자전거 시뮬레이터에서 블록 시간을 최소화하기 위한 통신 프로토콜의 설계

Design of Communication Protocols with Minimum Blocked Time for an Interactive Bicycle Simulator

이 경 노, 이 두 용
(Kyungno Lee and Doo Yong Lee)

Abstract : The interactive bicycle simulator presented in this paper consists of a Stewart platform manipulator, magneto-rheological steering and braking devices, and a visual simulator. To provide a rider with reality, these devices should be controlled in real-time and motions of the devices and the visual should be also synchronized. If any of the devices and the visual gets unsynchronized due to significant blocking of control signals, the reality of the simulator is no longer secured. This paper presents communication protocols that minimize the blocked time of the control processes to guarantee the synchronization. The protocols are designed based on IPC (InterProcess Communications) of QNX, TCP/IP, and serial communication. The performance of the designed communication protocols is evaluated with the implemented bicycle simulator, and found satisfactory.

Keywords : communication protocol, interactive bicycle simulator, blocked time, synchronization

I. 서론

컴퓨터 그래픽으로 재생되는 가상 현실 환경과 운동감을 재현하여 주는 장치를 결합하여 현실감을 획기적으로 향상시킨 시뮬레이터의 성공 여부는 그 모델이 얼마나 정확하게 실제 세계를 표현하느냐에 의해서 결정된다. 시뮬레이터는 사용자에게 실제 세계에서의 조종환경과 동일한 환경을 제공해주어야 하며, 가상 현실 환경 속의 영상과 사물의 움직임, 그리고 사용자가 느끼는 상황의 변화가 실제 세계에서의 느낌과 유사해야 한다[1]. 특히 사용자의 작용에 대한 시뮬레이터의 반응이 제한된 시간이내에 처리되어야 사용자는 현실감을 느낄 수 있다. 일반적으로 시뮬레이터는 이러한 시간적 제약 조건을 만족시켜야 하는 실시간 시스템이며, 디스플레이 장치와 여러 개의 센서 및 엑츄에이터 등의 병행 활동을 위해 분산 및 멀티프로세서로 구성되어야 하는 시스템이다. 이 경우, 시뮬레이터의 상태 정보 공유 및 제어 프로세스들간의 동기화를 위해서 상호 통신을 수행하게 되는 데, 다른 프로세스와의 동기화과정에서 프로세스의 블록현상 (block)이 발생하게 되고 이런 블록된 시간이 길어지면 프로세스의 시간 제약 조건을 어기게 되어 시뮬레이터의 현실감을 떨어뜨리게 한다. 따라서 이러한 문제점을 해결하고자, 제어 프로세스의 블록된 시간을 최소화하고 분산된 제어 프로세스들 사이의 정보 공유 및 효과적인 동기화를 위한 프로세스들 사이의 통신 방법 결정과 통신 프로토콜의 설계가 필요하다.

본 논문에서는 그림 1에서 보는 바와 같이 자체 개발한 양방향 자전거 시뮬레이터를 대상으로 제어 프로세스 사이의 동기화 및 정보 교환 방법을 제안하고, 프로세스의 블록 시간을 최소화하기 위해 설계된 통신 프로토콜을 논하고자 한다.

양방향 자전거 시뮬레이터는 사용자가 스튜워트 플랫폼 매니퓰레이터 (SPM) 위에 설치된 자전거를 타고 전방에 투사되는 그래픽 환경 내에서 실제 자전거를 주행하듯 페달과 조향 장치를 조작하면서 현실적 운동감과 시각 효과의 재현을 경험할 수 있게 해준다. 자전거 시뮬레이터는 자전거를 포함하여, 운동감을 재현하는 스튜워트 플랫폼 매니퓰레이터 (SPM), 반력을 구현하는 조향 장치와 바퀴 페달 장치, 시각 시뮬레이터와 투사장치, 그리고 각 세부 시스템의 제어 모듈과 그 연결 네트워크으로 구성되어 있다[2]. 그리고 제어 시스템은 3 대의 제어 컴퓨터로 구성되어 있다. 제어 컴퓨터 I 은 DOS환경에서 SPM의 운동을 제어한다. 스튜워트 플랫폼 매니퓰레이터 (SPM) 및 제어 시스템은 90년대 초반에 진동 제어 실험을 목적으로 개발을 시작하였다. 제어 시스템 개발 당시에는 DOS환경이 가장 보편적이고 하드웨어 제어가 쉬운 운영체제이었기 때문에 DOS환경을 바탕으로 한 제어 시스템

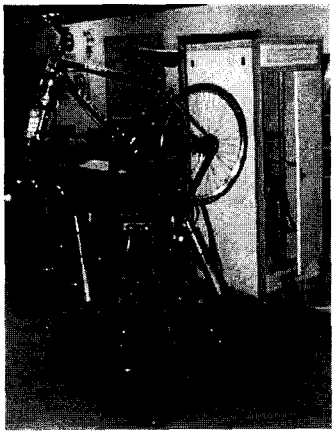


그림 1. 한국과학기술원의 양방향 자전거 시뮬레이터 [2].

Fig. 1. KAIST interactive bicycle simulator[2].

