논문 2021-16-32

이더넷 기반 실시간 통신 네트워크 프로토콜 구현

(Protocol Implementation for Ethernet-Based Real-Time Communication Network)

권 영 우, 응 우 옌 후 동, 최 준 영*
(Young-Woo Kwon, Dung Huy Nguyen, Joon-Young Choi)

Abstract : We propose a protocol for Ethernet-based industrial real-time communication networks. In the protocol, the master periodically transmits control frames to all slaves, and the ring-type network topology is selected to achieve high-speed transmission speed. The proposed protocol is implemented in the form of both firmware and Linux kernel modules. To improve the transmission speed, the MAC address table is disabled in the firmware implementation, and the NAPI function of the Ethernet driver is removed in the Linux kernel module implementation. A network experiment environment is built with four ARM processor-based embedded systems and network operation experiments are performed for various frame sizes. From the experimental results, it is verified that the proposed protocol normally operates, and the firmware implementation shows better transmission speed than the Linux kernel module implementation.

Keywords : Ethernet, Industrial network, Real-time communication

Ⅰ. 서 론

산업용 이더넷 네트워크는 비용, 효율적인 구현, 유연한 토폴로지, 고대역폭 및 광범위한 호환성과 같은 다양한 장점으로 인해 기존 필드버스 네트워크 대신 자동화 응용 분야에 널리 사용되고 있다. 이러한 추세는 2020년 전세계 산업용 네트워크 시장 점유율 통계에서도 증명이 되는데 산업용 이더넷이 64%, 필드버스가 30%의 점유율을 달성하고 있다. 현재 산업현장에서는 EtherCAT, Profinet IRT, Ethernet/IP, POWERLINK 등 다양한 상용 Ethernet 기반 산업용 네트워크 프로토콜이 사용되고 있으며 그 중 EtherCAT은 실시간성을 보장하면서 지연시간, 응답시간, 지터, 제어주기 측면에서도 가장 높은 성능을 나타내고 있다 [1-4].

특히 EtherCAT에서는 프레임이 네트워크를 이동하는 동안 각 슬레이브가 주소 지정된 데이터를 즉석에서 처리하는 'on-the-fly' 방식을 적용함으로써 프레임을 처리하는데 별도의 지연이 발생하지 않고 하드웨어 전송에서만 지연이 발생한다. 따라서 Fast Ethernet의 전이중 모드에서 최대 유효 데이터 전송률이 100 Mbps를 초과하며, 100 μs 이하의 짧은 주기로 고속 동작이 가능하여 EtherCAT은 고급 모션 제어 또는 전력 전자 분야의 매우 빠른 제어 루프에 사용되고 있다 [5-7].

그러나, EtherCAT은 마스터와 다수의 슬레이브로 구성되는 네트워크 구조로서 마스터는 운영체제기반에서 동작하는 프로그램으로 대부분 고가로 출시되고 있으며 고성능 컴퓨팅 시스템을 요구하고 있어 고비용과 복잡성을 초래하고 있다. 또한, 슬레이브는 EtherCAT 슬레이브 컨트롤러(EtherCAT slave controller, ESC)라는 특정 하드웨어가 필요한데, ESC는 FPGA, ASIC 형태로 구현되거나 전용 프로세서에 소프트웨어 형태로 구현되어 고비용과 복잡성을 증가시키고 있다 [8].

이러한 기존 EtherCAT의 단점을 극복하기 위해 본 논문에서는 특별한 운영체제 기반 마스터 프로그램 및 ESC 하드웨어가 필요 없는 소프트웨어 기반 이더넷 산업용 실시간 네트워크 프로토콜을 제안하고 제안된 프로토콜을 ARM 프로세서 기반 펌웨어와 리눅스 커널 모듈 형태로 각각 구현한다. 또한 4개의 상용 ARM 프로세서 기반 임베디드 보드로 구성되는 네트워크 하드웨어를 구성하고 구현된 펌웨어와 리눅스 모듈 형태의 프로토콜을 각각 적용하고 프로토콜 동작 실험을 수행하여 성능을 비교한다.

