무선 센서 네트워크에서 신뢰성 향상을 위한 효율적인 체크포인트 프로토콜

정 동 원[†] · 최 창 열[††] · 김 성 수[†††]

요 약

유비쿼터스 환경에서 정확한 데이터를 얻기 위해, 무선 센서 네트워크의 신뢰성은 필수적이다. 이것을 위해, 롤백 기술을 통한 자가 치유는 신뢰성 향상에 도움을 준다. 하지만, 로컬 시스템만을 고려한 기존 롤백 기술은 무선 센서 네트워크에서 자칫 전체 시스템 차원의 결함을 발생시킨다. 따라서, 롤백 기술을 무선 네트워크 차원으로 지원하기 위해 체크포인트 프로토콜이 제시되었다. 하지만, 무선 센서 네트워크가 가지고 있는 특유의 제한 조건 때문에 각각의 프로토콜들은 메모리, 성능, 그리고 전력 소모 효율에 있어서 상충관계가 존재한다. 따라서, 본 논문에서는 주소 기록 기반 프로토콜(address log based protocol, ALBP)이라 불리는 새로운 프로토콜을 제시한다. 이 기법은 비동기 방식을 지원하는 플랫폼 기반의 프로토콜로, 무선 센서 네트워크에서 중요한 고려사항인 전력 소모량, 메모리 사용량, 그리고 마감시간을 맞추기 위한 빠른 응답 시간을 만족시킬 수 있다.

키워드 : 무선 센서네트워크, 간접메모리접근, 체크포인트 프로토콜

An Efficient Checkpoint Protocol in Wireless Sensor Network for Reliability

Dongwon Jung[†] · Changyeol Choi[††] · Sungsoo Kim[†††]

ABSTRACT

The reliability concept of wireless sensor network is essential to get exactly actual data from the ubiquitous environment. A rollback technique for the self healing helps to increase it. However, a fault can occur in wireless sensor network when to use a previous rollback technique because it is designed just for the local system. So, checkpoint protocols are suggested in order to use a rollback technique in the network without the fault. However, there is trade-off among performance overhead, power consumption, and memory overhead for each of protocols. Hence, we suggest a novel global checkpoint protocol, so called address log based protocol(ALBP), based on an asynchronous protocol. It is a platform based protocol to reduce power consumption, performance overhead, and memory overhead which are the most of consideration in wireless sensor network.

Key Words : Wireless Sensor Network, Indirect Memory Access, Checkpoint Protocol

1. 서 론

유비쿼터스 환경에서는 사용자가 인식할 필요성 없이 현실 세계의 데이터를 얻기 위해서 무선 센서 네트워크 개념이 도입되었다. 이 때, 센싱된 데이터들의 신뢰도를 높이기 위해, 센서 프로세스의 비정상적인 오류를 치료하는 결함 허용 기법은 필수적이다. 이것을 위해 대부분의 프로세스는 재부팅으로 소프트웨어적인 결함이 치료되는 것을 기반으로, 한 노드 내의 프로세스들을 체크포인트 상태에서부터 재시작하여 시스템 오류를 치료하는 롤백 (rollback) 기술이 개발되었다[1,2]. 하지만, 무선 센서 네트워크와 같이 분산 환경처럼 독립적인 프로세스들이 여러 개의 노드에 흩어져 동작하는 환경에서, 로컬 시스템만을 고려한 롤백 기술 사용은 프로세스들간의 실행환경의 불일치(inconsistent state)을 유발시킨다[3, 4]. 특히, 불필요하게 송신되는 데이터의 양을 줄이기 위해 싱크 노드로부터 수신된 쿼리를 기반으로 센싱 단계에서 데이터를 여과하도록 설계된 Direct Diffusion과 같은 프로토콜을 사용 할 경우, 센서 프로세스의 결함을 치유하기 위한 롤백 동작은 타 노드의 쿼리 및 데이터에 대한 정보의 유실로 인하여 오동작을 유발시킨다[5]. 이로 인하여, RFID 센싱과 같이 정교함과 효율성을 요하는 분야에 있어

※ 본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅 및 네트워크 원천기반기술개발 사업의 지원에 의한 것임.
† 준 회 원 : 아주대학교 정보통신전문대학원 석사과정
†† 준 회 원 : 아주대학교 정보통신전문대학원 박사과정
††† 종신회원 : 아주대학교 정보통신전문대학원 정교수
논문접수 : 2006년 3월 28일, 심사완료 : 2006년 7월 27일