접수일자 : 2000. 2. 12., 수정완료 : 2000. 7. 20.
이경노, 이두용 : 한국과학기술원 기계공학과

이 개발되었다. 그 후 개발된 시스템을 수정, 보완하여 자전거 시뮬레이터용으로 변환한 것이다. 제어 컴퓨터 Ⅲ은 윈도우 NT환경에서 시각 시뮬레이터를 제어한다. 3D 모델링 및 시각 시뮬레이터는 Win 9x/Win NT 운영체제에 맞게 설계된 가상 현실감 응용 시스템 개발 표준 API (Application Programming Interface)인 Kitten을 이용하는 방법과 상용 도구인 MultiGen Creator™로 지형, 지물을 모델링한 후 상용 실시간 렌더링 툴인 Vega™을 이용하는 방법을 이용하여 구현하였다. Kitten, MultiGen Creator™, 그리고 Vega™의 운영 환경이 윈도우즈인 관계로 제어 컴퓨터 Ⅲ는 윈도우 NT환경에서 구현되었다. 제어 컴퓨터 Ⅱ는 자전거 동역학을 이용하여 SPM, 조향 반력 장치, 페달 반력 장치, 시각 시뮬레이터의 제어관련 기준 값을 계산하고, 자전거 조향 장치와 바퀴, 페달 반력 장치의 제어를 담당하며, 세부 시스템들을 통합, 관리하여 동기화시켜주는 역할을 한다. 이러한 역할들을 실시간으로 동시에 수행하기 위해서 선택될 운영체제는 병렬 프로세싱 기능을 제공하고, 각 프로세스의 우선 순위 변경, 작업 시간의 설정 및 관리기능과 정확하고 빠른 프로세스들 사이의 메시지 교환 방법을 제공해 주어야 한다. QNX는 요구되는 기능들을 모두 제공해줄 뿐만 아니라, 실시간 운영체제로서 자전거 시뮬레이터의 실시간 제어에 적합하여 제어 컴퓨터 Ⅱ의 운영 체제로 결정되었다.

그리고 제어 컴퓨터 Ⅰ과 Ⅱ는 DOS환경을 사용하였기 때문에 네트웍 대신 보편적이고 구현이 쉬운 시리얼 통신으로, 제어 컴퓨터 Ⅱ와 Ⅲ은 윈도우 NT 환경을 사용하였기 때문에 TCP/IP 통신으로 제어관련 입출력 정보를 송수신하도록 구성되어 있다. 이처럼 분리되어 있는 제어 시스템들은 각 시스템의 특성에 따른 시간제약조건을 엄수하고, 사용자의 입력 정보를 실시간으로 공유하고 동기화하여 동작해야 한다. 그리고 이러한 동기화 과정에서 프로세스의 블록현상이 발생한다. 권이완[3]은 실시간 분산 처리 시스템에서 CPU의 클럭을 동기화시켜서 프로세스들의 동기화를 구현하였으나, 이 방법은 앞서 설명한 바와 같이 자전거 시뮬레이터의 제어 컴퓨터의 운영 시스템이 UNIX 계열이 아닌 관계로 그 방법의 적용이 어렵다. 이기웅[4]은 실시간 시스템에서 동기화를 위한 방법으로 세마포를 사용하는 경우, 고순위 프로세스가 언제나 저순위 프로세스가 사용하던 자원을 선점하도록 하는 방법을 제시하였으나, 이는 저순위 프로세스가 고순위 프로세스에 의해 블록되는 시간은 고려하지 않았다. 따라서 사용자의 입력 정보를 공유하여 프로세스들이 연동되도록 하는 한편, 그로 인해 발생되는 프로세스의 블록현상이 최소화되도록 시스템 특성에 맞는 프로세스간 동기화 및 정보 공유 방법, 통신 프로토콜의 설계가 필요하다. 이러한 통신프로토콜의 필요성은 차량 시뮬레이터를 개발한 기존 연구에서도 지적된 바가 있다[5][6]. 차량 시뮬레이터를 개발하면서 독립된 시스템들을 네트웍 상에서 병렬적으로 연결한 연구에서는 독립된 시스템이 네트웍에 연결될 때마다 네트웍 상에서 처리해야 할 입출력 데이터가 증가하므로 시스템이 일정 수준이상으로 확장되면 네트워크의 지연이 초래될 것이라 하였다[5]. 따라서 이를 보상하기 위해서 네트웍을 지속적으로 모니터링하여 시스템의 통합적 관리 및 입출력 데이터를 동기화 시켜줄 수 있는 관리제어 (Supervisor) 컴퓨터의 추가가 필요할 것이라 하였다. 그리고 차량 시뮬레이터에서 운전자에게 현실감을 부여하기 위해서는 트랜스포트 지연 (Transport Delay)을 최소화하는 시스템의 설계가 필요하다고 하였다[6]. 그러나, 차량 시뮬레이터[5][6], 건설 차량 시뮬레이터[7], 그리고 자동차 운전 연습기[8] 등과 관련한 기존의 연구에서는 네트웍 상의 시간 지연, 트랜스포트 지연, 그리고 동기화과정에서 발생하는 프로세스의 블록 현상을 정량적으로 분석하고 이를 최소화하기 위한 방법들을 제시하지 않았다.

본 논문은 자전거 시뮬레이터의 시스템 통합을 위해 구현된 통신 프로토콜과 프로세스들을 설명하고 그에 따른 성능 평가 결과를 논한다. 통신 프로토콜은 실시간 운영 시스템 (RTOS)인 QNX에서 지원해주는 기본적인 메시지 전달 방법을 기반으로 응용, 확장되었으며, 제어 프로세스들이 시스템 정보를 교환할 때 블록되는 시간이 최소화되도록 설계되었다. 그리고 이는 실제 자전거 시뮬레이터를 동작시키면서 얻은 정보를 토대로 각 세부시스템의 송수신 흐름, 블록킹 현상의 유무, 제어 주기 등의 성능 지수를 통해 평가되었다.

