KIPS Tr. Comp. and Comm. Sys.
Vol.3, No.12 pp.443~448 pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2014.3.12.443

The Design of Fault Tolerant PSTR Using Virtualization Techniques on the Embedded System

Jinho Yoo† · Kyujong Han††

ABSTRACT

This paper is a study related to fault tolerant design based on PSTR using virtualization techniques. If the fault tolerant PSTR based on virtualization techniques is implemented the communication performance between primary and shadow will improves and monitoring function is easy to available about activities of primary and shadow. The legacy PSTR model is implemented in its hardware. The primary play a main role and shadow play a switched action when the errors occurrs in the primary. The switched action of shadow make it possible to restart the primary function newly. This paper implements fault tolerant primary-shadow model using virtualization techniques on the embedded environment.

Keywords : Virtualization, Communication, Virtual Machine, Virtual Device, Sharing Device

가상화 기술을 이용한 임베디드 시스템상의 고장감내 PSTR 설계

유 진 호† · 한 규 종††

요 약

본 논문은 가상화 기술을 이용하여 PSTR에 기반한 고장감내 설계에 관련한 연구이다. 고장감내 PSTR 모델을 가상화 기술에 기반하여 구현하게 되면 프라이머리와 쉐도우 간의 통신성능이 향상되고 각 프라이머리와 쉐도우 내의 동작에 대한 모니터링이 용이하게 된다. 기존의 PSTR 모델은 프라이머리 하드웨어와 쉐도우 하드웨어 구성을 전제로 하여 프라이머리에서 본 임무를 수행하고 쉐도우에서는 프라이머리에서 고장이 발생했을 경우 취해야 할 동작을 준비하게 된다. 쉐도우에서 이러한 동작을 준비하므로 임무에 차질이 없도록 프라이머리는 다시 주 임무를 수행할 수 있게 된다. 본 논문에서는 임베디드 환경에서 고장감내 PSTR 모델을 가상화 기술을 이용하여 구현한다.

키워드 : 가상화, 통신, 가상 머신, 가상장치, 장치공유

1. 서 론

컴퓨터 하드웨어가 급속하게 발전함에 따라 컴퓨터에서 수행되는 소프트웨어에 비해 하드웨어 자원의 성능이 월등해졌다. 가상화 기술은 이렇듯 월등한 하드웨어 자원의 가용성을 높여 하드웨어 사용을 최대화하고 비용을 절감할 수 있는 소프트웨어 기술이다[1].

본 연구는 PSTR(Primary-Shadow TMO Replication) 모델을 기반으로 고장감내를 구현하는 시스템 모델을 가상화 기술을 바탕으로 구현한다. 기존의 하드웨어 환경으로 구성된 PSTR 고장감내 환경을 가상머신상에서 소프트웨어로 구성함에 따라 시스템 간의 통신, 시스템 구성 전반에 걸쳐 소프트웨어 특성에 맞는 효율적인 방법을 찾을 수 있다. 본 연구에서는 시스템 간 통신방법, 입출력 장치의 공유, 프로세서 부하 균형, PSTR 처리절차를 중심으로 고장감내 시스템을 설계하고 시스템 구현실험을 통해 살펴보도록 한다.

본 연구의 시스템은 프라이머리와 쉐도우라는 두 개의 컴퓨팅 개체로 구성되어있다. 프라이머리는 본 임무를 수행하기 위한 기능을 가지는 컴퓨팅 개체이고, 쉐도우는 프라이머리의 고장 시 사용하게 되는 대기모드의 컴퓨팅 개체이다. 프라이머리에서 본 임무를 수행하다가 프라이머리에서 오류가 발생하게 되면 본 임무 수행에 큰 영향을 주게 되므로 쉐도우로 절체가 일어난다. 쉐도우는 대체동작을 수행하고 프라이머리를 다시 기동시키게 되며 프라이머리가 정상화되면 쉐도우는 프라이머리에 제어권을 넘기고 프라이머리는 본 임무를 계속하게 된다. 이러한 기능 구현을 위해 본 연구에서는 시스템 간의 효율적인 통신방법을 제시하고 입출력 장치의 공유, PSTR 모델 처리절차에 대해서 논한다.