전체 논문의 구성은 다음과 같다. II 장에서는 실시간으로 동작할 수 있는 이더넷 기반 산업용 통신 네트워크 프로토콜을 제안한다. III 장에서는 제안된 프로토콜을 펌웨어 및 리눅스 커널 모듈 형태로 각각 구현할 때 필요한 절차 및 핵심 기술을 기술한다. IV 장에서는 상용 임베디드 보드로 구성된 네트워크 환경을 구축하고 구현된 프로토콜을 실험하고 실험 결과를 분석한다. V 장에서는 결론을 도출한다.

*Corresponding Author (jyc@pusan.ac.kr)
Received: Sep. 27, 2021, Revised: Oct. 9, 2021, Accepted: Nov. 16, 2021.
Y.W. Kwon: Hanwhasystems (Junior Engineer.)
D.H. Nguyen: Pusan National University (M.S. Student)
J.Y. Choi: Pusan National University (Prof.)
※ 본 연구는 산업통상자원부(MOTIE) 및 산업기술평가관리원(KEIT) 연구비 지원에 의한 연구임 (No. 20012815).

ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2021.16.6.247

Ⅱ. 이더넷 기반 실시간 네트워크 프로토콜 설계

이더넷은 고속 데이터 전송이 가능하면서 프로토콜이 간단하고 유연하며 효율이 높아 세계에서 가장 널리 채택된 네트워크 프로토콜이다. 더욱이 이더넷은 구현 비용 또한 저렴하기 때문에 산업 환경에서 널리 사용될 수 있는 유망한 네트워크 기술로 알려져 있다.

그러나 산업 환경에서 표준 이더넷을 사용할 때의 근본적인 문제는 데이터 라우팅 및 전달을 위한 TCP/IP 프로토콜이 산업 자동화 및 처리 응용프로그램에서 요구되는 시간 결정성, 실시간성을 제공하지 않는다는 것이다. 그러나 이더넷 네트워크에 사용되는 물리적 이더넷 장치는 충분히 활용 가능하였고 기계 컨트롤러, 센서, 액추에이터 및 기타 장치 간에 결정론적 실시간 통신을 제공하도록 이더넷의 물리적 계층 기반으로 다양한 산업용 이더넷이 개발되었다 [9].

현재 산업용 네트워크에서 사용되고 있는 실시간 이더넷 프로토콜들은 EtherCAT, EtherNet/IP, PROFINET, POWERLINK, Sercos III 등이 포함되며 이 중 EtherCAT은 우수한 실시간 성능을 나타내고 있다. 그러나 EtherCAT은 표준 이더넷 하드웨어를 기반으로 결정성을 보장하기 운영체제 기반 마스터 프로그램과 부가적인 슬레이브 하드웨어를 사용하는 아키텍처를 채택하고 있어 비용과 복잡성이 높은 단점이 있다. 이러한 문제를 해결하기 위해 부가적인 슬레이브 하드웨어 필요 없이 기존 이더넷 장치를 활용하는 실시간 프로토콜을 다음과 같이 제안한다.

1. 네트워크 구성 요소

네트워크는 단일 마스터와 다수의 슬레이브 노드로 구성된다. 마스터는 산업용 제어 알고리즘이 동작하는 노드로서 모든 슬레이브가 공유하는 프레임을 생성하고 전송을 시작하는 역할을 수행한다. 각 슬레이브는 마스터가 전송한 프레임을 수신하고 다른 슬레이브에 전송하는 역할을 수행한다. 즉, 마스터는 산업용 제어 네트워크에서 전체 슬레이브 시스템을 제어 및 관리하는 역할을 수행하기 위해서 각 슬레이브에서 필요한 정보를 포함하는 프레임을 생성하여 전송하고 각 슬레이브는 마스터가 전송한 프레임을 수신하여 각 슬레이브에서 필요한 정보를 획득하고 마스터 또는 다른 슬레이브에서 필요한 정보를 프레임에 추가하여 마스터 또는 다른 슬레이브에 전송한다.

2. 네트워크 토폴로지

일반적인 이더넷 통신의 경우 버스 및 스타 토폴로지를 많이 사용하며 프레임을 원하는 목적지로 보내기 위해 주소를 처리하는 과정이 필요한데 이로 인하여 주소 처리 과정에서 상당한 지연이 발생한다. 따라서 일반적인 이더넷 프레임 전송 방법을 사용하면 고속의 전송 속도를 얻는 것이 어려워 산업용 실시간 고속 제어 분야에 적용하는 것은 적합하지 않다.