Ⅱ. 구현된 제어 시스템과 송수신 정보

1. 제어 시스템의 특징 및 시스템 요구 조건

자전거 시뮬레이터의 제어 시스템은 3 대의 컴퓨터로 분리되어 있으며, 제어 컴퓨터 Ⅰ은 펜티엄 PC로서 운영 시스템이 DOS이며, 부드러운 동작을 위해서 SPM의 운동을 1 ㎳ 주기로 제어한다. 제어 컴퓨터 Ⅱ는 펜티엄 Ⅱ PC로서 운영 시스템이 실시간 운영 시스템 (RTOS)인 QNX이며, 자전거 동역학을 이용하여, SPM, 조향 및 페달 반력 장치, 그리고 시각 시뮬레이터의 제어관련 기준 값을 계산하고, 자전거 조향 장치와 바퀴, 페달 반력 장치의 제어를 담당한다. 자전거 시뮬레이터에서 사용자에 의한 최대 입력 주파수는 실제 측정 결과 10 ㎐ 미만으로 판단되기 때문에 핸들과 페달 반력 장치가 사용자의 입력 값을 빠르게 추종하도록 제어 주기를 100 ㎐ 이상의 주파수로, 즉 10 ㎳이하의 주기로 제어되어야 한다. 그리고 자전거 동역학 계산은 적응형 간격제어 5 차 Runge-Kutta 알고리즘 (5th order Runge-Kutta algorithm with stepsize control)으로 계산하는 데, 자전거 동역학 방정식의 해가 언제나 수렴하려면 초당 500 번이상의 반복 연산이 필요하다[9][10]. 그러므로 자전거 동역학 계산은 핸들과 페달 반력 장치의 제어 시스템처럼 제어 주기를 100 ㎐ 고정하면 자전거 동역학 방정식의 해가 발산하는 문제가 발생하게 된다. 제어 컴퓨터 Ⅲ은 펜티엄 Ⅱ PC로서 윈도우 NT 환경에서 시각 시뮬레이터를 제어하는 데, 시각 시뮬레이터의 렌더링 (rendering) 속도가 대략 60 ㎳로 느린 편이다. 만약 제어 컴퓨터 Ⅲ에서 화면에 1회 디스플레이하는 동안 제어 컴퓨터 Ⅱ와 1 회 송수신 통신을 수행한다고 가정하자. 제어 컴퓨터 Ⅲ의 3D 모델는 초당 15~20 프레임으로 디스플레이되므로, 제어 컴퓨터 Ⅲ이 자전거 동역학 서버에 노면 데이터를 넘겨주는 속도는 대략 1 초에 20 회 정도이다. 자전거 동역학이 초당 1000 회 계산되고 있다고 가정하면 그래픽 컴퓨터에서 전달되어온 노면 데이터 1 개로 50 회의 동역학 계산을 해야한다. 만약 이 50 회 모두 같은 노면 데이터, 즉 같은 평면 방정

식으로 계산하게되면 노면이 급격히 변하는 경우 자전거 동역학 계산의 해가 발산할 수 있다. 그리고 제어 컴퓨터 III에서 화면에 1 회 디스플레이하는 동안 제어 컴퓨터 II와 지나치게 많은 송수신을 반복하면 디스플레이되는 시간 간격이 커져서 화면이 부자연스럽게 움직일 수 있다. 따라서 자전거 동역학 계산 해의 발산을 막고, 화면의 부드러운 움직임을 위해서 시행착오적으로 그 회수를 조정해본 결과 제어 컴퓨터 III에서 화면에 1 회 디스플레이하는 동안 제어 컴퓨터 II와 10 회의 송수신 통신을 수행하는 것이 가장 적합하였다. 따라서 화면 부드러운 변화와 자전거 동역학 방정식의 해가 언제나 수렴하기 위해서 화면에 1 번 디스플레이하는 동안 제어 컴퓨터 II와 10 번의 송수신 통신을 수행한다. 그리고 자전거 시뮬레이터의 제어 컴퓨터간의 통신 방법은 그림 2에서 보는 바와 같이 제어 컴퓨터 I 과 II는 RS232 포트를 이용한 시리얼통신으로, 제어 컴퓨터 II와 III사이에서는 TCP/IP 네트웍으로 구성되어 있다.

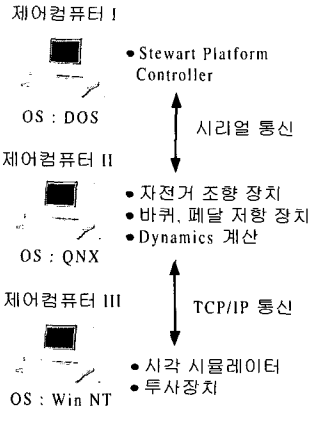


그림 2. 제어 컴퓨터 구성.

Fig. 2. Configuration of the control system.

2. 송수신 정보

자전거 시뮬레이터에서 제어 컴퓨터의 제어관련 입출력 정보의 내용은 그림 3에 나타내었다. 제어 컴퓨터 I 이 제어 컴퓨터 II로 송신하는 정보의 크기는 8 byte, 제어 컴퓨터 II로부터 수신하는 정보의 크기는 24 byte이며, 제어 컴퓨터 II가 제어 컴퓨터 III으로 송신하는 정보의 크기는 76 byte, 제어 컴퓨터 III으로부터 수신하는 정보의 크기는 52 byte이다.

III. 통신 프로토콜의 설계

II장에서 설명한 시스템의 요구조건을 토대로 자전거 시뮬레이터의 입출력 제어 정보의 흐름을 살펴보면 다음과 같다. 제어 컴퓨터 II의 자전거 동역학 모듈은 핸들, 페달의 반력 장치로부터 직접 사용자의 입력 정보를 취득하는 한편, 제어 컴퓨터 I 과 III으로부터 사용자의 입력정보를 전달받아 동역학을 계산하고 각 제어기들의 기준 값 (reference input)들을 반환한다. 제어 컴퓨터 I 은 반환된 기준 값을 이용하여 SPM을 제어하고 제어 컴퓨터 III은 3D 모델을 갱신한다.

본 장에서는 이러한 일련의 정보 교환 과정에서 발생하게

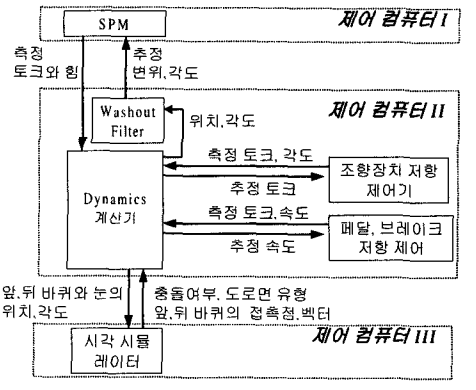


그림 3. 제어컴퓨터 및 프로세스들간의 입출력 정보.