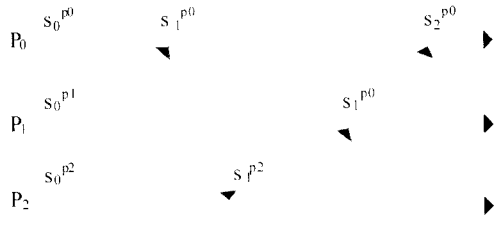
서, 대역폭의 낭비를 방지하기 위한 쿼리 기반 프로토콜의 사용은 네트워크의 신뢰성을 하락시키고, 결국, 센싱된 데이터를 믿지 못하는 결과를 초래할 수 있다. 따라서, 이러한 불일치 현상을 해결하기 위해, 노드 기반의 롤백 기술을 네트워크 차원으로 확대하는 한편, 분산 환경에서 프로세싱 환경의 일치 상태(consistency state)를 맞추기 위해 체크포인트 프로토콜들이 연구되었다[3, 4, 6, 7, 8]. 이러한 체크포인트 프로토콜들은 체크포인트 하는 방법에 따라 통신 유도(communication induced), 동기식, 비동기식 프로토콜로 분류할 수 있다[4]. 통신 유도 프로토콜은 프로세스 차원에서 구현된 프로토콜로 송신 프로세스가 수신 프로세스에게 체크포인트가 필요할 시점을 통지함으로써 불일치 현상을 막기 위해 제안되었다[9,10]. 또한, 동기식 프로토콜은 동기화를 통해 전체 프로세스를 일시에 체크포인트 하는 기법으로 불일치 현상이 발생할 여지를 제거하기 위해 개발되었다[11,12]. 마지막으로 비동기식 프로토콜은 노드 기반의 체크포인트를 사용하되 과거의 상태에서 재시작하는 프로세스가 다른 프로세스들과의 실행환경을 일치 상태로 만들기 위한 메커니즘을 지원한다[6, 7].

하지만, 무선 센서 네트워크 특유의 제약 조건들 때문에 이러한 프로토콜들을 사용할 경우 여러 가지 문제점들이 발생한다. 첫 번째 제약 조건은 무선 센서 네트워크 환경의 수 많은 센서 노드의 수이다. 통신 유도 프로토콜의 경우, 프로세스들의 수가 증가할수록 통신 횟수도 증가하기 때문에 수신 프로세스들이 체크포인트 해야 할 확률이 높아진다. 결국, 이 방식은 체크포인트를 위해 많은 시간을 소비하게 되기 때문에 실용적으로 사용하지 못한다. 두 번째 제약 조건은 센서들의 짧은 통신 거리이다. 한 노드에서 송신 전파를 발생시킴에도 불구하고 한 번의 송신으로는 네트워크에 흩어져 있는 모든 노드에서의 수신이 불가능하기 때문이다. 따라서, 직접 라우팅(direct routing)의 양단간 통신이 아닌 중간 노드를 매개체로 이용하여 통신하는 간접 라우팅(indirect routing) 방법을 사용한다. 이러한 제약조건으로 동기 조정 노드(coordinate node)에서 발생시킨 동기화 메시지는 전파 지연(propagation delay)과 전송 지연(transmission delay)으로 인해서 동기 조정 노드에서부터 거리가 멀어짐에 따라 동기화 시간에 오차가 발생한다. 따라서, 이 오차 범위를 고려하여 특정 시간 동안 전체 노드의 실행을 멈춘 후 체크포인트하는 동기화 기반의 체크포인트 프로토콜의 사용은 전체 시스템을 오랜 시간 동안의 유휴한 상태에 머물게 하여 결국 성능 하락을 초래한다[13]. 이 문제를 피하기 위해 비동기 기반의 프로토콜들이 제시되었다. 하지만 이 기법들의 특성상 프로세스들간 체크포인트 상태 외에도, 송수신된 메시지를 저장해야 하는 단점이 존재한다[14]. 프로세스 간에 전송되는 메시지 양 또는 센서 노드의 수가 많을수록 롤백을 위해 추가적으로 필요한 메모리의 양은 급증한다. 특히 비동기 방식을 기반으로 한 인과 기반 메시지 저장 프로토콜의 경우, 선 메시지 전송 후 저장 방식을 채택하여 요청(request)에 대한 응답시간을 크게 줄였지만 송신자가 모든 메시지를 저장해야 하는 단점 때문에 무선 센서 네트워크에서의 수 많은 노드와 연결된 싱크 노드와 백업에 필요한 추가 메모리에 대한 부담을 고려해볼 때 합리적이지 못하다[3,6]. 더 큰 문제는 인과 관계 즉, 메시지를 송신한 소스 프로세스와 수신한 목적 프로세스 사이의 종속관계(dependency relation)를 정확하게 정의하기 위해서, 이 기법은 메시지를 보낼 때마다 필요한 정보를 피기백(piggyback)을 통해 얻게 되어 송신 횟수가 다른 기법들에 비해 많다는 점이다. 센서 노드의 제한적인 파워량을 고려해 볼 때, 이와 같은 빈번한 메시지 송신은 많은 양의 전력을 소모하기 때문에 센서 자체의 수명을 단축 시킨다.

