파일 시스템을 이용한 효율적인 임베디드 네트워크 I/O 가상화

김종서° 장혜천 진현욱
건국대학교 컴퓨터공학부
{flyingv, comfact, jinh}@konkuk.ac.kr

Efficient Embedded Network I/O Virtualization Using File System

Jong-Seo Kim°, Hye-Churn Jang, Hyun-Wook Jin
Department of Computer Science and Engineering, Konkuk University

1. 서론

기존의 가상화 기술은 고성능 컴퓨팅 자원을 효율적으로 사용하기 위해 개발되었다 [1, 2]. 현재 이러한 가상화 기술은 임베디드 및 산업 시스템에서도 물리적인 공간을 줄이고 다양한 장치 운용환경을 효율적으로 지원하기 위한 목적으로도 그 영역을 넓혀 가고 있다. 그러나 현재의 Virtual Machine Monitor (VMM)는 다양한 임베디드 네트워크 장치들이 갖는 특성을 아직 일반화하지 못하고 있다. 그 결과 게스트 도메인에서 임베디드 네트워크를 사용하기 위해서는 호스트 도메인에서 응용 수준의 중계 기법[3]을 사용하여야 하는데, 추가적인 데이터 복사와 불필요한 네트워크 처리 과정이 발생하여 성능 저하가 초래된다.

임베디드 네트워크 장치들은 일반적으로 문자형 장치 드라이버 (Character Device Driver)로 구현되어 있으며, 대부분의 장치 의존적인 작업들은 IOCTL(I/O Control System call)을 통해서 구현되어 있다. 그러나 이러한 문자형 장치 드라이버 기반의 임베디드 네트워크 장치들을 지원하기 위한 고려는 아직 가상화 환경에서 제대로 이루어지지 못하고 있다. 따라서 게스트 도메인으로부터 장치 제어 명령을 호스트 도메인의 임베디드 네트워크 장치로 전달하기 위해서는 이를 위한 중계 어플리케이션을 구현 하여 사용해야 한다.

기존 기술들의 이러한 문제점을 개선하고자 본 논문은 대부분의 가상화 기술들이 지원하는 공유 폴더 기능과 리눅스 파일 시스템 기반 장치제어 방식을 융합하여 효율적인 임베디드 네트워크 I/O 가상화를 제안한다. 제안된 기법은 임베디드 시스템에 적합한 전가상화 (Full Virtualization) 기법을 사용하는 VirtualBox에 구현하고 그 성능을 CAN(Control Area Network)환경에서 측정한다. 제안된 기법은 게스트 운영체제의 커널과 임베디드 네트워크를 사용하는 기존 응용프로그램의 수정을 요구하지 않는다. 측정 결과 제안된 기법은 기존의 중계 어플리케이션 방식보다 약 35%의 성능을 향상시킬 수 있음을 보인다.

2. 파일 시스템 기반의 네트워크 I/O 가상화

게스트 도메인에서 호스트 도메인의 임베디드 네트워크 장치를 이용하기 위해서는 Host-only Networking을 이용하는 중계 어플리케이션을 구현하여야 한다. 이러한 Host-only Networking은 안정적이고 확장이 용이한 소켓을 이용하며 이는 게스트 도메인과 호스트 도메인을 연결하기에 좋은 구조이지만 불필요한 오버헤드가 발생한다. 또한 게스트 도메인이 전달한 데이터를 호스트 도메인에서 중계 역할을 하는 중계 어플리케이션이 실제 장치 드라이버로 다시 전달하는 과정이 추가적으로 필요하다. 이것은 그림1 에서 보이듯이 처리 계층을 더욱 복잡하게 만들기 때문에 데이터 복사 문제와 도메인 간 스케줄 타이밍 문제 등을 야기 할 수 있다.

