논문 2011-48SC-2-1

효과적인 수중로봇 S/W 프레임웍 구현을 위한 데이터 공유구조

(Data Sharing Architecture for an Effective Implementation of Underwater Robot S/W Framework)

정 순 용*, 최 현 택*

(Soon Yong Jeong and Hyun-Taek Choi)

요 약

수중로봇 S/W 프레임웍은 센서 데이터 처리, 추진기 제어, 인지 및 행위 제어 등의 다양한 하위모듈로 구성된다. 로봇의 성능은 알고리즘 자체의 우수성 뿐 아니라 그 구현에도 큰 영향을 받는데, 효과적인 구현에 가장 큰 영향을 미치는 부분 중의 하나는 모듈간의 신호 및 데이터 전달을 담당하여 센싱 및 제어 주기에 영향을 주는 데이터 공유 모듈의 효율성이다. 이상적인 데이터 공유 모듈은 시스템의 H/W 및 S/W 구성에 상관없이 데이터 싱크로부터 데이터 소스에 지연 없이 접근할 수 있게 해야 한다. 그러나 실제에 있어서는 시스템 구성 특성에 의한 데이터 소스 모듈의 접근 처리 용량 한계, 네트워크 지연 및 운영체제의 스케쥴링 등으로 인하여 다양한 접근 지연이 요인이 존재한다. 본 논문은 수중로봇과 같이 소수의 컴퓨터로 이뤄진 소규모 분산시스템에서 이러한 접근 지연을 효과적으로 처리하기 위한 데이터 공유 모듈 구조 및 프로그래밍 모델을 제안하고 있다.

Abstract

An underwater robot S/W framework consists of various sub-modules such as sensory data processing module, thruster control module, cognition module and behavior control module. Performance of a robot is determined by not only the efficiency of algorithms used but also effectiveness of their implementations. One most important factor of the effective implementation is the efficiency of data sharing module, as it transmits signals and data between the sub-modules and thus is directly related to the cycles of sensing and control The ideal data sharing module enables immediate access to any data source irrespective of system configurations. In reality, however, there are lots of obstacles including limitation of processing capacity of source modules, delay over network, and scheduling latency of operating systems. The paper proposes a new data sharing architecture and programming models to effectively handle such obstacles in implementation of underwater S/W framework on a small scale distributed computing system.

Keywords: underwater robot, framework, data sharing, S/W architecture, robot middleware

Ⅰ. 서 론

오늘날 수중로봇은 점점 더 다양하고 복잡한 미션 수행을 위해 지능적이고 자율적인 형태로 진화하고 있다. 지능형 자율 수중로봇은 주변 환경 인지와 추론 및 작업 수행을 위해 다수의 센서와 추진기 및 이를 처리하기 위한 다양한 S/W 모듈을 필요로 한다. 로봇 S/W 프레임웍은 이러한 시스템의 복잡성을 다루기 쉽게 하면서 S/W 모듈의 재활용성과 이식성을 높여 생산성을 높이고 시스템의 유지 및 개선을 쉽게 한다. 이미 지상의 로봇에는 다양한 종류의 로봇 S/W 프레임웍들이 개발 되었으나[1~3], 수중로봇의 경우 수중이라는 환경적

* 정회원, 한국해양연구원
(Korea Ocean Research & Development Institute (KORDI))
접수일자: 2010년10월22일, 수정완료일: 2011년3월8일

특성으로 인한 환경 인지, 통신, 전력공급 등의 기술적 제약과 상업적 수요 부족 등으로 아직 주목할 만한 결과는 없는 편이다. 최근 해양연구원에서는 '고정밀 임무수행을 위한 인공지능 기반의 수중로봇 기술 개발' 과제를 통해 수중로봇에 맞춘 S/W 프레임웍 개발을 진행중인데[4-6], 이 논문은 효과적인 수중로봇 S/W 프레임웍 개발의 기초가 되는 데이터 공유모듈 구조에 대해서 다룬다.

지능형 수중로봇 S/W 프레임웍은 다양한 센서데이터 처리 모듈과 추진기 제어 모듈, 인지 모듈, 추론 모듈 및 행위 제어 모듈[7~9] 등 다양한 하위 모듈로 구성되어 있다. 로봇의 성능은 이들 각각의 모듈에 적용된 알고리즘의 우수성뿐만 아니라 알고리즘이 얼마나 효과적으로 구현되는 지에도 크게 영향을 받는다. 일반적으로 센싱 및 제어 주기가 짧을수록 로봇의 상황에 대한 대응이 빨라지고 지연에 의한 비선형 효과도 줄어듦으로 전체적인 로봇 성능이 향상된다. 따라서 효과적인 S/W 프레임웍 구현에 가장 큰 영향을 미치는 요소 중의 하나는 모듈간의 신호와 데이터 전달을 담당하는 데이터 공유 모듈의 효율성이라 할 수 있다.