※ 본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었음.
† 정 회 원 : 백석대학교 정보통신학부 교수
†† 비 회 원 : 건국대학교 인터넷미디어공학과 석사과정
Manuscript Received : September 16, 2014
First Revision : October 14, 2014
Accepted : October 16, 2014
* Corresponding Author : Jinho Yoo(yoojh@bu.ac.kr)

2절에서는 본 연구의 배경이 되는 PSTR 모델을 설명하고 관련 연구들을 살펴본다. 3절에서는 PSTR 모델이 구현된 가상 머신 시스템의 관리모듈 구성을 논하고, 직렬장치의 입출력 동시 공유와 서로 다른 가상머신상의 태스크 간 통신에 대해 기존의 하드웨어 통신환경에서 사용했던 소켓 통신을 개선하여 가상입출력 장치를 제안하고 구현하였다. 또한 3절에서 구현한 기능에 대한 성능평가 실험 결과를 분석하고 4절에서는 실험과 시연에 대해서 논한다. 마지막으로 해당 연구 내용에 대한 결론을 기술할 것이다.

2. 관련 연구와 배경

본 절에서는 관련 연구들을 살펴보고 고장감내 구현을 위한 PSTR 모델에 대해서 설명하도록 한다.

2.1 관련 연구들

기존의 많은 시스템에 가상화 기술이 적용되어 하드웨어의 효율적 사용을 위해 적용되고 있다. 실제 하드웨어를 소프트웨어로 구현함에 따라 하드웨어가 직접적으로 제공하는 장치의 성능이 충분히 지원될 것인가에 대한 I/O 성능지연에 관련한 연구가 있다[2]. 이는 I/O성능이 시스템 전체에 어떤 영향을 주는지에 관련한 내용과 I/O의 성능지연 등을 포함하고 있으며 실시간성에 대한 연구도 진행되고 있다[3]. I/O스케줄링의 성능평가에 관련한 것은 장치공유에서 중요한 부분을 차지한다.

2.2 연구배경

본 연구는 가상화 기술을 이용하여 고장감내 PSTR을 구현하는 임베디드 시스템에 관한 것이다. PSTR 모델은 고장감내를 위한 것으로 본 임무를 수행하는 프라이머리와 프라이머리 오류상황에서 절체되어 사용되는 쉐도우로 구성된다[4].

PSTR 모델은 Fig. 1[4]과 같이 고장감내를 지원하기 위해 두 개의 컴퓨팅 개체인 프라이머리와 쉐도우 시스템을 가진다. 프라이머리에서 주 임무를 수행하다가 오류가 발생하는 경우 지속된 주 임무 수행을 보장해주기 위해 쉐도우 시스템으로 주 임무 수행권한이 절체된다[5].

Fig. 1의 시계모양은 실시간요구사항을 나타내며 별표는 전역변수 영역의 공유를 위한 것으로 공유절차가 제공되어야 한다. P2는 원래 수행하려는 주 임무이고 수락테스트는 실시간으로 결과를 저장한다. 연산의 성공여부를 통해 프라이머리와 쉐도우가 동기화한다. Fig. 1의 실선 화살표는 외부로부터의 입출력과 프라이머리 쉐도우 간 통신을 의미한다. 본 연구의 응용 시나리오는 PSTR 모델에 근거하여 외부입출력은 직렬포트를 사용하고 프라이머리 쉐도우 간 태스크 통신은 전용 네트워크 장치를 사용한다.

프라이머리가 만들어낸 결과의 유효성 테스트 및 검증과정에서 쉐도우와 동기가 이루어진다. 주 임무가 프라이머리에서 이루어지므로 프라이머리의 장치입출력은 모든 동작이 반영되며 쉐도우의 경우는 프라이머리 작업의 유효성 판단과 검증하는 작업을 하므로 판독연산만을 수행하게 된다. 프라이머리에서 오류가 발생되어 유효성 테스트의 결과가 오류로 판명될 경우 쉐도우가 주 작업이 되므로 직렬포트에 대한 출력이 쉐도우로 절체된다.