파일시스템 기반의 네트워크 I/O 가상화는 많은 가상화 장치가 제공하는 공유 폴더 추가기능을 통하여 구현 될 수 있다. 일반적으로 공유 폴더는 호스트 도메인에서 지정한 파일 시스템을 게스트 도메인에서 접근 할 수 있게 해주는 기능이다. 이 기능을 확장하여 게스트 도메인에서 실행되는 응용 프로그램은 호스트 도메인의 장치 파일에 접근하여 파일 명령을 통해 장치를 제어 할 수 있다. 이 기법은 Passthrough I/O[4]처럼 I/O성능을 크게 향상시킬 수 있고 게스트 및 호스트 운영체제와 디바이스 드라이버의 수정이 필요 하지 않기 때문에 적용이 쉽다는 장점이 있다.

VirtualBox가 제공하는 공유 폴더 기능은 게스트 도메인에 가상의 파일시스템으로 등록된다. 파일에 대한 기본적인 읽기와 쓰기 작업 및 탐색, 잠금 등의 기본적인 기능을 제공하지만 파일 시스템에서 IOCTL은 필요하지 않으므로 구현되어 있지 않다. 기존 VirtualBox 공유폴더 기능에서 파일 읽기, 쓰기 기능은 내부 버퍼를 할당한 후 VirtualBox의 HGCM(Host Guest Communication Manager)을 통해 전달한다. 쓰기의 경우 버퍼에 담긴 데이터를 실제 파일에 출력만 하지만, 읽기의 경우 버퍼에 실제 데이터를 받아와서 가상 파일시스템을 호출한 게스트 도메인의 응용 프로그램까지 전달하여야 한다. 본 논문에서는 HGCM을 확장하여 게스트 도메인에서 실행되는 응용 프로그램이 장치 파일에 접근하여 임베디드 네트워크를 직접 제어할 수 있도록 한다. 이를 위해서는 대부분의 임베디드 네트워크 드라이버가 구현하고 있는 IOCTL을 HGCM에서 지원해야 한다. IOCTL의 경우 호출 시 명령 값과 인자 값이 전달된다. 이때 인자 값이 일반적인 값인지, 메모리 번지인지에 따라 HGCM Call을 통과하는

본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT 연구센터 지원사업(NIPA-2010-C1090-1031-0003)과 한국전자통신연구원 위탁과제(7010-2010-0079)의 연구결과로 수행되었음.

처리가 크게 달라 지기 때문에 전달받은 명령 값에 따라 인자 값의 종류를 구분하여야 한다. 만약 인자가 일반 값이라면 값만을 전달 하고, 인자가 메모리 번지일 경우 앞서 설명한 파일 읽기 또는 쓰기 과정에서 수행하는 처리과정을 수행하여야 한다. 쓰기 과정처럼 전달받은 데이터를 실제 파일시스템의 IOCTL을 호출할 때 전달하여 주어야 하고, 버퍼에 결과 값을 기록하는 경우를 대비해 해당 버퍼를 읽기 과정처럼 게스트 도메인의 응용 프로그램까지 전달하도록 구현하여야 한다. HGCM Call을 통하여 VirtualBox로 전달된 IOCTL의 명령 값과 인자 값의 경우 각 명령 값에 따라 인자 값의 데이터 형식과 크기가 달라질 수 있으므로, 각 명령 값을 구분하여 인자 값 처리를 결정하는 부분이 추가 되었다.

그림 2은 VirtualBox에서 공유 폴더 기능을 사용하여 게스트 도메인의 응용 프로그램이 CAN 네트워크를 제어하는 그림을 나타내었다. 그림2 에서 보이듯 데이터 경로를 단순화시키고 중계 어플리케이션을 위한 두 번의 복사 과정이 줄어들게 된다.