이상적인 데이터 공유 모듈은 데이터 싱크에서 지연없이 데이터 소스에 접근 가능하게 한다. 그러나 실제에 있어서는 반드시 데이터 접근 지연(Data Access Latency)이 발생한다. 데이터 접근 지연의 원인으로는 데이터 소스 모듈의 접근 처리 용량 제한에 의해 생긴 병목현상에 의한 지연과 데이터 소스 모듈과 데이터 싱크 모듈사이의 네트워크 지연, OS(Operating System)의 스케쥴링에 의한 지연이 존재한다. 데이터 공유 모듈은 이러한 데이터 접근지연 문제뿐 아니라 로봇 S/W 프레임웍의 가장 기초적인 모듈로서 다른 상위 모듈의 개발 및 통합을 위한 다양한 S/W 요구사항을 만족해야 한다. 예를 들어 모듈간의 의존성 처리, 부분적이 오류에 의한 전체 시스템 패닉 방지, 시스템 설정 변경 지원 등이 있다. 본 논문은 이러한 요구 사항을 만족하면서 효과적으로 S/W 프레임웍을 구현하기 위한 데이터 공유 구조와 프로그래밍 기법을 제안하고 있다.

본 논문은 다음과 같이 구성되어 있다. Ⅱ장에서는 대상 시스템 및 데이터 공유 모듈에 대한 문제 분석이 이뤄지며, Ⅲ장에서는 제안된 데이터 공유 모듈 구조와 프로그래밍 모델을 설명하고, Ⅳ장에서 논문을 마무리 짓는다.

Ⅱ. 문제 분석

1. 데이터 공유 모듈 요구사항 분석

가. 대상 시스템

본 논문에서 제안하는 데이터 공유 구조는 한국해양연구원에서 개발 중인 수중로봇을 대상으로 설계되었다. 한국해양연구원에서는 2010년부터 5년에 걸쳐 고정밀 임무수행을 위한 지능형 수중로봇 개발 프로젝트를 진행 중에 있으며, 이 과제를 통해 개발되는 수중로봇은 비전 데이터, 수중 음향 데이터 및 제어 알고리즘 처리를 위하여 3개의 컴퓨터가 이더넷으로 연결된 소규모 분산시스템으로 구성되어 있다. 그림 1은 대상 수중로봇 시스템 구조를 보여 주고 있다.

지능형 수중로봇이 비교적 복잡한 분산시스템을 필요로 하는 까닭은 프로젝트에서 추구하는 높은 수준의 로봇 자율성과 더불어 수중이라는 작업환경 특성에 대처하기 인함이다. 수중에서는 전자기파 감쇄가 심하여 전자기파 기반의 무선 통신이 거의 불가능하므로 센서 네트워크나 컴퓨팅 서버의 도움을 얻을 수 없다. 따라서 로봇 자체적으로 상황 인지를 위한 센서 정보를 수집하고 분석하기 위해서 다양한 센서를 운용해야하고, 이를 처리하기 위한 고사양의 컴퓨팅 자원이 필요하다.

해양연구원 로봇을 구성하는 3개의 컴퓨터는 각각 주제어, 비전 및 음향 컴퓨터로 명명 되었다. 각 컴퓨터의 주요 역할은 다음과 같다. 주제어 컴퓨터는 추진기와 관성 센서 및 상태 모니터링 센서가 연결되어 추진기 제어 및 상위 제어 알고리즘을 구동하며, 비전컴퓨터는 카메라, 조명 등의 비전 관련 기기들이 연결되어 있어서 영상 처리 및 영상 기반 인지 모듈을 구동한다.