따라서, 본 논문에서는, 주소 기록 기반 프로토콜(Address log based protocol, ALBP)을 제시한다. 이 기법은 비동기 방식을 지원하는 플랫폼 기반 프로토콜로, 인과 기반 프로토콜을 기반으로 이것이 가진 구조적인 한계를 해결함으로써 송신 횟수를 줄이고 모든 비동기식 프로토콜들이 가지고 있는 메모리 관리 문제를 해결하기 위해 주소 기반 체크포인트 방식을 채택하고 센서에 원격 관리 요소(remote management element, RME)를 탑재하였다. RME는 자신만의 CPU와 메모리를 가진 시스템으로, 이것을 통해 센서 프로세스를 거치지 않고 필요한 데이터에 접근할 수 있는 간접 메모리 접근(indirect memory access)를 사용할 수 있을 뿐만 아니라 복구 메시지를 얻기 위한 처리 행위가 필요할 때에도 센서의 성능 하락은 발생하지 않는다[15, 16]. 따라서, 무선 센서 네트워크에서 중요한 고려사항인 파워 소모량, 메모리 사용량, 그리고 마감시간을 맞추기 위한 빠른 응답시간을 만족시킬 수 있다. 이를 위해 2장에서는 체크포인트 프로토콜의 필요성 및 인과 기반 프로토콜의 구조적 한계를 설명하고 더 나아가 송신 횟수를 줄이기 위한 방법론을 제시한다. 3장에서는 이 프로토콜의 기본적인 개념 및 새로운 프로토콜을 사용하는 시스템을 위한 ALBP 구조를 제시하며, 4장에서는 구체적인 프로토콜의 작동 원리 및 ALBP의 계층을 설명하며, 5장에서는 ALBP과 기존 프로토콜과의 성능을 비교·분석한다. 6장에서는 ALBP의 정당성을 입증하고, 마지막으로 6장에서는 본 논문의 연구 결과와 향후 연구 주제에 대해 설명한다.

2. Antecedence 그래프

노드 기반 롤백 기술의 기본 개념은 결함이 발생한 프로세스를 과거의 안전한 상태(safety state)로 되돌리는 것이다. 하지만 서로 다른 노드에서 동작을 하는 프로세스들이 네트워크를 통하여 메시지를 주고 받는 환경에서 과거로의 상태 복귀로 인한 재시작은 이전에 수행했었던 일을 반복하게 되므로 다른 프로세스들에게 중복 메시지를 보내게 된다. 따라서, 이 비정상적인 메시지를 받은 프로세스들도 결함이 발생하게 되고, 결국엔 연속적인 결함으로 인해 모든 프로세스가 초기 상태로 되돌아가게 되는 도미노 현상(domino effect)이 발생한다[4].



(그림 1) 메시지 인과 분석을 위한 Antecedence 그래프

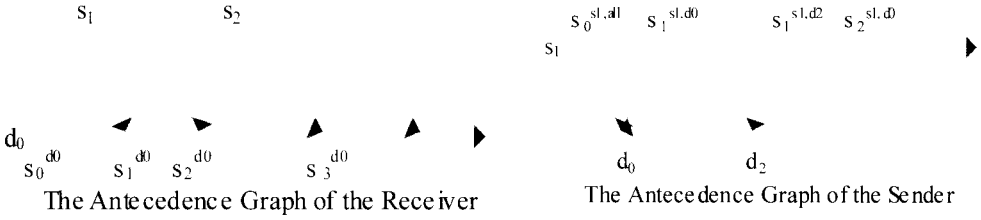
그러므로, 비동기 프로토콜의 기본 개념은 노드 기반의 롤백 기술을 사용하되, 결함이 발생한 프로세스가 체크포인트 상태에서 결함이 발생하기 전까지 수신했던 메시지들을 다시 받을 수 있도록 도와주거나 송신하는 메시지를 차단하는 것이다. 특히, 인과 분석 프로토콜은 선 메시지 전송 후 저장 방식을 위해 (그림 1)에서와 같은 Antecedence 그래프를 사용하여 메시지를 보낸 송신 프로세스를 추적하고, 복구 메시지를 요청하게 된다[8,17]. 이 그래프는 프로세스들간에 오고 간 메시지들의 송수신자 종속 관계를 나타낸다. 가로의 선을 프로세스의 실행 상태라고 했을 때, 각각의 프로세스들은 연속적인 상태 간격(state interval) σ_n^{pi} 집합으로 표현된다. 이 때, n값을 상태 간격 인덱스(state interval index, SII)라고 했을 때, 이 값은 해당 프로세스 p_i 가 메시지를 받을 때 마다 증가 된다. 이 값에 기반하여, 체크포인트 상태 σ_0^{pi} 로 돌아간 프로세스는 σ_n^{pi} 까지의 메시지를 받음으로써 일치 상태가 될 수 있다.