Fig. 3. The input and output signals among the control computers and processes.

되는 블록 현상의 요인을 분석하고 이를 최소화하기 위해 설계된 통신 프로토콜을 설명한다.

1. 제어 컴퓨터 II내의 프로세스간 통신

제어 컴퓨터 I 은 SPM 제어와 시리얼 통신, 제어 컴퓨터 III은 3차원 모델의 화면 재생과 TCP/IP 통신으로, 즉 제어 대상 시스템 하나와 통신 모듈 하나로 구성되어 멀티 프로세싱이 필요 없으나, 제어 컴퓨터 II는 자전거 동역학 계산, 조향 장치 제어, 페달 반력 장치 제어, 시리얼 통신 및 TCP/IP 통신 등의 여러 가지 역할을 동시에 수행해야 하므로 멀티 프로세싱이 필요하다. 그리고 각 장치들을 독립적으로 제어 및 관리하고자, 제어 컴퓨터 II는 6 개의 프로세스로 구성되었다. 자전거 동역학을 계산하는 동역학 프로세스, 조향 장치의 반력을 제어하는 핸들 프로세스, 페달, 브레이크의 반력을 제어하는 페달 프로세스, 제어 컴퓨터 I 과 시리얼 통신을 수행하는 시리얼 프로세스, 제어 컴퓨터 III과 TCP/IP통신을 수행하는 tcpip 프로세스, 그리고 이들 프로세스를 생성, 종료시키거나 프로세스들 사이의 통신을 위한 서버역할을 수행하며, 입출력 정보를 갱신하는 역할을 담당하는 메인 프로세스로 구성되었다.

프로세스간의 기본적인 통신 프로토콜은 그림 4에서 보는

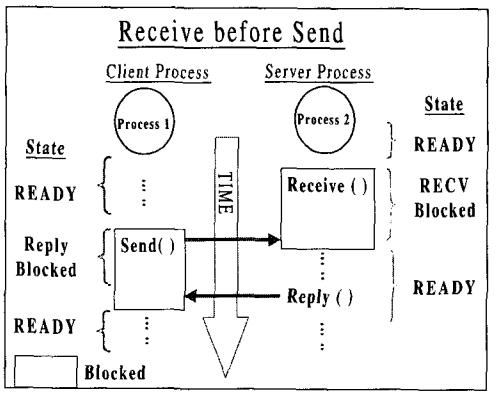


그림 4. 프로세스간의 통신[11].

Fig. 4. InterProcess communications[11].

바와 같다[11]. 여기서 메인 프로세스는 Server_Process에 해당하며, 나머지 5 개의 프로세스는 Client_Process에 해당한다. QNX에서는 프로세스들간의 통신을 IPC (InterProcess Communicaition)이라고 칭한다. 여러 개의 프로세스가 동시에 메인 프로세스에 접속하게 되는 경우, 우선 순위가 가장 높은 프로세스에 대한 응답을 먼저하며, 우선 순위가 같을 경우에는 FIFO 원칙에 따라 응답을 한다.

프로세스간 통신 중에서는 그림 4에서 보는 바와 같이 Server_Process와 Client_Process가 블록 (Block)되는 현상이 발생한다. 이 현상은 QNX에서의 메시지 전달 방법이 동기화를 기본으로 하고 있기 때문이다. 여기서 Server_Process, 즉 메인 프로세스의 블록 현상은 메인 프로세스가 외부 하드웨어를 제어하지 않고, 메인 프로세스의 블록이 다른 프로세스의 작업 수행에 영향을 주지 않기 때문에 중요하지 않다. 메인 프로세스는 다른 프로세스가 응답을 필요로 할 때 곧바로 반응해야 하므로 항상 대기상태로 유지되어야 한다. 그러나, Client_Process가 블록되는 시간은 그 시간동안 외부 하드웨어의 제어가 불가능하므로, 최대한 짧게 해야한다. 이런 블록 시간을 최소화하는 방법은 Server_Process가 Client_Process에 최대한 빨리 응답을 해주는 것이다. 따라서, 본 시스템에서는 메인 프로세스가 Client_Process로부터 새로운 정보를 받으면 그 정보만 갱신하고, 곧바로 응답을 해주도록 설계되었다.

2. 제어 컴퓨터 Ⅰ과 Ⅱ사이의 통신 프로토콜

그림 5. 시리얼 통신의 송신 정보 패킷.

Fig. 5. The signal packet form of the serial communication.

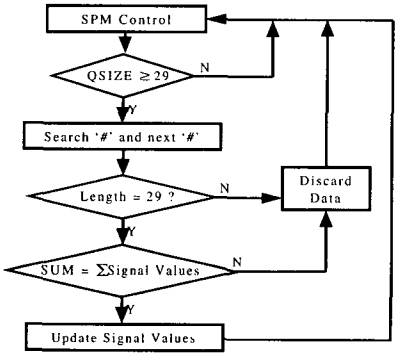


그림 6. 시리얼 통신을 통해 수신된 정보의 타당성 검증 절차.

Fig. 6. The validation procedure for the received signal values.

제어 컴퓨터 Ⅰ과 Ⅱ는 RS232 포트를 이용한 시리얼통신으로 정보를 송수신한다. 일반적으로 시리얼 통신만으로는 안정적인 데이터 송수신을 보장할 수 없다. 잘못된 정보의 수신으로 인해 사용자가 타고 있는 SPM이 오작동 하는 것을 방지하기 위해서 그림 5에서 보는 바와 같이 '#' 기호와 'SUM'이라는 전송되는 신호 값들의 합을 함께 전송한다. 제어 컴퓨터 Ⅰ은 그림 6과 같은 절차를 거쳐서 시리얼 통신을 통해서 수신된 정보의 타당성을 검증한다. 그림 6에서 QSIZE는 수신 버퍼에 쌓여있는 신호들의 바이트 수를 의미하며, Length는 '#'에서 그 다음 '#'까지의 바이트 수를 의미한다. 수신된 바이트의 수가 전송된 바이트 수와 일치하지 않거나 수신된 신호들의 합과 SUM의 값이 일치하지 않으면 수신된 신호들을 무시하고, 과거의 신호 값을 이용하여 SPM을 제어하게 된다. 시리얼 통신을 통해서 송수신되는 정보의 크기는 제어 관련 정보 32 byte와 이 정보의 타당성 검증을 위해서 추가된 기호 '#'이 1 byte, 전송되는 신호 값들의 합인 SUM이 4 byte로서 32 byte + 2*5 byte = 42 byte이다.