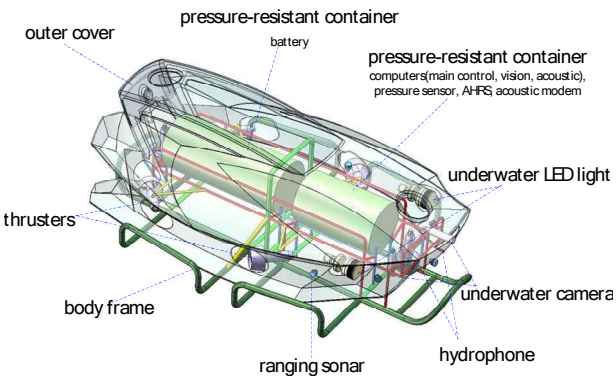

그림 1. 대상 수중로봇 시스템
Fig. 1. Target underwater robot system.

마지막으로 음향 컴퓨터에는 하이드로폰이 연결되어 있어서 음향 데이터 처리 및 음향 기반 인지 모듈을 구동한다.

나. 기능 요구사항

데이터 공유 모듈은 S/W 프레임웍을 구성하는 모든 모듈간의 신호 및 데이터 전달을 가능하게 해야 한다. 기본적으로 모듈은 분산시스템 내의 어느 위치든 배치될 수 있으며, 신호 및 데이터는 단방향, 양방향 전달뿐 아니라 브로드캐스팅이 가능해야 한다. 데이터 전달 최대 지연 시간은 가능한 짧아야 하며 모듈간의 데이터 스루풋은 가능한 커야 한다. 구체적이고 정량적인 성능 요구사항은 수중로봇의 미션 요구사항에 맞춰 H/W 스펙과 함께 결정되어야 한다.

다. S/W 프레임웍 구현 요구사항

데이터 공유 모듈은 모듈간의 통신 채널이라는 기본적인 요구사항 외에도 S/W 프레임웍의 가장 기초적인 모듈로서 프레임웍 내의 모듈 개발 및 통합을 위한 다음의 세 가지 요구사항을 만족해야 한다.

첫째, 프레임웍 모듈 구성이 변경 가능해야 한다. 수중 로봇 개발 과정에서 다양한 센서/추진 장치 및 S/W 모듈이 추가/제거 될 수 있기 때문이다.

둘째, 모듈간의 의존성 분산이 가능해야 한다. 즉, 어떤 모듈이 제공해 주는 서비스가 다른 다양한 모듈로부터도 얻을 길이 있어서 해당 모듈의 부재나 오류로부터 시스템이 보호받을 수 있어야 한다.

셋째, 안정성을 확보할 수 있도록 모듈 패닉이 시스템 패닉으로 번지는 것을 방지해야 한다. 모듈 패닉이 시스템 패닉으로 번지는 것을 통제하지 못할 경우 최종 시스템의 안정성 보장 뿐 아니라 프레임웍 개발 기간에도 영향을 준다.

2. 데이터 접근 지연 원인 분석

이번 절에서는 이러한 데이터 접근 지연 요소에 대해 자세히 살펴보고, 대상 시스템에서 이를 효과적으로 해결할 수 있는 방안에 대해 생각해 보자.

가. 데이터 소스 모듈의 병목

데이터 싱크 모듈은 항상 가장 최신의 데이터를 얻기를 원한다. 그런데 데이터 싱크 모듈들이 동시에 데이터 소스 모듈에게 신규 데이터를 요청할 경우, 소스 모듈의 접근 처리용량 초과로 인해 병목 현상이 발생할 수 있다. 이러한 병목 현상은 데이터 접근 지연을 늘리고, 데이터 동기화를 깨뜨려 알고리즘 오작동을 유발할 수도 있다.

데이터 소스 병목 현상은 캐시(혹은 프락시 패턴)를 사용함으로써 해결할 수 있다[10]. 즉, 소스 모듈에 데이터를 요청하기 전에 캐시에 유효한 데이터가 있는 지 확인하고, 없는 경우에만 캐시 업데이트를 통해 소스 모듈에 접근하는 방식이다. 캐시는 캐시를 어디에 둘 것인지, 캐시 업데이트 방식을 어떻게 할 것인지에 따라 다양하게 구현할 수 있다.