3. 시스템 구현

PSTR 모델은 일반 하드웨어 시스템 위에 구현되어왔다. 본 연구에서는 PSTR 모델을 사용한 고장감내 기능을 가상

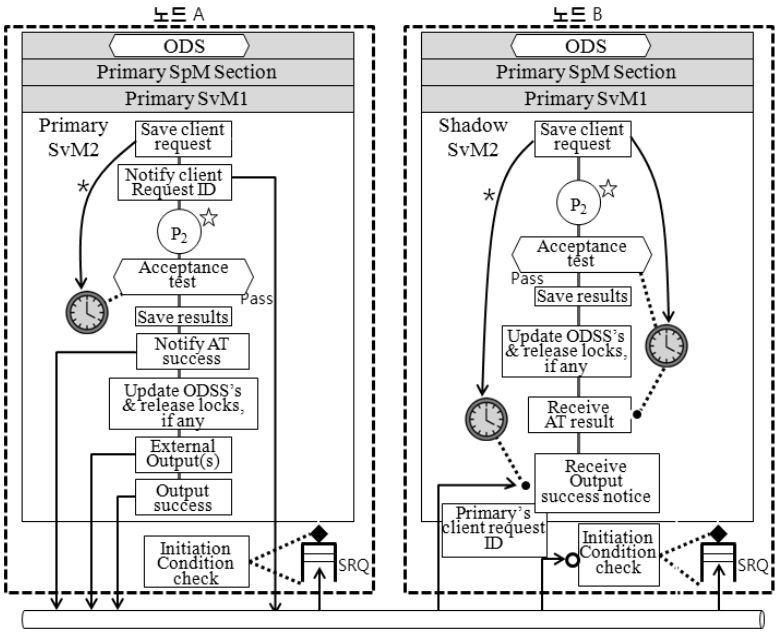


Fig. 1. PSTR model

머신 기반 위에 구현한다[5]. 구현 가상 머신 시스템으로 버추얼박스 시스템 오픈소스 버전을 사용하고 PSTR 모델의 구현을 위해서 가상 머신 시스템을 수정하고 기능을 추가하도록 한다.

PSTR 모델 구현을 위해 가상 머신에서 수정되거나 추가되는 기능은 다음과 같다. 첫째로 호스트의 직렬통신 데이터를 프라이머리와 쉐도우 가상머신에서 공유하기 위한 부분, 그리고 둘째로 프라이머리상의 응용프로그램과 쉐도우의 응용프로그램 간의 통신기능이다. 직렬통신 데이터를 공유하기 위해서 호스트의 직렬포트로 들어오는 데이터를 프라이머리와 쉐도우에 적절히 분배하는 기능모듈이 필요하다. 직렬포트의 입출력 속도를 하드웨어 입출력의 연장선상에서 조절해주어야 한다. 직렬포트의 경우 115kbps의 속도를 지원해주면 되기 때문에 소프트웨어 처리에는 무리가 없으나 안정적인 속도를 지원해주기 위한 기능은 추가되어야 한다. 프라이머리 쉐도우 간의 통신은 기존에는 IP통신을 사용하고 있다. 이는 데이터를 전달하는 데 있어서 항상 가상 머신 내 운영체제의 IP통신 프로토콜을 통과하여야 한다. 전송 속도의 경우도 네트워크 전송정책을 따라야 하는 부분이 있다. 프라이머리와 쉐도우 간의 통신의 특성은 작은 데이터를 빠르게 전송하는 특성을 가지고 있다.

본 연구에서는 실제 구현을 통해서 프라이머리 쉐도우 간의 통신요구를 수용하기 위한 소프트웨어 네트워크 장치를 제안하고 구현하였으며 실험 결과를 제시하였다[6]. 직렬통신 분배정책과 프라이머리 쉐도우 간의 통신을 관리하기 위한 관리모듈을 호스트상에 구성하여 간헐적인 모니터링을 통해 시스템의 상태를 감시한다.