시리얼 통신은 일반적으로 인터럽트 (interrupt)를 이용하여 데이터를 송수신한다. 이 방법은 비록 블록되는 시간이 없지만, 데이터 저장을 위한 메모리 용량이 커야 하고, 본 시스템의 경우는 가장 최근에 송수신된 정보만이 필요한 데, 불필요하게 과거의 정보를 저장하는 등 비효율적이다. 그리고 무엇보다도 메인 프로세스와 동기화가 불가능하다. 따라서, 제어 컴퓨터 Ⅰ은 SPM을 제어하기 때문에 블록되는 시간이 없어야 하므로 수신은 인터럽트에 의한 방법을 사용하고, 제어 컴퓨터 Ⅱ는 동기화 (synchronization)의 방법으로 정보를 수신한다. 이러한 동기화 방법을 사용함으로써 시리얼 통신을 위한 버퍼의 크기를 줄일 수 있고, 자전거 동역학에 의해서 계산된 정보를 실시간으로 동기화시켜 송수신할 수 있다. 그림 7은 이러한 관점에서 설계된 통신 프로토콜의 흐름을 보여준다.

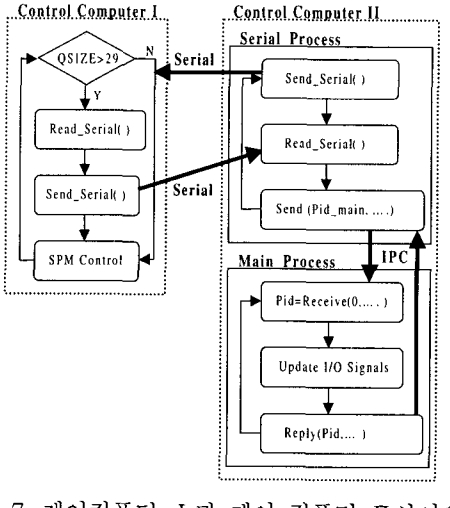


그림 7. 제어컴퓨터 Ⅰ과 제어 컴퓨터 Ⅱ사이의 통신 프로토콜.

Fig. 7. Communication protocol between control computers Ⅰ and Ⅱ.

시리얼 프로세스는 동역학 프로세스로부터 계산된 SPM 제어 정보를 인터럽트 방식에 의해서 제어 컴퓨터 Ⅰ로 송신한다. 그리고, 시리얼 프로세스는 Read_Serial()상태에서 블록되어 제어 컴퓨터 Ⅰ로부터의 수신을 기다린다. 제어 컴퓨터 Ⅰ은 SPM을 제어하다가 시리얼 버퍼에 29 byte이상의 정

보가 쌓이면 수신된 제어 정보를 읽어들이고, 그림 7에서 보는 바와 같이 Send_Serial()을 통해 측정된 토크와 힘을 송신한 후, 계속적으로 SPM을 제어한다. 따라서 제어 컴퓨터 I에서 블록되는 시간은 없다. 시리얼 프로세스는 제어 컴퓨터 I로부터 정보를 수신하면, Send(Pid_main, ...)을 통해 메인 프로세스에 새로 수신된 토크와 힘 정보를 송신하고, 자전거 동역학에 의해서 계산된 정보들을 수신한다.

3. 제어 컴퓨터 II와 III사이의 통신 프로토콜

제어 컴퓨터 II와 제어 컴퓨터 III사이에서는 TCP/IP 네트워을 이용하여 통신한다. 일반적으로 TCP/IP 통신은 정보를 송수신하기 위해서 서버에 접속을 할 때마다 소켓을 생성하고, 접속하고, 소켓을 해지하는 등 정보 교환이외의 소요시간이 길기 때문에 실시간으로 정보를 송수신할 때는 적합하지 않다[12][13]. 그러나, 본 자전거 시뮬레이터에서는 제어 컴퓨터 두 대간의 통신이기 때문에 일단 제어 컴퓨터 II와 III사이에 통신 소켓이 형성되면 이를 계속 유지하기 때문에 실시간으로 정보 교환하는 데에도 무리가 없다. 그리고 TCP/IP 통신 방법은 정보를 안정적으로 송수신하기 때문에 시리얼 통신처럼 수신된 정보를 검증하기 위한 특별한 절차나 패킷이 필요없다. 그림 8은 제어 컴퓨터 II와 III이 128 byte 크기의 정보를 TCP/IP 통신을 이용하여 송수신할 때 소요되는 시간이다.

그림 8에서 가로 축은 서버와 소켓 연결이 된 후의 송수신 반복회수를 의미하고, 세로 축은 1 번 송수신했을 때 소요되는 시간이다. 즉 두 번째 막대에서 0.797의 의미는 100 번 송수신을 반복했을때 걸린 시간을 100으로 나눈 값이다. 송수신 회수를 n, 제어 컴퓨터 II와 III사이에 통신 소켓이 형성되는데 걸리는 시간을 t_0라고 하면, 그림 8에서 1 회 송수신 소요시간 t는 $\frac{t_0 + n*t_1}{n}$ 즉, t= $\frac{t_0}{n}$ +t_1의 식으로 계산된다. 따라서, 그림 8에서 보는 바와 같이 송수신 반복 회수 n이 커지면서 t_0의 영향은 없어지게 되고, 1 회 송수신에 평균 0.76 ㎳의 시간이 걸린다.