나. 네트워크에 의한 지연

네트워크에 의한 지연은 노드간의 물리 채널을 통과하면서 생기는 지연과, 네트워크 계층을 통과하면서 생기는 지연, 컨텍스트 스위칭에 의한 지연 등이 있다. 컨텍스트 스위칭에 의한 지연은 멀티프로세스 모델을 지원하는 OS에서 프로세스 간 통신 채널을 통해 데이터가 전달 될 때, 프로세스 컨텍스트 변경에 의해 생기는 지연으로 IPC(Inter-Process Communication) 매커니즘의 종류 및 스케쥴러의 종류에 따라 달라질 수 있다. 일반적으로 유저 컨텍스트에서 커널 컨텍스트로 데이터 복사가 일어나는 소켓기반 IPC 매커니즘의 오버헤드가 공유메모리 기반 IPC보다 크다. 네트워크 계층을 통과하면서 생기는 지연은 데이터 패킷의 분해와 합성을 하는 과정에서 생기는 오버헤드에 의한 지연으로, TCP와 같이 신뢰할 수 있는 네트워크를 이용할 경우 네트워크 상황이 나쁠 때 커질 수 있다. 마지막으로 물리채널에 의해 생기는 지연은 데이터 패킷이 허브나 라우터를 거쳐 목적지로 전달될 때 생기는 지연으로, 네트워크 구조를 단순화 하여 네트워크 거리를 줄이고, 병목 현상이 발생하지 않도록 네트워크 대역폭 이내에서 데이터를 전송함으로써 지연을 줄일 수 있다. 수중로봇과 같이 허브를 중심으로 독립된 성형 형상(Star Topology)을 구성하는 소규모 네트워크에서는 외부 네트워크로부터 간섭이 없으므로 물리채널에 의한 지연은 거의 최적화된 상태로 볼 수 있다. 일단 통신이 수행되면 네트워크에 의한 지연은 불가피할 수밖에 없으므로, S/W 모듈 배치 단계에서 데이터 결합도를 높여 프로세스 혹은 컴퓨터 간 통신 줄이는 것이 가장 좋다.

다. 스케쥴링 지연

스케쥴링이란 멀티태스킹을 지원하는 OS가 태스크의 실행 순서를 정하는 작업을 말하는데, 스케쥴링 지연은 태스크의 개수가 실제 프로세서의 개수보다 많을 때 태스크가 프로세서 실행권한을 얻을 때까지의 대기로 인해 생기는 지연을 말한다. 로봇과 같이 복잡한 시스템은 멀티태스킹을 지원하는 OS를 사용할 수밖에 없으므로, 우선순위에 의해 스케쥴링하여 중요한 태스크의 스케쥴링 지연을 줄일 수 있는 실시간 OS를 사용하고 멀티코어나 분산 시스템의 사용으로 평균적인 스케쥴링 지연을 낮추는 방법을 사용해야 한다.

표 1은 이상에서 분석한 데이터 접근 원인과 대응 방법을 정리하고 있다.

표 1. 데이터 접근 지연 원인 및 대응 방법
Table 1. Sources of data access latency and means to countermeasure.

지연 원인	완화 방안
소스 모듈 병목	cache 구현을 통한 부하 분산
네트워크 지연	데이터 결합도 향상을 통한 트래픽 최적화, 네트워크 topology 최적화
스케쥴링 지연	실시간 OS 채용, 분산 컴퓨팅 활용

Ⅲ. 제안된 데이터 공유 구조

1. 데이터 공유 구조 개요

이번 장에서는 앞 장에서 살펴본 데이터 공유 모듈 요구 사항을 만족하면서 데이터 접근 지연을 최소화할 수 있는 데이터 공유 구조를 제안한다. 그림 2는 제안된 데이터 공유 구조의 개요도이다.

제안된 공유구조는 네트워크 특성에 따라 데이터 공유범주를 프로세스 내부, 컴퓨터 내부, 전체 분산 시스템으로 나누어 접근하고 있다. 범주가 넓어질수록 지연시간의 평균값 및 변동성이 커지게 되는데, 제안된 데이터 공유 모듈은 통신 범주별 네트워크 지연의 상대적 크기를 고려하여 캐시 패턴을 일반화 하는 구조를 통해 소스모듈 병목과 네트워크 지연을 동시에 처리할 수 있게 하였다. 그림 2에서 보는 바와 같이 각각의 통신 범주 경계에는 블랙보드 클라이언트, 블랙보드 서버 및 블랙보드 프락시가 있어서 캐시 효과를 구현하고 있다.