성능평가 대상 하드웨어 시스템의 스펙은 프로세서 AMD 64bits, 2.0 GHz로 구성되고, 메모리 기억장치는 8G이다. 운영체제는 리눅스 커널 3.4.0의 실시간 커널 패치를 적용하고 프로세스의 실시간 지원 기능을 사용한다.

3.1 관리모듈의 구성

관리모듈은 하나의 가상 머신으로 만드는 방법과 호스트 상의 모듈로 구성하는 방법이 있다. 가상 머신의 형태로 탑재하여 관리하게 되면 다른 시스템에 이식성이 좋고 호스트 상의 모듈로 구성하게 되면 통신성능이 우수하여 시스템에 로드를 주지 않는다[6]. 본 연구에서는 기존 연구 결과를 참조하여 통신성능의 효율성을 중심으로 판단하고 호스트 모듈로 구현한다[7].

3.2 가상 머신의 호스트 직렬장치 동시 공유

호스트 시스템 내에는 프라이머리와 쉐도우 가상 머신이 있고 두 가상 머신은 호스트의 직렬포트를 동시에 공유한다. 이러한 공유는 소프트웨어로 작성되는 가상 머신 장치에 입출력을 위한 소프트웨어 처리로 가능해진다[8]. 호스트의 직렬포트로 들어오는 데이터를 호스트 드라이버 단에서 받아서 복제된 데이터의 형태로 두 가상 머신에 전달한다. 프라이머리 직렬포트에 대한 쓰기연산은 주 임무를 수행 중인 가상 머신이 할 수 있다. 프라이머리가 주 임무를 수행하고 있는 동안은 프라이머리가 쓰기연산의 권한을 가지며 프라이머리 오류가 발생하여 절체된 쉐도우가 주 임무를 수행하고 있다면 쓰기연산의 권한은 쉐도우 가상 머신이 가지게 된다. 호스트의 직렬포트에 대한 판독연산과 쓰기연산의 속도를 조절하는 기능을 지원하여 가상 머신에서 입출력되는 데이터가 일정한 간격으로 안정되게 수행할 수 있도록 한다. 호스트 직렬포트 동시 공유를 위한 소프트웨어 모듈을 작성하고 이를 가상화 모듈에 포함시킨다. 커널의 실시간 지원기능을 사용하여 구현한다.

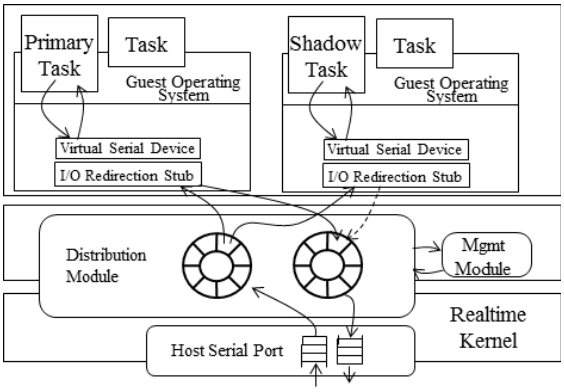


Fig. 2. Serial Port Sharing

직렬포트 동시 공유는 Fig. 2에서 보는 바와 같이 호스트 상에 직렬포트를 제어하는 드라이버가 있고 그 위에 분배모듈이 위치한다. 분배모듈은 프라이머리 가상 머신과 쉐도우 가상 머신 각각의 가상 직렬장치와 통신하여 호스트의 직렬데이터 전송을 위한 통신을 수행하며 타임스탬핑을 한다. 직렬포트 동시 공유를 위한 기능은 관리모듈의 관리하에 모니터링된다.