이러한 단점을 극복하기 위해 본 논문에서는 이더넷 프레

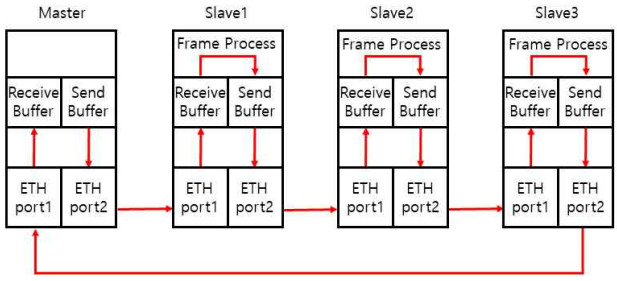

그림 1. 링 토폴로지의 동작 과정
Fig. 1. Operation process of ring topology

임의 주소 처리가 필요 없는 그림 1과 같은 링 형태의 네트워크 토폴로지를 제안한다. 즉 마스터와 슬레이브를 포함하는 모든 네트워크 노드는 하드웨어적으로 2개의 이더넷 포트를 지원하고 그림 1과 같은 링 형태로 네트워크를 구성하면 전송 노드와 수신 노드가 항상 일대일로 연결된다. 이러한 연결 구조에서 항상 방송 주소로 프레임을 전송하면 주소 처리를 할 필요 없이 즉시 프레임이 목적지로 전송되고 주소 처리 지연이 제거되어 고속의 전송 속도 달성이 가능하게 된다.

3. 프레임 전송 규칙

실시간 산업용 통신 네트워크의 주요 목적은 다수의 산업용 장치를 동기화하여 제어하는 것이므로 프로토콜의 기본 동작 방식은 미리 정해진 크기의 프레임을 주기적으로 전송하고 모든 슬레이브를 통과한 후 마스터에 프레임이 되돌아오면 한 주기의 동작이 완료된다.

마스터와 3개의 슬레이브로 구성된 네트워크를 나타내는 그림 1을 통해 프레임 전송 방법을 자세히 설명하면 다음과 같다. 마스터는 매주기 전송 각 슬레이브에서 필요한 정보를 포함하는 프레임을 생성하여 2번 포트를 통해 슬레이브 1의 1번 포트로 전송하고 슬레이브 1은 수신된 프레임에서 필요한 정보를 획득 및 추가하여 2번 포트를 통해 슬레이브 2의 1번 포트로 전송한다. 이러한 과정을 반복하면 마스터에서 전송된 프레임은 최종 슬레이브 3까지 도착을 하게 되고 슬레이브 3은 2번 포트를 통해 프레임을 다시 마스터로 전송하며 전체 네트워크 동작의 한 주기를 완료하게 된다. 이러한 과정을 통해 매 주기 마스터는 각 슬레이브에서 필요한 제어 정보를 전달하고 각 슬레이브는 마스터 또는 다른 슬레이브에서 필요한 제어 정보를 수신 프레임에 추가하여 목적지에 전송한다.

Ⅲ. 이더넷 기반 실시간 네트워크 프로토콜 구현

II 장에서 설계된 이더넷 기반 실시간 네트워크 프로토콜을 ARM 프로세서 기반의 임베디드 시스템에서 펌웨어 및 리눅스 커널 모듈 형태로 구현한다. 펌웨어 형태 구현 방법을 제시함으로써 자원이 부족한 저성능의 임베디드 시스템에서도 활용이 가능하다. 또한 리눅스 기반의 프로토콜 구

현 방법을 제시함으로써 오픈 소스 프로그램 환경의 장점 및 범용성을 충분히 활용하고 커널 모듈 형태로 구현함으로써 높은 실시간 데이터 전송 속도를 확보한다.

1. 펌웨어 형태 프로토콜 구현

2개의 이더넷 포트를 지원하는 ARM 프로세서 기반 임베디드 시스템에서 동작하도록 제안된 프로토콜을 구현하는데 프로세서의 레지스터를 조작하는 경우가 많기 때문에 대부분 어셈블리어로 구현이 된다. 펌웨어 형태로 프로토콜을 구현할 때 필요한 핵심적인 단계를 기술하면 다음과 같다.