이러한 캐시 구현은 캐시 데이터가 유효한 기간 동안 소스 모듈에 데이터를 요청하는 대신 캐시 데이터 값을 읽으면 되므로 데이터 소스모듈에 부하를 크게 줄이게 된다. 또한 제안된 구조는 모듈 배치를 통하여 네트워

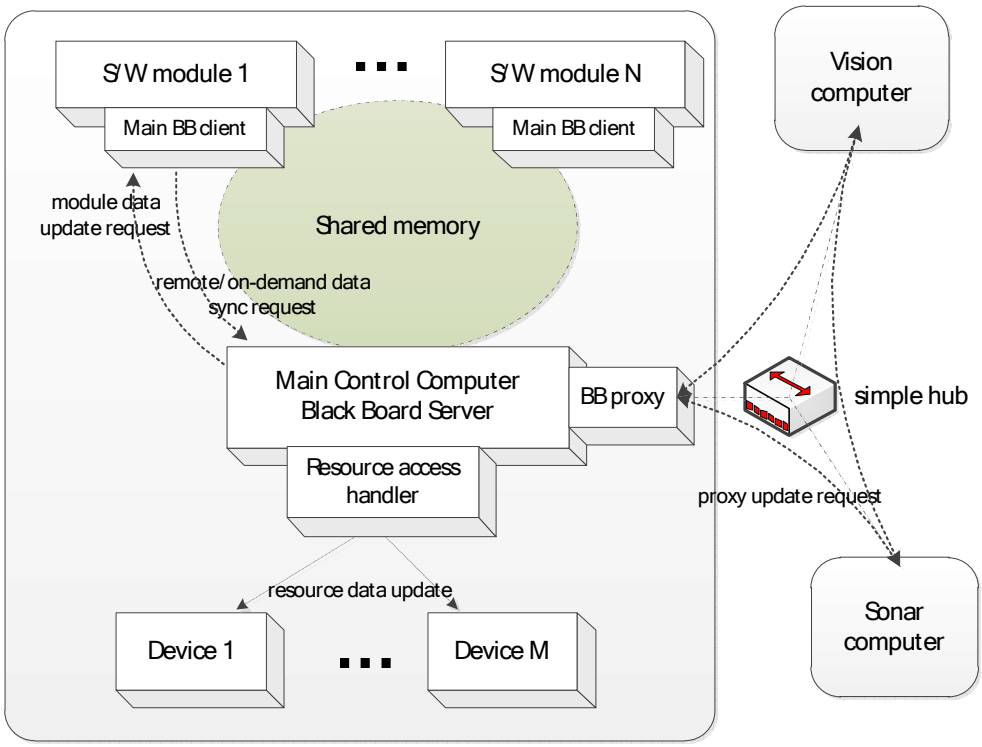

그림 2. 제안된 데이터 공유구조 개요
Fig. 2. Outline of the proposed data sharing architecture.

크 지연을 어느 정도 관리할 수 있는 길을 열어 주고 있다. 즉, 데이터 소스와 싱크의 결합 정도에 따라 결합도가 매우 높은 경우 동일 프로세스에, 결합도가 높은 경우는 동일 컴퓨터에 우선적으로 모듈을 배치하여 범주를 넘어선 통신을 줄여 네트워크에 의한 지연을 어느 정도까지는 관리할 수 있다.

그림에서는 설명되지 않고 있지만, 제안된 공유 구조에서는 공유변수 인터페이스를 통해 모든 모듈에 대한 접근이 가능하게 한다. 이를 통해 모듈 개발자가 특정 모듈에 대한 의존성을 피하고 여러 가지 모듈 데이터를 합성하여 보다 안정적인 시스템을 개발 할 수 있는 길을 열어 두었다. 시스템에서 사용되는 공유 변수의 종류와 특성은 스크립트로 정의할 수 있도록 하여 전체 프레임웍 구성을 변경할 수 있게 하였으며, 프로세스 모델을 기반으로 구현되어 단일 모듈 패닉이 전체 시스템 패닉으로 번지는 것을 막을 수 있도록 한다.

2. 공유 데이터 모델

가. 공유 데이터 종류

제안된 데이터 공유 구조의 실제 구현에서는 모든 모듈이 공유 데이터 인터페이스를 통하여 통신하도록 하였다. 예를 들어 심도센서 데이터 처리를 담당하는 모듈이 현재 심도 정보를 제공하는 경우, 심도 정보를 필요로 하는 모듈은 누구나 심도 데이터에 대한 접근 클래스 객체를 통해 접근할 수 게 되는 방식이다.

표 2. 공유 데이터 특성 분류
Table 2. Properties of shared data.