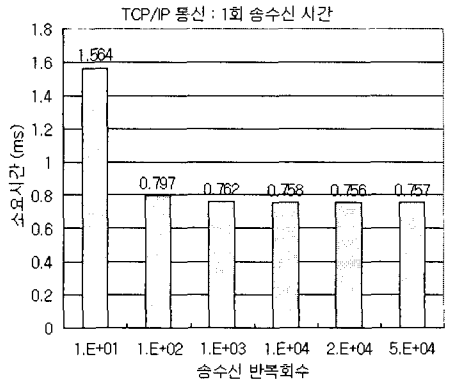


그림 8. TCP/IP 통신의 성능평가.

Fig. 8. Performance of TCP/IP communication.

TCP/IP 통신의 경우는 시리얼 통신과는 달리, 0.76 ㎳로 송수신 시간이 매우 짧고, 렌더링하는 시간이 60 ㎳로 비교적 길다. 핸들과 페달 프로세스는 10 ㎳, 동역학 프로세스는 1 ㎳ 미만의 주기로 정보가 갱신되기 때문에 이 3D 모델의 렌더링 시간 60 ㎳동안 핸들과 페달 프로세스는 6 번의 정보 갱신을, 동역학 프로세스는 60 번의 정보 갱신을 하게 된다. 따라서 제어 컴퓨터 III의 시각 시뮬레이터는 tcpip 프로세스가 메인 프로세스와 정보 교환을 행한 이후부터 제어 컴퓨터 III과 정보 교환을 하게 될 때까지 걸린 시간만큼의 과거 동역학 계산 값들과 동기화 된다. 그러므로 사용자의 최근 입력 정보, 그리고 이를 이용하여 계산된 자전거 동역학의 계산 값들과 시각 시뮬레이터의 동기화를 위해서 그림 9와 같이 메인 프로세스와 동기화되도록 프로토콜을 설계하였다.

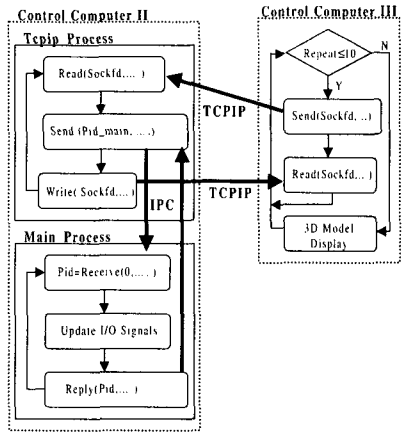


그림 9. 제어 컴퓨터 II와 III 사이의 통신 프로토콜.

Fig. 9. Communication protocol between control computers II and III.

메인 프로세스와의 동기화 과정에서 제어 컴퓨터 III의 read(sockfd,...)에서 블록현상이 발생하는 데, 이 블록 시간을 최소화하기 위해서 tcpip 프로세스와 메인 프로세스간의 통신 시간이 짧아야 한다.

IV. 통신 프로토콜의 성능 평가

자전거 시뮬레이터의 세부 제어 시스템 및 하드웨어 장치와 이들 간의 통신 모듈을 완성하고, 사용자가 실제로 탑승하여 시뮬레이터를 동작시켜 본 결과를 분석하였다.

그림 10은 제어 컴퓨터 II내의 메인 프로세스와 다른 5 개

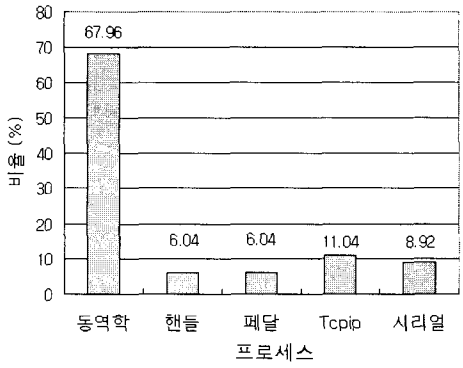


그림 10. 메인 프로세스와 다른 5 개 프로세스들과의 송수신 비율.

Fig. 10. Distribution of the communication loads.

프로세스들간의 총 통신회수에 대한 각 프로세스 별 메인 프로세스와의 통신회수의 비율이다.

동역학 프로세스는 67.96 %의 통신회수 비율을 나타내고, 이 때 자전거 동역학을 초당 1120 번 계산한다. 핸들 프로세스와 페달 프로세스는 QNX에서 지원하는 timer를 사용해서 100 Hz로 고정된 상태이다. 따라서 핸들 및 페달 프로세스가 100 Hz로 고정되어 있다는 사실과 그림 10에서의 프로세스별 비율을 종합해 보면, 핸들 프로세스와 페달 프로세스의 송수신 비율 값이 6.04 이고 tcpip 프로세스의 송수신 비율 값이 11.04 이므로 100 Hz : x = 6.04 : 11.04 라는 비례식이 성립한다. 이를 계산하면, TCP/IP 통신은 대략 182 Hz로 송수신됨을 알 수 있다. 마찬가지 방법으로 시리얼 통신은 147 Hz로 송수신됨을 알 수 있다. 특히 TCP/IP 통신 속도는 그림 8에서 제시된 TCP/IP 통신 속도에 비하면 매우 느린 편인데, 그 이유는 3D 모델을 렌더링할 때 시간이 많이 소요되기 때문이다.