첫 번째 이더넷 프레임을 수신할 때 인터럽트가 발생하도록 벡터 테이블을 생성하고 인터럽트 발생 시 수행 동작을 Interrupt service routine (ISR)으로 등록한다. 코프로세서의 CP15 레지스터에 벡터 테이블 주소를 등록하고 ARM 프로세서의 IRQ 모드에서 인터럽트가 발생할 수 있도록 활성화하고 ARM 프로세서의 이더넷 프레임 수신 인터럽트 번호를 확인하여 작성된 ISR을 등록한다. ISR의 핵심 동작 내용은 수신된 프레임에서 필요한 정보를 획득 및 추가하고 다른 이더넷 포트를 통해 방송 주소로 변경된 프레임을 전송하는 것이다.

두 번째 이더넷 제어기에 대한 초기화 및 설정이 필요하다. 특히 이더넷 제어기의 초기화 과정 시 프레임 송수신 속도를 향상시키기 위해 MAC 주소 테이블 비활성화 및 DMA 설정을 필수적으로 수행해야 한다. 일반적으로 MAC 주소 테이블은 LAN에 연결된 다수 이더넷 장치의 MAC 주소를 관리하여 프레임 송수신의 효율을 제고하기 위해 사용되지만 제안된 프로토콜에서는 모든 포트가 일대일로 연결되어 있으므로 MAC 주소 테이블을 사용할 필요가 없다. 따라서 MAC 주소 테이블을 비활성화하고 이로 인하여 관련된 처리 과정이 필요 없게 되어 프레임 송수신 속도가 증가한다. 또한 메모리에 저장된 송수신 프레임과 이더넷 제어기의 송수신 버퍼 사이의 데이터 전송 속도를 높이기 위해 인터럽트 기반 DMA 장치의 사용이 필수적이며 이로 인하여 이더넷 전송 속도가 향상되는 효과를 얻을 수 있다.

세 번째 네트워크 전송 속도를 정밀하게 측정할 수 있는 하드웨어 기반 클럭의 설정이 필요하다. ARM 기반 프로세서의 Cycle Count (CCNT) 레지스터는 System Control Coprocessor의 레지스터이며 프로세서의 클럭 사이클에 따라 레지스터의 값이 1씩 증가하는 카운터이다. 따라서 프로세서의 클럭 주파수를 알고 있다면 CCNT 레지스터를 이용하여 고해상도의 시간을 측정할 수 있다.

이외에 펌웨어 형태 구현을 위해 추가적인 절차가 필요하나 일반적인 사항으로 자세히 기술하지는 않고 대신 그림 2에 전체 절차의 개요 및 순서를 나타낸다.

2. 리눅스 커널 모듈 형태 프로토콜 구현

일반적인 리눅스 운영체제 환경에서 제안된 프로토콜이 동작할 수 있도록 기존 리눅스 이더넷 드라이버를 기반으로 프로토콜을 구현한다. 한편 기존 리눅스 이더넷 드라이버의