구분	수식어	설명
공 유 범주	local	공유 범주가 컴퓨터 내부
	global	분산 시스템 전체
소 유 방식	publish	publish/subscribe 모델
	share	공동 소유, 공유 변수
갱 신 방식	ondemand	subscriber 요청시에만 갱신
	anytime	owner가 언제든 갱신
구 현 수준	user	사용자 모듈 수준에서 정의
	system	공유 시스템 모듈 수준에서 정의

공유 데이터는 공유 범주, 소유방식, 갱신 방식, 구현 수준에 따라 다양한 특성을 가질 수 있다. 표 2는 공유 데이터의 특성을 종류에 따라 분류하고 있다. 다양한 데이터 특성은 더 예측가능하고 효율적인 방법으로 데이터를 공유할 수 있게 하며, 수식어는 공유 데이터 설정에 사용되어 데이터 공유 모듈에서 해당 공유데이터를 처리하는 데 활용된다. 참고로, 공유데이터는 하나 이상의 멤버 변수를 가지며, 하나의 모듈에서 제공하거나 사용할 수 있는 공유 데이터의 종류와 개수에 제한은 없다.

나. 공유 데이터 설정

제안된 데이터 공유 구조에서는 공유 데이터 설정 파일을 이용하여 프레임웍의 구성 변경 가능성을 구현하고 있다. 공유 데이터 설정파일에는 각 공유 데이터의

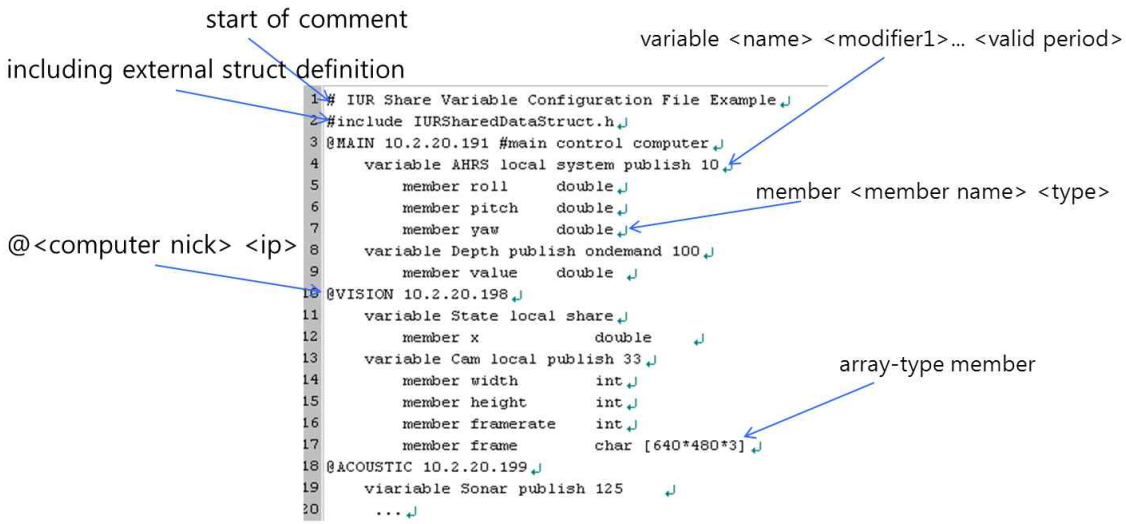

그림 3. 공유데이터 설정파일 예
Fig. 3. An example of shared variable configuration file.

배치(즉, 모듈 배치)와 공유데이터의 특성 및 멤버 변수를 정의할 수 있다. 그림 3은 설정 파일의 일례를 보여준다.

3. 데이터 공유 시스템 구현

가. 공유 변수

앞 절에서 설명한 바와 같이 모듈간의 데이터 통신은 공유변수 인터페이스를 통해 이뤄진다. 이 공유 변수의 실체는 블랙보드 시스템에서 관리하는 공유 메모리인데, 공유 변수는 이 공유메모리의 관리 단위가 된다. 공유 변수를 관리하기 위해서 각 공유 변수는 이름, 식별 ID, 소유 모듈 컴퓨터 위치, 접근 플래그, 유효 기간, 공유 메모리 옵셋, 크기, 소유자 및 구독자 리스트 등의 정보를 가진다. 실제 사용자 모듈에서는 각 공유 변수에 접근하기 위한 전용 클래스의 객체를 생성하여 이를 통해 접근 가능하게 되는데, 객체 생성 과정에서 공유변수에 대한 소유 정보 및 접근 권한을 설정할 수 있다. 각 공유변수의 관리 정보 및 클래스는 공유변수 설정파일로부터 파이썬[11] 스크립트를 통해 자동적으로 생성된다. 표 3은 요청-응답 모델로 선언된 심도 값에 대한 접근 클래스 및 사용 예를 보여 준다.