그림 11은 메인 프로세스의 총 통신 회수 증가에 따른 다른 5 개 프로세스들의 통신 회수의 비율 변화를 나타낸다. 여기서 특이한 점은 시뮬레이터의 동작 초기에는 동역학 프로세스의 비율이 낮았다가 통신 회수가 증가함에 따라서 높아지다가 결국 68 %정도에서 수렴한다는 것이다. 이 현상의 원인은 시뮬레이터 동작 초기에 동역학 프로세스, 핸들 프로세스, 페달 프로세스, 시리얼 프로세스, 그리고 tcpip 프로세스가 동시에 메인 프로세스에게로 사용자의 입력 정보를 송신하는 과정에서 Send 블록이 발생하였기 때문이다. 초반에 동시에 집중되었던 Client 프로세스들의 접속은 시간이 흐르면서, 핸들 프로세스와 페달 프로세스는 100 Hz로, 시리얼 프로세스는 시리얼 통신 시간에 의해서 147 Hz로, tcpip 프로세스 역시 tcpip 통신 시간과 렌더링 시간에 의해서 182 Hz로 수렴하게 되어 규칙적인 접속을 하게 되고, 따라서 cpu자원이 남을때마다 동작하는 동역학 프로세스의 통신회수는 자연 증가하게 된다. 이런 이유로 그림 11과 같은 그래프가 나타나게 된다. 그림 11에서 메인 프로세스와의 통신회수가 500 번 이상이 되면 모든 프로세스들이 블록이 없이 정상 동작함을 알 수 있는데 이 때까지 소요되는 실제 시간은 평균 0.35 초이고, 이 시간동안 각 프로세스 별 블록시간은 수 ms로서 시뮬레이터의 동작에 큰 영향을 주지 않는다.

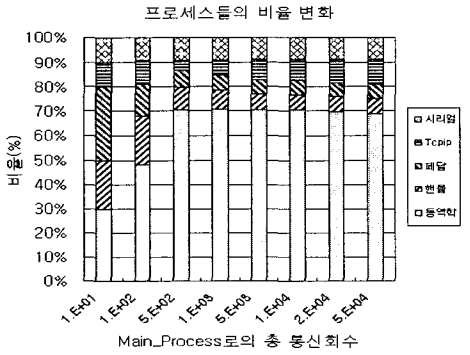


그림 11. 시간 흐름에 따른 메인 프로세스와 다른 5 개 프로세스와의 송수신 비율 변화도.

Fig. 11. Distribution change of the communication loads.

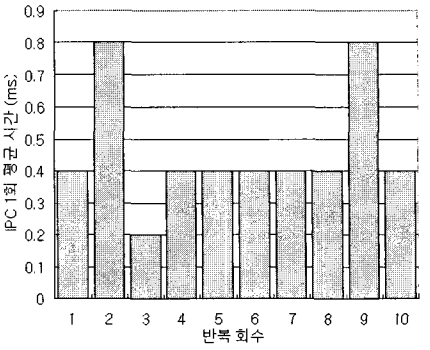


그림 12. 메인 프로세스와 tcpip 프로세스간의 통신 성능 평가.

Fig. 12. Performance of IPC between main process and tcpip process.

그림 12는 tcpip 프로세스와 메인 프로세스 사이의 1 회 송수신 시간을 나타낸 것이다.

그림 12는 송수신 회수가 50 회 반복될 때마다 평균을 계산하여 1 회 송수신할 때 소요되는 시간을 계산한 것으로 10 회 반복하였으며 10 회 평균 송수신 시간은 0.42 ms이다. 그러므로 Ⅲ장에서 설명한 바와 같이 제어 컴퓨터 Ⅲ의 블록 시간은 TCP/IP 통신에 소요되는 평균 시간 0.76 ms와 프로세스 사이의 통신에 소요되는 평균 시간 0.42 ms를 합쳐 1.18 ms이다. 최대 프로세스 간 통신 소요 시간 0.8 ms을 합쳐도 1.56 ms로 이는 사용자가 느끼지 못할 만큼 작은 값이다.

표 1. 프로세스의 우선 순위 변경에 따른 프로세스별 상대 비율, 동역학 계산 속도, IPC 소요 시간의 변화.

Table 1. The communication loads, computation cycle of the bicycle dynamics, IPC loads according to the changes of the priorities of processes.

		동역학	핸들	페달	tcpip	시리얼	초당 동역학 계산 회수	1회 IPC 평균 시간(ms)
표준 (모두 10ω)	상대비율 (%)	69.96	5.97	5.96	9.94	8.17	1180회	0.54
	×100 (㎐)	11.75	1.0	1.0	1.67	1.37		
tcpip, 시리얼, main만 12ω	상대비율 (%)	60.90	6.37	6.37	14.75	11.61	961회	0.065
	×100 (㎐)	9.56	1.0	1.0	2.32	1.82		
tcpip, main만 12ω	상대비율 (%)	64.08	6.01	6.01	15.93	7.98	1068회	tcpip : 0.03 시리얼 : 0.83
	×100 (㎐)	10.67	1.0	1.0	2.6	1.3		

다른 프로세스 사이의 송수신 시간도 편차가 있지만 그림 12에서 보는 바와 같이 평균 0.42 ms의 값을 갖는 데, 각 프로세스의 우선 순위 (priority)를 변경함으로써 송수신 시간을 줄일 수 있다. 표 1은 3 가지 스케쥴링 방법에 따른 자전거 동역학의 계산 속도와 동역학, 핸들, 페달, tcpip, 그리고 시리얼 프로세스의 상대 비율, 그리고 tcpip 프로세스, 시리얼 프로세

스와 메인 프로세스 사이의 1 회 송수신 평균 시간의 변화를 보여준다. 표 1에서 10o는 프로세스의 우선 순위가 10이고 스케쥴링 방식이 adaptive 방식임을 나타낸다. Adaptive 방식은 커널 (kernel)에서 정의된 하나의 time-slice동안 프로세스가 동작한 후, 자신의 우선 순위를 한 단계 낮추는 방식이다. 프로세스의 우선 순위는 그 값이 클수록 우선 순위가 높은 것이다.