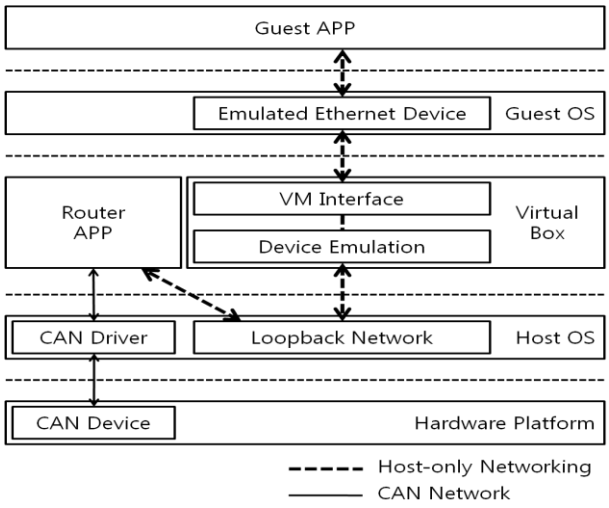

그림1. 중계 어플리케이션을 이용한 CAN 장치제어

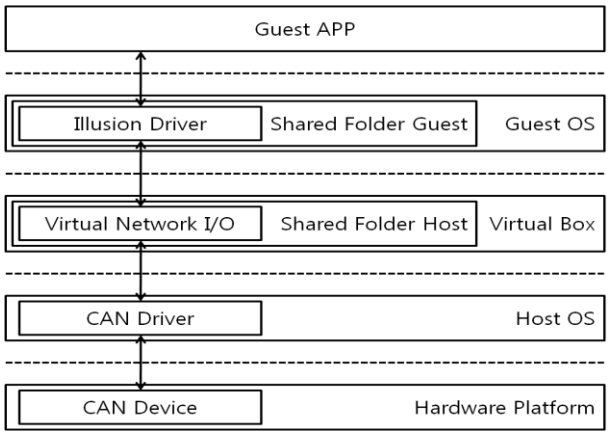

그림2. 파일 시스템 기반의 네트워크 I/O 가상화

3. 성능측정

구현된 파일 시스템 기반의 네트워크 I/O 가상화 방식의 성능 분석을 위하여 두 CAN 종단 노드 간 데이터의 왕복 시간(Round-trip Time)을 중계어플리케이션 환경, 파일 시스템 기반의 네트워크 I/O 가상화 환경 그리고 비 가상화 환경에서 각각 측정하였다. 그 결과 파일 시스템 기반의 기법은 중계 어플리케이션 방식보다 약 35% 정도의 성능 향상을 보여준다. 또한 비 가상화 환경보다 중계 어플리케이션 방법은 42%의 성능 저하를 나타낸 반면, 파일 시스템 기반의 방식은 10% 정도만 성능 저하를 보였다. 이는 가상화 장치 및 운영체제의 계층을 통과하기 위한 횟수의 감소와 스케줄링 영향의 감소 및 Host-only Networking의 TCP/IP 처리 과정의 생략에서 발생한 성능 증가로 판단된다.

4. 결론 및 향후 계획

본 논문에서 제안된 임베디드 가상화 환경을 위한 파일 시스템 기반의 네트워크 I/O 가상화 방법은 간편한 적용과 비교적 적은 추가 부하로 가상화 환경에서 호스트 도메인의 임베디드 네트워크 장치를 제어 할 수 있음을 보였다. 측정 결과 제안된 기법은 기존의 중계 어플리케이션 방식보다 약 35%의 성능을 향상시킬 수 있음을 보였다. 향후 계획으로는 본 논문에서 제안한 방법을 기반으로 공유 폴더 방식에서 최적화 과정을 거쳐 가상화 환경으로 인한 추가 부하를 줄이고, 다양한 임베디드 네트워크 장치를 지원 할 예정이다.

참고문헌

[1] Mendel Rosenblum and Tal Garfinkel, " Virtual Machine Monitors: current Technology and Future Trends," Computer, Vol. 38, Issue 5, pp. 39-47, May 2005.

[2] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield " Xen and the Art of Virtualization," In Proc. of the 19th ACM Symposium on Operating Systems Principles, 2003.

[3] Sung-Moon Chung and Hyun-Wook Jin, " Isolationg System Faults on Vehicular Network Gateways Using Virtualization," In Proc. of The Sixth IEEE/IFIP International Symposium on Trusted Computing and Communications (TrustCom 2010), December 2010.

[4] M.Tim Jones " Linux Virtualization and PCI Passthrough," developerWorks, IBM Corporation, October 2009.