나. 클라이언트-서버 모델

제안된 구조에서는 데이터 공유 인프라로 공유 메모리 기반의 블랙보드 시스템을 사용하고 있다. 여기에 사용된 블랙보드 시스템은 인공지능 시스템 구현에서 문제해결에 사용되는 복잡한 블랙보드 시스템[12~13]은 아니며 공유 메모리와 같은 공통 리소스를 효율적으로 관리하는 게 주목적인 경량화된 블랙보드 시스템이다. 블랙보드 서버는 공유 메모리와 시스템 레벨의 자원을 관리하며 블랙보드 클라이언트는 각 프로세스에서 블랙보드 서버에 접근하기 위한 통신 인프라를 제공한다. 클라이언트-서버 모델의 이점 중의 하나는 모듈마다 독립된 클라이언트 객체를 가지게 함으로써 프로세스 모델로 구현하기 쉽게 한다는 점이다.

다. 데이터 공유 네트워크 확장

제안된 데이터 공유 구조는 프락시 패턴을 이용하여 분산 시스템 전체로 데이터 공유 범주를 확장하고 있다. 여기서 말하는 프락시는 인터넷 상의 프락시 서버를 말하는 것은 아니며, 그와 유사한 기능을 가지는, 분산 시스템 상의 데이터 접근을 일반화하기 위한 구현 중의 하나로 외부 컴퓨터 데이터에 대한 접근을 관리하는 모듈이다. 블랙보드 서버는 이러한 프락시를 통해서 컴퓨터 외부의 데이터에 대한 접근 요청이나 업데이트 요청을 컴퓨터 내부의 모듈과 동일하게 처리하게 된다.

표 3. 공유 변수 접근 클래스 및 사용 예
Table 3. An example of class declaration to access a shared variable and its usage.

class example
(automatically generated)
	/// class Depth - scope:global, location:local,
	/// ownership:publish, level:user, update:on-demand
	class IUExport Depth : public IURShvInstance
	{
	  void MemberSync(bool writesync);
	public:
	  Depth(ShvUpdateHandler cb = NULL, bool asOwner = false);
	  IURShvInstance::MemberVar<double> value;
	};

owner side usage example	static int OnDepthReading(void *_depth)
	{
	  ... // code for Depth Sensor Reading
	}
	// declaration of Depth class instance as owner
	Depth * depth = new Depth(OnDepthReading, true);

user side usage example	Depth * depth = new Depth(); // declaration of Depth class as
	printf("depth = %f\n", depth->value()); // usage example

4. 프레임웍 프로그래밍 모델

가. 모듈 배치

제안된 구조에서는 모듈 위치와 상관없이 공유데이터에 대한 접근이 가능하다. 그렇지만 적절한 모듈 배치를 통해서 네트워크 지연을 줄이거나 시스템의 안정성을 높일 수 있다. 프레임웍 내부 모듈 및 이를 이용하는 어플리케이션은 모듈간의 결합도, 허용 가능한 지연시간 및 여유 컴퓨팅 자원 등을 고려하여 배치가 이뤄져야 한다. 모듈간의 결합도가 높아 데이터 공유가 빈번하게 일어난 다면 그러한 모듈들은 동일 컴퓨터에 배치하는 게 유리 하다. 빈번한 데이터 통신은 필요 없으나 계산량이 많이 필요한 모듈은 컴퓨팅 자원이 여유로운 컴퓨터에 배치하는 것이 좋다. 그 외의 모듈들은 컴퓨터 간 로드 밸런싱과 시스템 안정성 등을 최대화 하는 방향으로 배치해야 한다. 가령, 분산 시스템 상의 한 컴퓨터가 패닉에 빠져 해당 컴퓨터의 모듈이 제 기능을 못하더라도 이를 대체할 수 있는 기능을 다른 컴퓨터의 모듈에 분산함으로써 안정성을 높일 수 있다.