시리얼, tcpip, 메인 프로세스의 우선 순위가 높아지면, 동역학, 핸들, 페달 프로세스에 의한 블록 현상이 없어지기 때문에 프로세스의 상대 비율이 높아지고, 프로세스 사이의 통신 시간이 줄어든다. 표 1에서 보면, tcpip 프로세스는 우선 순위가 10에서 12로 높아지면서 프로세스들 사이의 상대 비율이 9.94 %에서 14.75 %로, 프로세스의 처리 속도가 167 Hz에서 232 Hz로, 프로세스 사이의 1 회 송수신 시간이 0.54 ㎳에서 0.065 ㎳로 줄었다. 그러나, 동역학 계산 회수는 초당 1180 회에서 961 회로 줄었다. 자전거 동역학 계산 값의 안정적인 수렴을 위해서 최소 500 회 이상의 반복 계산이 필요하다는 조건을 만족시키므로 자전거 시뮬레이터 시스템에는 영향을 주지 않는다. 프로세스의 우선 순위를 변경하여 프로세스 사이의 송수신 시간을 줄임으로써 제어 컴퓨터 Ⅱ에서 프로세스 사이의 통신에 의한 블록 시간을 0.065 ㎳로, 또 제어 컴퓨터 Ⅲ에서의 블록 시간을 0.825 ㎳로 줄일 수 있었다. 이 방법은 통신 로드 (communication load)가 큰 프로세스가 있을 경우에 적용하면 효과적이다.

V. 결론

본 논문에서는 양방향 자전거 시뮬레이터의 동기화 및 블록 시간의 최소화를 위해서 설계된 통신 프로토콜을 논하였다. 제어 시스템이 3 대의 제어 컴퓨터로 분산되어 있고, 특히 제어 컴퓨터 Ⅱ에서는 실시간 운영 시스템인 QNX를 기반으로 6 개의 프로세스가 병렬적으로 실행되고 있는 환경 속에서 각 제어기가 취득한 사용자의 입력 정보를 공유하고, 프로세스들을 동기화시키는 한편, 프로세스들의 블록 시간을 최소화하는 통신 프로토콜을 설계하였다. 그 결과 제어 컴퓨터 Ⅰ과 Ⅱ를 연결하는 시리얼 통신에서는 제어 컴퓨터 Ⅰ이 블록되는 시간을 없앨 수 있었으며, 제어 컴퓨터 Ⅱ와 Ⅲ을 연결하는 TCP/IP 통신에는 제어 컴퓨터 Ⅲ이 블록되는 시간을 최대 0.825 ㎳로 줄일 수 있었다. 그리고 제어컴퓨터 Ⅱ내에서의 프로세스들간의 통신상에 발생하는 블록 시간은 그림 12에서 보는 바와 같이 평균 0.42 ㎳이며, 프로세스의 우선 순위를 변경함으로써 표 1에서 보는 바와 같이 0.065 ㎳로 줄일 수 있었다. 시뮬레이터의 동작 초기에는 Ⅳ장에서 설명한 바와 같이 블록현상이 발생하지만 이것은 0.35 초 미만의 시간동안 발생할 뿐이었다. 실제로 구현된 자전거 시뮬레이터는 사용자의 입력에 대해서 화면, SPM, 페달 반력, 그리고 핸들 반력이 실시간으로 연동되어 사용자에게 현실감을 재현하기에 충분했다.

참고문헌

[1] 박찬모외 3인, RTS 기술개발, 제1차년도 연차보고서, 1995.

[2] 이두용외 16인, "한국과학기술원 자전거 주행 시뮬레이터", 대한기계학회 동역학 및 제어 부문-생산 및 설계부문 공동학술대회, pp. 130-137, 1999.

[3] 권이완, 실시간 분산처리 시스템의 시간동기화, 석사학위논문, 한국과학기술원, 1987.

[4] 이기웅, 실시간 시스템에서의 순위반전 대책 규약들의 성능분석, 석사학위논문, 한국과학기술원, 1994.

[5] 최동찬 외 3인, "차량 시뮬레이터의 가상 주행 환경을 위한 데이터베이스 및 실시간 그래픽스 엔진 개발," HCI2000 학술대회, pp. 498-503, 2000.

[6] 이운성 외 2인, "국민대학교 차량 시뮬레이터," 대한 기계학회 동역학 및 제어부문 2000년도 동계 workshop, pp. 42-47, 2000.

[7] 손권 외 2인, "건설 차량 실시간 그래픽 주행 시뮬레이터," 한국정밀공학회지, 제 16 권, 제 7 호, pp. 109-118, 1999.

[8] 이승호, 김성덕, "PC 기반형 자동차 운전 연습기 개발," 제어 · 자동화 · 시스템공학회지, 제 3 권, 제 4 호, pp. 415-421, 1997.

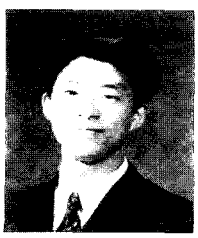
[9] 이정열, 자전거 시뮬레이터용 운동생성 알고리즘 개발, 석사학위논문, 한국과학기술원, 2000.

[10] Jeong-Yeol Lee et al., "Bicycle Dynamics for Bicycle Simulator," *The 2nd Japan-Korea Symposium of Fronties in Vibration Science and Technology*, Japan, pp. 50-51, 1999.

[11] Realtime Programming under QNX 4 Course Notes.

[12] W.Richard Stevens, Unix Network, 1992.

[13] 노병혁, 오성진, 유닉스 프로그래밍, 사이버출판사, 1998.



이 경 노

1973년 1월 8일생. 1996년 연세대학교 기계공학과, 공학사. 1998년 한국과학기술원 기계공학과, 공학석사. 1998년~현재 동대학원 박사과정. 관심분야는 로보틱스, 이동 로봇군 제어, 시뮬레이터, 가상현실, 이산이벤트시스템 제어.

이 두 용

1985년 서울대학교 제어계측공학과, 공학사. 1987년 Rensselaer Polytechnic Institute, M.S. 1993년 동 대학교, Ph.D. 1993년~1994년 동 대학교, Postdoctoral Research Associate. 1994년~현재 한국과학기술원 기계공학과 교수. 관심분야는 로보틱스, 생산시스템 최적화 및 제어, 가상현실, 이산이벤트시스템 제어.