하지만, 개별 프로세스가 자신의 인과 분석을 위하여 이 그래프를 완성시키기 위해서는 수신자의 *SII* 값이 필요하게 된다. 따라서, 하나의 메시지에 대한 수신자의 *SII* 값을 얻기 위해 인과 분석 프로토콜은 피기백을 사용해야만 한다. 게다가, 네트워크 지연으로 결함이 발생한 프로세스에서 결함을 치료하기 위한 복구 요청(recovery request)이 오류 발생 전의 정상적인 메시지 보다 먼저 수신자에게 도착할 수 있기 때문에, 정확한 종속관계를 얻기 위해서 인과 분석 프로토콜은 복구 동작(recovery operation) 중 모든 프로세스들에게서 Antecedence 그래프를 수신 받아 자신의 그래프와의 병합 과정을 거치게 된다. 하지만, 이 방식은 무선 센서 네트워크와 같이 노드 수가 많은 경우에 매우 불합리하다.

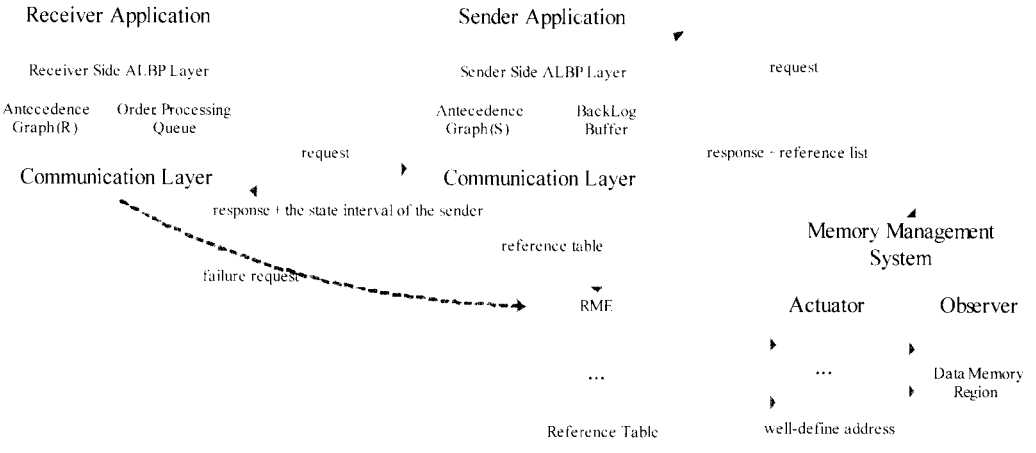
따라서, 본 논문에서는 ALBP가 수신 프로세스의 상태 간격 정보 얻기 위해 필요했었던 송신 횟수를 줄이는 한편 병합 과정을 없애기 위해, (그림 2)에서 나와 있는 것처럼 Antecedence 그래프를 두 개의 서브그래프로 분리시켰다. 이것의 기본적인 개념은 각각 프로세스의 *SII*를 피기백을 통해 정확한 값을 얻는 것이 아닌, 송신자 관점에서 이제까지 메시지를 보내온 행동에서 수신 *SII*를 유추함으로써 송신자 측에서의 불필요한 피기백을 없애는 것이다. 송신자의 Antecedence 그래프에서는 상태 간격을 $\sigma_i^{sj,dk}$ 표기할 때, 이것은 메시지를 보낸 수신 프로세스 s_j, 메시지를 받을 목적 프로세스를 d_k, 그리고 유추된 SII_i 들의 조합으로 표기한다. $\sigma_i^{sj,dk}$ 에 대해서, i 값은 s_j 가 d_k 메시지를 전송할 때만 증가되며, 다른 프로세스에게 보낼 경우는 다른 상태 간격 유추된 *SII*가 증가된다. 송신자가 메시지 $\sigma_i^{sj,dk}$ 와 같이 보낼 때마다, 수신자 측의 Antecedence 그래프에는 자신의 상태 간격 σ_i^{dk} 에 $\sigma_i^{sj,dk}$ 의 정보를 저장한다. 이 때, 수신자의 실제의 SII_i 은 오직 목적 프로세스 d_k가 메시지를 받을 때만 증가 된다. 그래서, 수신자의 Antecedence 그래프에서는 각각의 프로세스들이 받은 메시지의 수신 프로세스에서 예측한 *SII*를 알 수 있고, 자신의 상태 간격과 연결된 수신 프로세스의 상태 간격을 알 수 있다. 따라서, 송신자 Antecedence 그래프의 상태간격과 그 때 전송한 메시지에 대한 정보를 연결하면, 자신이 필요한 메시지에 대한 복구 요청을 자신의 상태 간격에 저장된 값만을 이용하여 복구 메시지(recovery message)를 얻을 수 있다. 따라서, 송신 프로세스가 수신 프로세스의 *SII*을 얻기 위해 필요로 했었던 피기백 없이, 결함이 발생한 프로세스에서는 자신에게 메시지를 보냈었던 송신 프로세스 *SII*와 자신의 *SII*를 연결함으로써 Antecedence 그래프를 완성할 수 있다. 게다가, 송신자 측의 Antecedence 그래프 갱신은 네트워크을 거치지 않고 바로 이루어지므로 병합 처리를 해야 될 필요성이 없다. 즉, 송신자의 $\sigma_i^{sj,dk}$ 에 그 당시에 송신한 메시지와 관련된 정보를 저장함으로써 ALBP는 효율적인 송신횟수를 유지하면서 적절한 복구 메시지를 요청할 수 있다.