나. 동기화

제안된 구조에서는 모듈간의 동기화를 위해 요청-응답 모델과 발행-구독 모델을 도입하고 있다. 요청-응답 모델은 싱크 모듈 요청 시에 소스모듈이 데이터를 업데이트하는 방식으로 싱크 모듈이 데이터 통신의 주체가 된다. 발행-구독 모델은 소스모듈에서 데이터를 업데이트 하고 업데이트 된 사실을 구독자인 싱크모듈들에게 알리면 싱크모듈은 필요한 경우 업데이트 된 데이터를 읽어가는 방식으로 소스 모듈이 주체가 된다. 이러한 동기화 타입은 공유변수 스크립트를 통해 설정할 수 있다. 이러한 느슨한 동기화 구조는 우리 뇌의 신경세포간의 정보 전달 구조로부터 모티브를 얻은 것으로, 향후 인공지능 프레임웍 구현 시에 다양한 인지 모듈들이 병렬적으로 배치되고 동작하기 위한 기반을 제공한다.

Ⅳ. 결 론

우리는 본 논문을 통하여 수중로봇의 작업 환경 특성 및 S/W 프레임웍 요구사항을 분석 하고 이를 구현하기 위한 데이터 공유 구조를 제안하였다. 수중로봇과 같이 이더넷으로 연결된 소규모 분산시스템의 데이터 공유 모듈에서 발생할 수 있는 데이터 접근 지연의 원인은 데이터 소스 모듈 처리 용량에 의한 병목 현상, 네트워크에 의한 지연 및 OS 스케쥴러에 의한 지연 등이 존재하는데, 이들은 각각 캐시를 통한 부하 분산, 네트워크 형상 최적화와 적절한 모듈 배치 및 실시간 OS의 도입으로 문제를 완화할 수 있다. 본 논문에서는 공유메모리에 기반 한 블랙보드 시스템과 프락시를 이용하여 이를 효과적으로 구현하기 위한 데이터 구조 및 프로그래밍 모델을 제시하였다. 제안된 데이터 공유 구조는 2010년부터 한국해양 연구원에서 개발 중인 수중 로봇 개발에 활용될 예정이다.

참 고 문 헌

[1] Nader Mohamed et al., “A review of middleware for networked robots”, IJCSNS, Vol.9 No.5, pp.139-148, May 2009.

[2] 이승익 외, “로봇 소프트웨어 아키텍처의 연구동향과 현황”, 전자통신동향분석 제20권 제2호, 1-13쪽, 2005년 4월.

[3] http://opros.or.kr/

[4] Hyun-Taek Choi et al., “New project and new underwater robot for new missions”, ICAM, pp. 93-97, Oct. 2010

[5] 정순용 외, “수중로봇 지능구현을 위한 신뢰성 있는 S/W 플랫폼 구조 설계”, 대한전자공학회 하계학술대회, 33권 1호, 2103-2105쪽, 2010년 6월.

[6] 정순용 외, “수중로봇 플랫폼을 위한 효율적인 데이터 공유 구조 설계”, 대한전자공학회 정보 및 제어학술대회, 127-128쪽, 2010년 10월.

[7] Kimon P. Valavanis et al., “Control architectures for autonomous underwater vehicles”, IEEE Control Systems, pp.48-64, Dec 1997.

[8] Ronald C. Arkin, “Behavior-based robotics”, The MIT Press, 1997.

[9] Stefan B. Williams et al., “A decoupled, distributed AUV control architecture”, International Symposium on Robotics, Vol.31, pp. 246-251, 2000.

[10] Ulrich Drepper, “What every programmer should know about memory”, http://www.akkadia.org/drepper/cpumemory.pdf, 2007.

[11] http://www.python.org/

[12] http://en.wikipedia.org/wiki/Blackboard_system

[13] http://en.wikipedia.org/wiki/AI

저 자 소 개

정 순 용(정회원)
1998년 포항공과대학교 전자전기공학과 공학사
2000년 포항공과대학교 전자전기공학과 공학석사
2005년 포항공과대학교 전자전기공학과 공학박사
2005년～2010년 삼성전자 DMC연구소 책임연구원
2010년～현재 한국해양연구원 전문연구원
<주관심분야: 제어 및 로보틱스, S/W 구조>

최 현 택(정회원)
1991년 한양대학교 전자공학과 공학사
1993년 한양대학교 전자공학과 공학석사
2000년 한양대학교 전자공학과 공학박사
1993년～1995년 한국통신 연구개발원 전임연구원
2000년～2003년 미국 하와이주립대학교 Post Doc.
2003년～현재 한국해양연구원 책임연구원
<주관심분야: 제어 및 로보틱스, 수중로봇, 지능로봇>