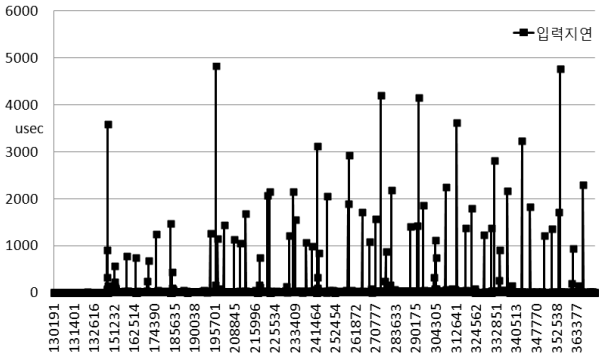


Fig. 3. Input Delay of Sharing Device

호스트의 직렬장치 공유 기능을 구현한 후 실제 하드웨어 장치에서 들어오는 데이터를 처리할 수 있는 성능요구를 만족시키는지를 실험하였다. 가상 머신상에 구현된 직렬장치는 하드웨어의 성능명세에 준하여 115kbps를 만족해야 한다. 또한 공유로 인해 발생하는 성능을 지연하는 요소를 파악하고 호스트로 들어오는 데이터 처리모듈과 가상 머신 간

의 데이터 통신이 균형 있게 이루어지도록 한다. Fig. 3[6]의 가로축은 시간의 흐름으로 실시간 클럭의 시간흐름에 따라 증가하는 시간값이다. 즉 시간의 흐름을 나타내고 세로축은 시간흐름에 대한 입력지연시간을 마이크로초로 표현한 것이다. 전송단위는 바이트 단위로 전송된다. 그래프에서 관찰되는 지연시간은 2us초~5ms이다. 시리얼의 입출력속도가 초당 115k를 최대로 한다고 할 때 데이터 전송의 입력지연은 0.7us 정도를 요구한다. 이를 참조하여 직렬입력을 일정하게 유지하기 위해 성능제어를 추가하였다.

3.3 프라이머리 쉐도우 태스크 간 통신지원

프라이머리와 쉐도우 태스크 간의 전용통신 기능을 지원하기 위해 Fig. 4와 같이 가상 네트워크 포트를 구현한다. 이 네트워크 포트는 가상 머신 간의 통신만을 지원하기 위한 것으로 하드웨어로 구현된 시스템에서 볼 수 없는 통신 채널을 가질 수 있다[9]. 이 네트워크의 주소체계는 버추얼박스에서 할당받은 가상 머신 UUID(Universal Unique IDentifier), 태스크ID 등을 사용하는 주소체계를 가지게 되며 타임스탬프를 두어 시간정보를 유지한다.

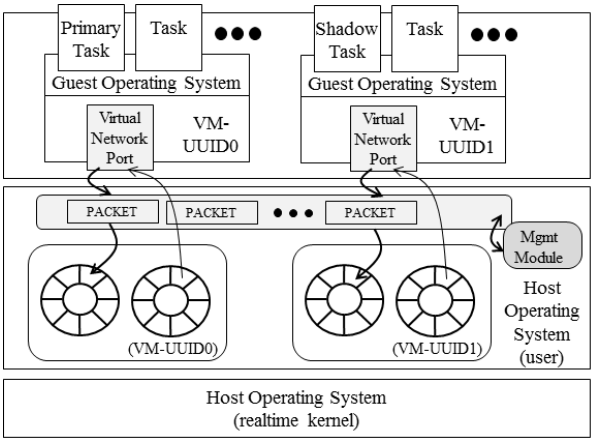


Fig. 4. Communication between tasks

패킷을 전송하는 과정에서 패킷의 지연시간을 측정하여 지연DB에 반영한다. 지연DB는 평균지연시간, 최근지연시간, 최소지연시간 등을 계산하여 패킷이 목적지까지 도착하는 예측시간정보를 제공한다. 각 단계는 생성단계(G_i), 큐잉단계(Q_i), 처리단계(P_i), 전송단계(T_i)로 나누어 각각의 지연시간을 관리한다. 시간요구 매개변수가 hard, soft의 여부에 따라 처리를 수행하며 hard인 경우는 예상전송시간보다 전송요구 시간이 작으면 수행하고, 전송요구시간이 크면 예상전송시간을 미리 계산된 평균최소지연시간(MIN_{delay})과 비교하여 만족하면 수행하고 그렇지 않으면 실패를 반환한다[6].

$$MIN_{delay} = \frac{1}{N}\Sigma_{i=0}^{N}(MIN(G_i) + Q_i + P_i + MIN(T_i))$$

평균지연시간은 아래와 같이 계산한다.