3. ALBP 구조

ALBP 구조는 (그림 3)처럼 크게 메모리 관리 시스템(Memory Management System, MMS), RME, ALBP 계층, 그리고 한 쌍의 Antecedence 그래프로 이루어져 있다. MMS는 프로세스가 사용하는 데이터 중 지속적인 관리(persistent management)가 필요한 데이터를 자신의 메모리



(그림 2) 송신자 Antecedence 그래프와 수신자 Antecedence 그래프로 분리된 모습



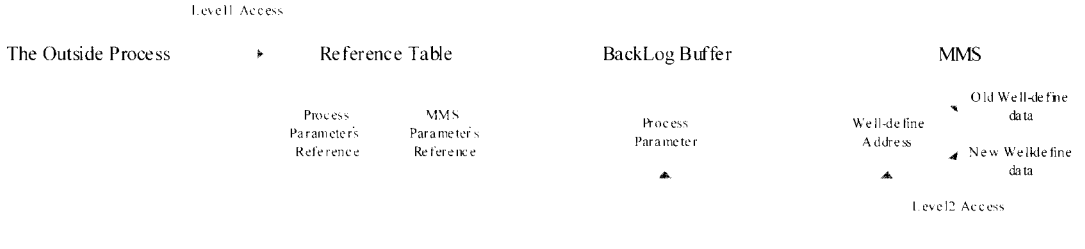
(그림 3) ALBP 프로토콜을 사용하는 기본적인 시스템 구조

에 저장한다. 그리고, 이 메모리와 해당 데이터를 얻기 위한 송신자 프로세스 사이의 요청 처리자(request handler)이면서 동시에 자신의 메모리의 데이터를 얻기 위한 요청에 대해서 응답(response)은 물론 이것의 주소에 대한 참조 목록(reference list)을 같이 넘겨주는 역할을 담당한다. ALBP 구조에서는 응답 메시지에 관여한 실제 데이터를 명시 데이터(well-defined data) 그리고 이 데이터에 대한 주소를 명시 주소(well-defined address)라고 정의한다. RME는 센서의 자원을 사용하지 않고, 자신만의 CPU와 메모리를 사용할 수 있는 센서의 메모리에 접근할 수 있는 임베디드 시스템이다. ALBP 계층은 어플리케이션 계층과 통신 계층 사이에서, 2장에서 언급했던 한 쌍의 Antecedence 그래프를 사용하여 송수신자 사이의 명확한 종속관계를 규명하는 한편 MMS로부터 생성된 참조 목록을 통하여 외부의 프로세스가 RME를 이용해 접근할 수 있도록 일련의 행동을 관리한다.