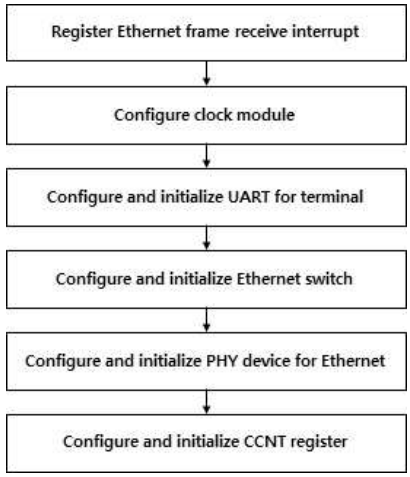

그림 2. 펌웨어 형태 프로토콜 초기화 과정
Fig. 2. Firmware protocol initialization process

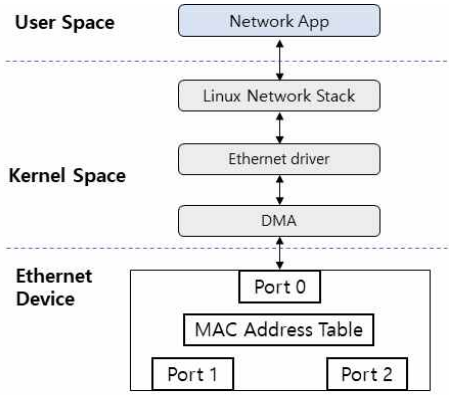

그림 3. 일반적인 리눅스 프레임 전송 구조
Fig. 3. Typical Linux frame transmission structure

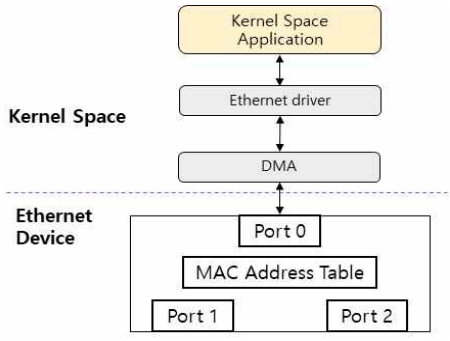

그림 4. 커널 모듈 프로토콜의 프레임 전송 구조
Fig. 4. Kernel module protocol frame transmission structure

단점을 극복하고 전송 속도를 향상시키기 위해 다음과 같은 두 가지 접근 방법을 추구한다.

첫 번째 응용프로그램과 이더넷 드라이버 사이에 불필요한 데이터 복사를 방지하기 위해 제안된 프로토콜을 리눅스 커널 모듈 형태로 구현한다. 그림 3에서 알 수 있듯이 일반

적인 리눅스 환경에서 이더넷 프레임 송수신 과정은 이더넷 제어기, 커널 영역, 사용자 영역을 통하여 처리되므로 프레임을 처리하는 과정에서 커널 영역과 사용자 영역 사이에 메모리 데이터 복사가 발생하고 처리시간이 증가하는 단점이 있다. 이러한 문제를 해결하기 위해 그림 4와 같이 커널 영역에서 직접 프레임 데이터를 처리할 수 있도록 프로토콜을 커널 모듈 형태로 구현하여 사용자 영역과의 상호 데이터 복사를 방지하여 전송 속도 향상 효과를 얻을 수 있다.

두 번째 기존의 이더넷 드라이버의 경우 프레임의 수신 시 NAPI 방식을 사용하는데 이 방식의 경우 프레임을 수신할 때마다 인터럽트를 발생시켜 프레임의 데이터를 획득하지 않고 수신 버퍼에 프레임을 저장해 두었다가 적절한 스케줄링을 통하여 폴링 방식으로 다수의 프레임을 한 번에 처리한다 [10]. 이러한 처리방식은 프레임마다 수신 즉시 처리해야 하는 실시간 네트워크에는 적합하지 않기 때문에 NAPI 기능을 제거하고 매 프레임 수신 시 즉시 인터럽트가 발생하여 프레임의 데이터 처리가 가능하도록 기존 이더넷 드라이버를 수정한다. 이러한 수정을 통하여 이더넷 프레임 수신에 대한 처리 시간이 감소하고 전체적인 전송 속도를 향상시키는 효과를 얻을 수 있다.

Ⅳ. 실 험

1. 실험 환경 구축

제안된 프로토콜의 유효성 및 성능을 검증하기 위해 1개의 마스터와 3개의 슬레이브로 구성되는 네트워크를 구축하고 프레임 전송 실험을 수행하며 실험 결과를 분석한다. 실험에서 마스터와 슬레이브로 사용되는 임베디드 보드는 TI 사의 AM3359 Industrial Communications Engine (ICE) v2 보드를 채택하고 프로세서의 주파수는 800MHz로 동작하도록 설정한다. 리눅스 커널 모듈 형태 구현의 경우 리눅스 운영체제는 RT 패치가 적용된 Linux 커널 4.9.59를 사용한다. ICE 보드에 장착된 AM3359 MPU는 2개의 이더넷 포트를 지원하는 이더넷 제어기를 주변장치로 포함하고 있어 2개의 이더넷 포트를 지원하는 임베디드 보드를 제작할 때 유용하고 ICE 보드도 2개의 외부 이더넷 포트를 지원하도록 제작되어 있다.

총 4대의 ICE 보드를 그림 5와 같은 링 토폴로지로 연결하고 각 보드에서 다른 보드로 전송하는 프레임의 구조는 일반적인 이더넷 프레임의 구조로 헤더와 페이로드로 구성되며 목적지 주소에는 방송 주소인 FF:FF:FF:FF:FF:FF를 적용한다.