$$AVG_{delay} = \frac{1}{N}\Sigma_{i=0}^{N}(G_i + Q_i + P_i + T_i)$$

패킷생성단계에서 아래와 같이 비교한다.

if $((RTT(packet) < AVG(delayDB))$
  Pakcet is allocated from preallocated queue pool
  $(=MIN(G_i))$
else
  Packet is allocated from memory

패킷전송단계에서는 아래와 같은 비교를 수행한다.

if $((RTT(packet) < AVG(delayDB))$
  Pakcet is placed in front of queue,
  Notify packet arrived using SIGIO
  $(=MIN(T_i))$
else
  Packet is placed in queue

RTT: Requested Transmission Time,
AVG(delayDB): average time of delay in delay DB

매개변수가 soft인 경우는 평균최소지연시간(MIN_{delay})과 비교를 수행한 후 만족여부에 관계없이 수행한다.

프라이머리 가상 머신과 쉐도우 가상 머신 간의 통신을 위한 전용 네트워크 포트를 구성하고 실험하였다. Fig. 5은 소켓통신과 전용 네트워크 포트를 32바이트, 64바이트, 1024바이트 크기의 데이터를 지속적으로 전송하여 그 전송시간을 측정한 결과이다.

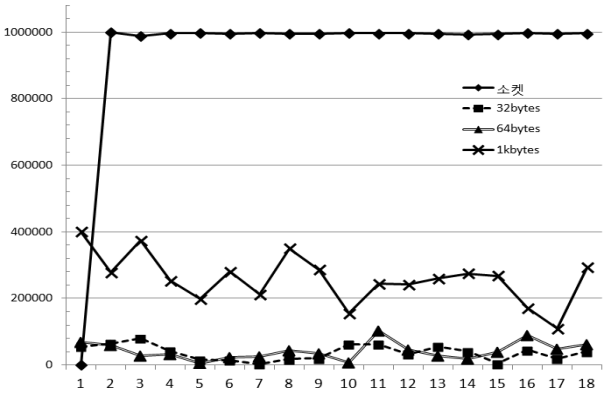


Fig. 5. Comparison of communication time

Fig. 5[6]에서 보는 바와 같이 소켓통신의 경우 일정량 초과의 전송이 일어나지 않는 한 비슷한 통신시간을 보여준다. 다른 가상 머신상의 태스크들 간의 일회 통신량은 1024바이트 이하의 지연 없는 즉각적인 통신을 필요로 하는 특성을 가지고 있다. 이러한 특성으로 볼 때 전용 네트워크 가상포트를 위한 통신이 소켓통신보다 적절한 장치인 것을 확인할 수 있었다.

4. 실험과 시연

Fig. 6. Primary Activity

Fig. 7. Primary Fail Activity

Fig. 8. Switch to Shadow

실험은 구현환경에서 시연 시나리오를 가지고 플라이트기어 환경에서 모의실험과 실제 실험환경을 한다. 메인보드 상에는 실시간 패치 리눅스가 탑재되고 본 연구에 의해 수정된 버추얼박스가 설치된다. 버추얼박스 위에 PSTR 모델 고장감내 구현을 위해 프라이머리 가상 머신과 쉐도우 가상 머신이 설치된다. 직렬포트는 모의실험인 경우 플라이트기어 소프트웨어에 연결되어 실험하고 실제 실험의 경우는 자세결정을 위한 제어보드에 연결되고 제어보드는 쿼드콥터에 연결된다.

시연 시나리오는 다음과 같다. 프라이머리는 쉐도우에 주기적으로 하트비트를 보내고 자신의 임무를 수행한다. 쉐도우는 프라이머리의 이상상태를 감시하면서 자신의 역할을 한다. 프라이머리에서 오류 발생 시 쉐도우는 I/O에 대한 권한을 프라이머리로부터 가져오고 프라이머리의 역할을 대체하고 프라이머리는 다시 기동되어 정상임무 수행가능 상태가 되면 다시 제어권을 쉐도우로부터 가져온다.

