 안정성은 서버 시스템의 CPU와 메모리 사용량을 주기적으로 체크하여 확인하였다.

Ⅳ. 결 론

실시간 무선 영상감시 시스템은 사용자 중심의 실용적인 애플리케이션으로 최근 무선 인터넷의 보급으로 보다 편리하고 간편한 영상감시 시스템이 증대될 것이다.

이러한 요구에 발 빠르게 대처하기 위해 기존 애플리케이션의 개발에 적용한 컴포넌트 기반의 개발방법을 실시간 영상감시 시스템에 적용함으로써 보다 효과적인 개발이 가능하다. 개발자는 제시된 JPEG Streamer를 사용함으로서 사용자 인터페이스 개발에 집중함으로써 전체 시스템 개발을 단순화하고, 생산성을 향상시키며 실시간 영상감시 시스템의 신뢰성을 높이고 기존의 NCS가 갖는 몇몇 제약사항들을 극복할 수 있다[3].

참고문헌

[1] 이진영, 윤영민, "무선망을 이용한 가정 및 사무실 보안 시스템에 관한 연구", 강남대학교 산업기술 연구소 논문집, pp.117-134, Vol.17, 2004.

[2] 한국인터넷정보센터,"2004년 4월 기준 인터넷통계 월보", 한국인터넷정보센터(KRNIC) 인터넷통계월보, 2004.

[3] 정기원, 최윤석, 김영희, "컴포넌트 기반 개발을 위한 통합CASE 프레임워크", 숭실대학교 대학원 논문집 제 20권, pp.329-343, 2002.

[4] 김세화, 박정근, 홍성수, "실시간 응용 프로그래밍을 위한 Pthread API 확장",한국정보과학회 컴퓨터 시스템 연구회 추계 학술발표회 논문집, pp. 105-112, 1999.

[5] 김화종, "컴퓨터 네트워크 프로그래밍", 홍릉과학출판사, pp. 103-144, 2002.

[6] Manfred Hauswirth, Mehdi Jazayeri, "A Component and Communication Model for Push Systems1", 7th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE-7), 1999.

[7] Harvey M. Deitel, Paul J. Deitel, David R. Choffnes, "Operating Systems, 3rd Edition", Prentice-Hall, 2004.

[9] 김성훈 역,"성공과 실패를 결정하는 1%의 네트워크 원리", 성안당, pp. 82-127, 2004.

[10] James H. Anderson, Yong-Jik Kim, "Shared-memory Mutual Exclusion: Major Research Trends Since 1986", 2001.

[12] Kale, T. Socolofsky, "A TCP/IP Tutorial", Network Working Group Request for Comments: 1180, RFC-1180, 1991.

저자소개

이 진 영(JinYoung Lee)

1995년 단국대학교 대학원 전산통계학과 졸업(석사)
1998년 단국대학교 대학원 전산통계학과 수료(박사)
1999년~현재 강남대학교 교양학부 조교수
※관심분야 : 이미지프로세스, 생체인식, 침입탐지 시스템

김 흥 준(HeungJun Kim)

1989년 단국대학교 전자계산학과 졸업(학사)
1993년 단국대학교 대학원 전산통계학과 (석사)
1999년 단국대학교 대학원 전산통계학과 (박사)
1999년~현재 진주산업대학교 컴퓨터공학부 부교수
※관심분야 : 컴퓨터구성, 모바일 네트워킹, etc.

이 광 석(Kwang-seok Lee)

1983년 2월 동아대학교 전자공학과 (공학사)
1985년 2월 동아대학교 전자공학과 (공학석사)
1992년 2월 동아대학교 전자공학과(공학박사)
2004년2월~2005년1월 Arizona State Univ. Fellowship
1995년~현재 진주산업대학교 전자공학과 교수
※관심분야 : 음성신호처리 및 인식, 생체 신호처리, 신경회로망, 지능화 기술