전송 속도 성능 측정 방법은 다음과 같다. 마스터는 프레임을 생성하여 첫 번째 슬레이브로 보내고 각 슬레이브는 연결된 다음 슬레이브로 프레임을 전송하며 마지막 슬레이브는 프레임을 다시 마스터로 전송하고 한 주기가 완료된다. 단일 주기 시간을 정밀하게 측정하기 위해 CCNT 레지스터를 사용하며 마스터에서 프레임을 송신하기 직전 CCNT 레지스터의 시간 값을 읽어 송신시간을 저장하고 프

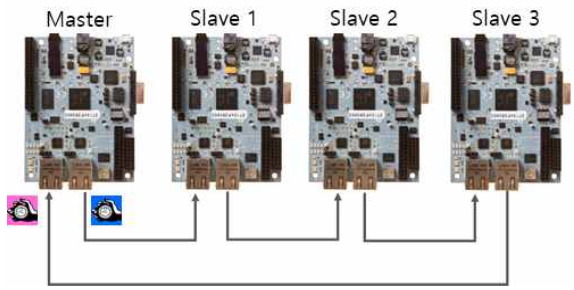

그림 5. ICE 보드 4대로 구성된 네트워크 실험 환경
Fig. 5. Network experimental environment consisting of 4 ICE boards

레임 송신 후 모든 슬레이브를 통과하고 마스터에 프레임이 다시 수신될 때 인터럽트 루틴에 의해 CCNT레지스터의 시간을 다시 읽어 수신시간을 저장한다. 수신시간과 송신시간의 차이를 계산하면 한 주기 동안 소요된 프레임의 전송시간을 정확하게 측정할 수 있다.

또한 프레임의 크기에 따라 전송 시간이 달라지는데 그 양상을 관찰하기 위해 64 Bytes, 128 Bytes, 256 Bytes의 3가지 프레임 크기로 실험을 수행하고 실험마다 마스터에서 모든 슬레이브를 통과하여 다시 마스터로 프레임이 도착하기까지 걸리는 시간을 1000번 측정하여 평균값을 계산하여 실험 결과를 획득한다.

2. 실험 결과

펌웨어와 리눅스 커널 모듈 형태의 전송 실험 결과는 표 1과 같이 나타나며 펌웨어의 경우 64 Bytes, 128 Bytes, 256 Bytes의 경우 각각 88 μs, 114 μs, 164 μs의 프레임 전송 속도가 측정되었으며 리눅스 커널 모듈의 경우 각각 143 μs, 212 μs, 264 μs로 측정되었다. 이러한 결과로부터 구현된 프로토콜의 정상적인 동작을 검증할 수 있고 펌웨어 형태의 구현 방법이 리눅스 커널 모듈 형태보다 우수한 성능을 달성한 것을 확인할 수 있다. 이러한 성능 차이에 대한 원인은 커널 모듈 형태 구현 방법이 비록 리눅스 환경에서 커널 모듈로 프로토콜을 구현하고 이더넷 드라이버의 NAPI 기능도 제거 했지만 이더넷 제어기 기능 처리 이외에도 다양한 리눅스 운영체제 자체의 처리 루틴으로 인해 이더넷 제어기 처리만 필요한 펌웨어 형태의 구현 방법보다 성능이 낮은 것으로 판단할 수 있다.

표 1. 구현 방법과 프레임 크기에 따른 전송 시간 결과
Table 1. Frame transmission time results according to implementation type and frame size

Implementation	Frame Size (Bytes)		
	64	128	256
Firmware	88 μs	114 μs	164 μs
Linux Module	143 μs min.:113 μs max.:167 μs	212 μs min.:187 μs max.:233 μs	264 μs min.:238 μs max.:298 μs

V. 결 론

본 논문에서는 이더넷 기반 산업용 실시간 통신 네트워크를 위한 프로토콜을 제안하였다. 2개의 이더넷 포트를 지원하는 ARM 프로세서 임베디드 시스템을 기반으로 고속 전송 속도를 달성하기 위해 링 형태의 네트워크 토폴로지를 채택하였다. 또한 제안된 프로토콜을 펌웨어 및 리눅스 커널 모듈 형태로 구현하였다. 전송 속도를 향상시키기 위해 펌웨어 형태 구현에서는 MAC 주소 테이블을 비활성화하고 이더넷 제어기 프레임 버퍼와 메모리 사이에 DMA를 적용하였고 리눅스 커널 모듈 형태 구현에서는 이더넷 드라이버의 NAPI 기능을 제거하였다. 4개의 ARM 프로세서 기반 임베디드 시스템으로 네트워크를 구성하고 다양한 프레임 크기에 대하여 전송 실험을 수행하였고, 실험 결과로부터 제안된 프로토콜의 정상적 동작과 펌웨어 형태의 구현이 리눅스 커널 모듈 형태의 구현보다 전송 속도가 우수한 것을 검증하였다.