Fig. 6은 프라이머리가 정상적인 임무 수행을 할 때 제어되는 플라이트기어 영상이다. 시스템 상황이 오른쪽 모니터링 화면에 보이고 있다. Fig. 7은 프라이머리가 임무를 실패했을 때 비행체가 심하게 기울고 임무가 절체되는 상황이다. 임의로 프라이머리 시스템을 실패시켰을 경우 쉐도우로 절체되기 전의 화면 캡처이다. Fig. 8는 프라이머리의 오류상황이 인지되고 프라이머리에서 쉐도우로 제어가 절체되어 수행되는 모습을 보여준다. 모의실험에서는 하트비트를 보내거나 프라이머리의 오류상황 감지부터 쉐도우로 절체되는 과정이 눈으로 관찰할 수 있는 형태로 일어나지 않으므로 하트비트와 프라이머리 감시주기에 큰 파라미터를 부여하여 실험하였고 그 결과 현상이 느리게 진행되어 진행과정을 관찰하였다.

Fig. 9는 실험환경을 구성한 것으로 실험을 위해 사용된 메인 하드웨어 위에 Fig. 9에서 보는 것과 같은 소프트웨어 스택이 구성되어있다. 호스트 하드웨어 위에 호스트 운영체제가 탑재되고 그 위에 가상화 환경이 구성되고 프라이머리 가상 머신과 쉐도우 가상 머신은 eCos운영체제를 탑재한 게스트머신의 형태로 존재한다. 호스트 하드웨어의 직렬장치와 무선조정기기의 입출력이 제어보드로 연결되고 두 개의 입력내용을 기반으로 쿼드콥터를 조정한다. 실제 실험에서 프라이머리와 쉐도우 간 절체는 육안으로 관찰되지 않고 모니터링 기능을 통해서 통지될 수 있다.

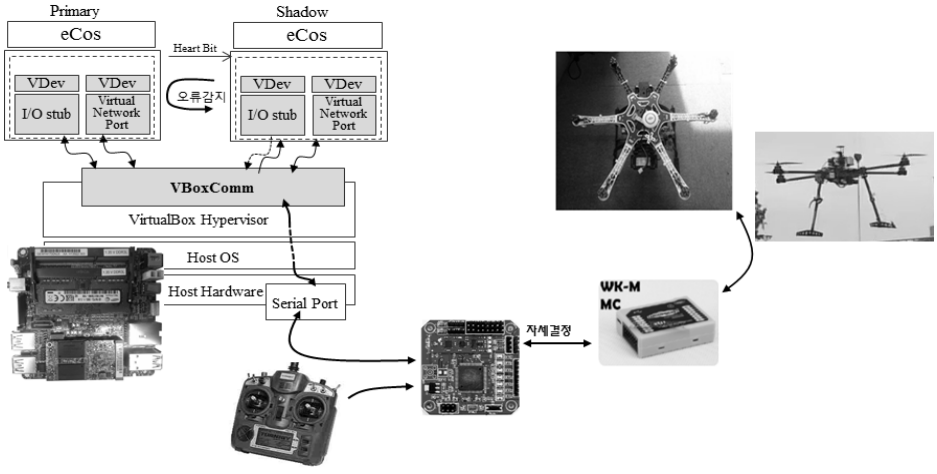


Fig. 9. Experiment Composition

5. 결 론

본 연구는 PSTR 모델을 가상 머신 환경에서 구현하는 고장감내 시스템에 관한 것으로 PSTR 모델을 가상화 환경에 최적화하기 위해 가상 머신을 수정하여 PSTR 모델에 최적화된 임베디드 가상화 환경에 적용하였다. 버추얼박스 가상환경에서 입출력 기능을 복제하고 입출력 흐름을 제어할 수 있는 기능을 추가하였고 가상 머신 간의 통신을 소량 데이터 전송에 최적화된 가상 네트워크 장치를 구현하였다. 입출력 흐름을 제어하는 가상 복제 장치의 성능을 살펴보았고 가상 네트워크 장치를 기존의 소켓통신과 비교하였다. 비교를 통해 최적화 효과를 관찰할 수 있었다. PSTR 모델에 최적화된 가상 머신 모듈을 하나의 모듈로 구성하여 가상 머신의 기존 운영수행과 동일하게 구성하였다. 가상화 기술을 이용한 PSTR 모델 모듈을 모의실험하기 위해 플라이트기어 소프트웨어를 사용하였고 하트비트와 감시주기에 큰 파라미터를 주어 오류와 절체상황을 관찰하였다. 실제 실험에서는 이미 검증된 PSTR 모델에 적응된 가상 머신 시스템을 플라이트기어로 입력되는 부분을 인터페이스를 조정하는 작업을 거쳐서 실험된다.