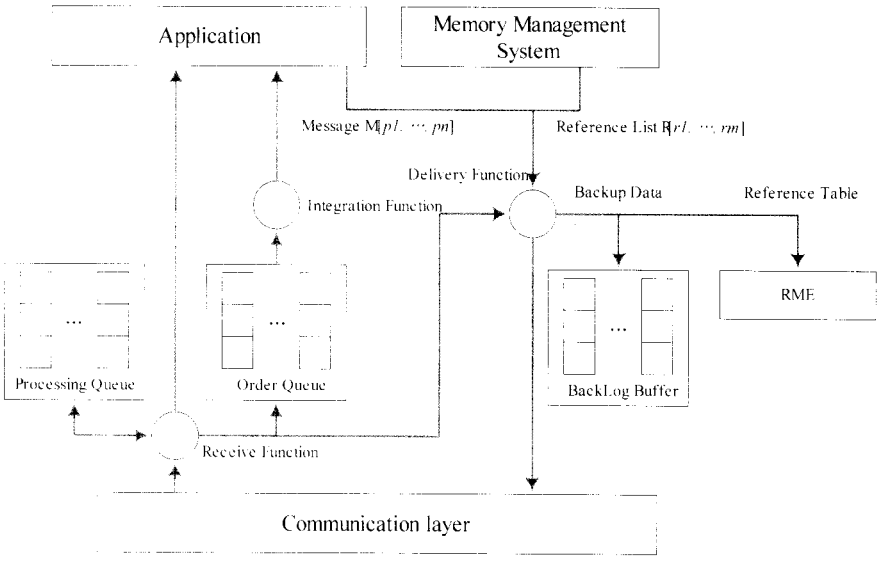
이 구조를 기본으로 하여 ALBP에서는 비동기 프로토콜들의 비효율적인 메모리 관리를 해결하기 위해 프로세스 사이의 통신에서 생성된 메시지 전체를 저장하는 것이 아닌 주소 기록 기반으로 하여 메시지의 작은 부분과 명시 주소만을 저장한다. 이 때, 센서의 프로세스나 ALBP 계층을 통한 직접 메모리 접근(direct memory access) 방식이 아닌, RME를 통한 간접 메모리 접근(indirect memory access)을 채택함으로써 복구 메시지를 얻기 위한 일련의 행동들은 센서의 자원을 소모하지 않게 되고, 이 행동을 위한 성능상의 하락은 존재하지 않는다.

이러한 간접 메모리 접근은 (그림 4)처럼 2 단계 주소 기반 메모리 접근 방식을 사용한다. 이것을 위해 우선 메시지의 명시 데이터에 대한 모든 참조 목록들의 집합인 참조 테이블(reference table)을 생성해야 한다. 그 이유는, 비록 MMS로부터 참조 목록을 얻었지만 하나의 메시지는 MMS 메모리 또는 프로세스 메모리에 저장되어 있는 파라미터들로 구성되어 있는 이상, 명시 데이터들에 대한 참조 목록이 부족하다. 더군다나, 프로세스는 요청에 의해서 자신의 메모리 상태가 변할 수 있음에도 불구하고, 본래의 명시 데이터를 백업하기 위한 관리 도구를 제공하기가 어렵다. 따라서, ALBP 계층에서는 프로세스 메모리로부터 얻어진 파라미터들을 백로그 버퍼에 저장하고 참조 목록을 생성함으로써 참조 테이블을 완성시킬 수 있다. 게다가, 특별한 조치 없이도 이 백업 데이터들은 프로세스 메모리와 독립적으로 저장되기 때문에 지속성(persistent)를 유지할 수 있다. 이 때, 명시 데이터나 이것을 참조하고 있는 명시 주소의 경우, MMS 메모리 상태 자체도 요청에 의해서 변할 수가 있기 때문에 잘못된 메모리 참조를 막기 위해 명시 데이터를 백업한 뒤 명시 주소 자체를 수정해야만 한다. 하지만, 명시 주소와 고정적으로 연결된 참조 목록은 수정될 필요가 없으므로 외부의 프로세스들은 2 단계 주소 기반 메모리 접근으로 적절한 명시 데이터를 얻을 수가 있다.