개발된 실시간 네트워크 프로토콜은 부가적인 하드웨어의 필요 없이 일반적인 이더넷 환경에서 동작할 수 있고 펌웨어 형태의 구현도 가능하므로 자원이 빈약한 저사양 저비용 임베디드 시스템에서도 충분히 적용 가능하여 다양한 활용 방안을 기대할 수 있다.

References

[1] P. Ferrari, A. Flammini, D. Marioli, A. Taroni, F. Venturini, “Experimental Analysis to Estimate Jitter in PROFINET IO Class 1 Networks,” Proceedings of IEEE Conf. Emerg. Technol. Factory Autom., pp. 429-432, Sep. 2006.

[2] G. S. Sestito, A. C. Turcato, A. L. Dias, M. S. Rocha, M. M. Da Silva, P. Ferrari, D. Brandao, “A Method for Anomalies Detection in Real-time Ethernet data Trafc Applied to PROFINET,” IEEE Trans. Ind. Informat., Vol. 14, No. 5, pp. 2171-2180, May 2018.

[3] K. O. Akpinar, I. Ozcelik, “Analysis of Machine Learning Methods in EtherCAT-based Anomaly Detection,” IEEE Access, Vol. 7, pp. 184365-184374, Dec. 2019.

[4] https://www.hms-networks.com/news-and-insights/news-from-hms/2020/05/29/industrial-network-market-shares-2020-according-to-hms-networks

[5] G. Prytz, “A Performance Analysis of EtherCAT and PROFINET IRT,” Proceedings of IEEE Int. Conf. Emerg. Technol. Factory Autom., pp. 408-415, Sep. 2008.

[6] G. Cena, I. C. Bertolotti, S. Scanzio, A. Valenzano, C. Zunino, “On the Accuracy of the Distributed Clock Mechanism in EtherCAT,” Proceedings of IEEE Int. Workshop Factory Commun. Syst., pp. 43-52, May 2010.

[7] H. Kang, K. Kim, H. W. Jin, “Real-time Software Pipelining for Multidomain Motion Controllers,” IEEE Trans. Ind. Informat., Vol. 12, No. 2, pp. 705-715, Apr. 2016.

[8] G. Prytz, “A Performance Analysis of EtherCAT and PROFINET IRT,” Proceedings of IEEE Conf. Emerg. Technol. Factory Autom., pp. 408-415, 2008.

[9] https://www.motioncontroltips.com/what-is-industrial-ethernet-and-how-does-it-differ-from-standard-ethernet/

[10] K. Zhu, X. Chen, J. Tan, “A Method of Adaptive Selection of Hybrid Interrupt-NAPI Scheme,” Proceedings of International Conference on Communications and Intelligence Information Security, pp.125-129, 2010.

Young-Woo Kwon (권 영 우)

2019 Electronics Engineering from Pusan National University (B.S.)

2021 Electronics Engineering from Pusan National University (M.S.)

2021~Department of Production Technology at Hanwhasystems (Junior Engineer)

Field of Interests: Embedded system and control system

Email: youngwoo94.kwon@hanwha.com

Dung Huy Nguyen (응우옌 후 동)

2017 Automation and Control Engineering from Hanoi University of Science and Technology, Hanoi, Viet Nam (B.S.)

2020~Electronics Engineering from Pusan National University (M.S. candidate)

Field of Interests: Embedded system and control system

Email: leegaram2019@pusan.ac.kr

Joon-Young Choi (최 준 영)

1994 Electronics and Electric Engineering from Pohang University of Science and Technology (B.S.)

1996 Electronics and Electric Engineering from Pohang University of Science and Technology (M.S.)

2002 Electronics and Electric Engineering from Pohang University of Science and Technology (Ph.D.)

2005~Department of Electronics Engineering at Pusan National University, Busan, Korea (Prof.)

Field of Interests: Embedded system and control system

Email: jyc@pusan.ac.kr