본 연구에서는 PSTR 모델 고장감내를 가상화 환경에서 구현하였고 가상화 환경에서 고려되어야 할 요소들을 구현하여 기능 실험하였고 임베디드 환경에서 가상화 기술을 이용한 효율적인 고장감내가 가능함을 보였다.

References

[1] Ning Li, Yuki Kinebuchi, Hitoshi Mitake, Hiromasa Shimada, Tsung-Han Lin, and Tatsuo Nakajima, "A Light-Weighted Virtualization Layer for Multicore Processor-Based Rich Functional Embedded Systems," *Proceedings of 15th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing,* pp.144-153, 2012.

[2] Diego Ongaro, Alan L. Cox, and Scott Rixner, "Scheduling I/O in Virtual Machine Monitors," *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pp.1-10, New York, USA, 2008.

[3] Zonghua Gu, Qingling Zhao, "A State-of-the-Art Survey on Real-Time Issues in Embedded Systems Virtualization," *Journal of Software Engineering and Applications,* pp.277-290, May, 2012.

[4] K. H. Kim, C. Subbaraman, "The PSTR/SNS scheme for real-time fault tolerance via active object replication and network surveillance," *Knowledge and Data Engineering, IEEE Transactions on,* Vol.12, No.2, pp.145, 159, Mar/April, 2000.

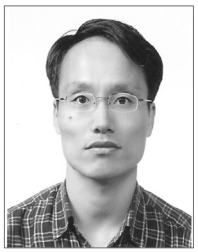
[5] Kim, K. H., Liu, and J.J.Q., "Techniques for implementing support middleware for the PSTR scheme for real-time object replication," *Object-Oriented Real-Time Distributed Computing, 2004. Proceedings. Seventh IEEE International Symposium on*, pp.163-172, 14-14 May, 2004.

[6] Jinho Yoo, Kyujong Han, Yong-Hyun Kim, Mirim Ahn, and Doo-Hyun Kim, "The Execution Environment Study of Primary Shadow TMO Replication Model on a Virtual Machine," *Korean Institute of Information Technology,* Vol. 11, No.8, pp.153-162, Aug., 2013.

[7] Fang Wei, Xian Xuefeng, "A Virtualization Resource Management Platform of Cloud Computing Based on the Xen," *Information Science and Engineering(ISISE), 2012 International Symposium on*, pp.220-222, 14-16 Dec., 2012.

[8] Maruyama, T., Yamada, T., "Sharing IO devices using hardware virtualization method for component-based industrial controllers," *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, pp.705-708, 15-18 Sep., 2008.

[9] Nakauchi, K., Shoji, Y., Ito, M., Zhong Lei, Kitatsuji, Y., and Yokota, H., "Bring your own network - Design and implementation of a virtualized WiFi network," *Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th*, pp.483-488, 10-13 Jan., 2014.



유 진 호

e-mail : yoojh@bu.ac.kr

1994년 광운대학교 컴퓨터학과(학사)

1996년 서강대학교 컴퓨터공학과(석사)

2006년 충북대학교 컴퓨터학과(박사)

1996년~1998년 엘지정보통신연구소 전임연구원

1999년~2008년 한국전자통신연구원 선임연구원

2006년 조지아텍 공동연구 파견연구원

2008년~현 재 백석대학교 정보통신학부 교수

관심분야 : 임베디드시스템, 가상화시스템, HCI

한 규 종

e-mail : karjensia@konkuk.ac.kr

2013년 백석대학교 소프트웨어학(학사)

2013년~현 재 건국대학교 인터넷미디어공학과 석사과정

관심분야 : 임베디드시스템, 가상화시스템