반면에, MMS의 메모리를 수정할 우려가 있는 요청을 처리하기 위해 관찰자(observer) 컴포넌트와 작용자(Actuator) 컴포너트를 MMS 안에 배치했다. 관찰자는 감시(monitoring)



(그림 4) 2 단계 주소 기반 메모리 접근



(그림 5) ALBP 계층의 내부 구조

컴포넌트로, MMS가 요청을 받을 때마다 이것을 실행하기 전에 수정 요청(modification request)인지를 판단하는 역할을 담당한다. 반면, 작용자는 이러한 수정 요청이 메모리에 영향을 주기 전에, 바뀌어질 데이터를 다른 곳으로 저장한 뒤 명시 주소의 값을 바꾸는 역할을 수행한다.

이러한 컴포넌트들 이외에도 ALBP는 2단계 복구 동작을 위해 Order 큐와 Processing 큐를 사용한다. 1단계 복구 동작은, 체크포인트 상태부터 결함 발생 전까지 수신했던 메시지를 다시 결함이 발생한 프로세스가 받을 수 있도록 하는 것이다. 하지만, 복구 요청이 브로드캐스팅 형식으로 전파되는 이상 복구 메시지들은 정렬되지 않는 상태로 수신되므로, 이것을 정렬하기 위해 Order 큐가 필요하다. 2단계 복구 동작은, 결함이 발생한 프로세스가 복구 동작 중 처리하지 못한 정상적인 메시지(normal message)를 순서대로 수신 받는 것이다. 따라서, Processing 큐에 이러한 메시지들과 같이 전송된 송신자의 상태 간격 정보를 저장함으로써, 복구 동작이 끝난 후 간접 메모리 접근 방식을 통해 필요한 메시지를 순서대로 얻을 수 있다.

4. ALBP 동작

이 장에서는 ALBP 계층의 구체적인 컴포넌트와 동작(operation)을 살펴본다.

4.1 ALBP 계층 구조

(그림 5)는 ALBP 계층의 내부 구조로, 기본적으로 3 개의 주요 함수들이 존재한다. Delivery 함수, receive 함수, integration 함수 들이 그것들이다. 수신 프로세스가 메시지를 보낼 때마다 delivery 함수가 호출되며, 선 메시지 전송 후 저장을 위하여 목적 프로세스의 송신자 측의 유추된 *SII* 값을 계산한 뒤, 메시지 안에 송신 상태 간격 정보를 저장하여 전송한다. 한편, receive 함수의 주요 목적은 통신 계층으로부터 올라온 메시지를 수신 프로세스에 전달하는 것이다. 이것 외에도, 메시지를 받음과 동시에 새로운 종속 관계를 수신 Antecedence 그래프에 적절하게 삽입한다. Integration 함수는 RME로부터 얻어진 비트 스트림 형태의 복구 메시지를 적절한 파라미터 형식으로 변환시켜 주는 역할을 하며, 변환된 파라미터들을 다시 하나의 메시지로 결합시켜 주는 역할을 한다.

4.2 기본 동작

ALBP 프로토콜의 통상적인 동작은 메시지 송수신뿐만 아니라 delivery 함수와 receive 함수를 통해서 종속관계를

```
function Delivery(M, [p1, ... , pn], R, [r1, ... , rm] )
// M is a message, with n parameters
// R is a reference list, with m references
// s is the identification of a source process
// d is the identification of a destination process

SII_sd ← SII [d] ++
save SII_sd, s, and d in σ_message state
insert a new dependency relation in Antecedence graph[d]
   with σ_message state
deliver a message with SII_message to d


If (M is not null)
for all parameters p of message
  if (p's reference r is in R)
        save p's reference r in the Reference_Table[d,
        SII_sd] in i order
  else
        save p to BackLog Buffer and get reference r
        save a reference r to the Reference_Table[d,
        SII_sd] in i order

if (the sensor node is RME model)
   deliver the Reference_Table[d, SII_sd] i to RME
else
   save the Reference_Table[d, SII_sd] i in BackLog buffer
```

(코드 1) Delivery function의 통상적인 동작

명확하게 각각의 Antecedence 그래프에 갱신하는 것이다. Antecedence 그래프에 종속관계를 삽입한다. 그 후, 참조자 테이블을 생성한다. 완성된 테이블은 기본적으로 RME로 전송하지만 RME을 탑재하지 않는 센서의 경우, 백로그 버퍼에 저장한다. 이 때, RME는 수신 프로세스 측에서 유추된 *SII* 값으로 구별 가능하도록, SII_{sender}, *d*, 그리고 *s* 의 값으로 분류시킨다.

(코드 2)는 수신자 측의 Receive 함수의 동작을 기술하고 있다. 수신 측에서는 메시지를 받음과 동시에 수신 Antecedence 그래프에 새로운 종속관계를 삽입시킨다. 이 때, 수신자 측에서는 적합한 참조 테이블을 참조하기 위해 각각의 상태 간격을 구분 할 수 있는 값 SII_{sender} 와 수신 프로세스의 *id*를 저장한다.

```
function Receive(M, SII_sender, s, d)
// s is the identification of a source process
// d is the identification of a detination process
// SII_sender is relative the Sender SII

SII_current ← max SII from Antecedence graph(σ_SII^s)
save s and SII_sender in σ_current_SII
save σ_SII in Antecedence graph and modify SII++
Pass M to a destination process d
```

(코드 2) Receive function의 통상적인 동작

4.3 복구 동작

복구 동작들은 ALBP 계층에서 프로세스의 결함 발생이 확인됐을 경우 활성화된다. (코드 3)에 기술된 복구 함수는 결함이 발생한 프로세스가 오류 발생 전까지의 수신했던 메시지를 순서대로 전달하는 기능을 담당하고 있다.

우선적으로, recovery 함수는 결함이 발생한 프로세스가 확인되는 순간, 해당 프로세스의 delivery 함수을 비활성화 시키는데, 그 이유는 복구 동작 중에 발생하는 모든 메시지는 이미 다른 프로세스들에게 영향을 주었기 때문에 시스템 차원의 결함이 발생하지 않도록 비정상적인 메시지를 송신하지 못하도록 해야 한다. 그 다음, 결함이 발생한 프로세스 종류를 판별해야 한다. 일반 프로세스의 경우는 전체 메시지가 MMS의 메모리나 백로그 버퍼에 안전하게 저장되어 있다. 하지만, MMS 프로세스가 결함이 발생하여 롤백을 수행하면 메모리 상태가 과거의 상태로 바뀌게 된다. 따라서, 명시 주소나 명시 데이터 값 모두 잘못된 메모리 영역을 포인팅하게 된다. 이러한 문제를 막기 위해서 만약 MMS 프로세스가 결함이 발생했다면 복구 동작을 시작하기 전에 명시 데이터들을 모두 백업 서버로 옮긴 후 새로 명시 주소를 작성해야만 다른 프로세스의 결함에 대해서 올바른 복구 메시지를 전달할 수 있다. 게다가, 복구가 완료되었는지 판별하기 위해서 Order 큐에서 정렬된 복구 메시지가 결함 프로세스로 하나씩 전송 될 때마다 $order_{recover}$라는 복구 상태 간격의 *SII*을 증가시킨다. 만약 변화된 값이 결함이 발생하기 전의 최대 *SII* 값인 $order_{max}$와 같다면 복구 동작을 요청

```
function Recovery(p)
// p is a failure process

disable the delivery function of p
SII_max ← max SII from Antecedence graph(σ_SII^s)
SII_recovery ← 0

if (p is a MMS process)
     transfer the content of MMS memory which is
   indicated by the well-defined address to backup
   server
   re-calculate the well-defined address

start the recovery operation of p

broadcast a failure request to the other processes
if the identification of a process b is in Antecedence
graph(σ_SII^s)
    if (the sensor node of b is RME model)
        send the failure request with (p, SII_sender)
      to the RME of b
    else
        send the failure request with (p, SII_sender)
      to the ALBP layer of b

while (SII_max > SII_recovery)
   wait until p receives the rest of recovery
 messages

while (The Processing Queue is not null)
  for all source process b in the Processing Queue
   if (the sensor node of b is RME model)
        send the failure request to the RME of b to
      get recovery message
   else
        send the failure request with (p, SII_sender)
      to the ALBP layer of b

enable the delivery function
end the recovery operation of p
```

(코드 3) Recovery function의 복구 동작

한 프로세스는 모든 메시지를 올바로 전송 받았다고 할 수 있다. 마지막으로, Processing 큐 안에 있는 추가 메시지 정보를 기반으로 메시지 송신자에게 복구 동작 동안 수신되었던 메시지를 RME혹은 ALBP 계층을 통해 가져오게 된다.

(코드 4)는 복구 동작 중 receive 함수의 동작을 나타내고 있다. 복구 요청이 전파된 후, 3가지 종류의 메시지가 도착할 수 있다. 첫 번째로 정상적인 메시지가 수신될 수 있다. 이것은 전파 지연으로 인해서 복구 요청이 다른 노드의 프로세스로 제때 도착하지 못하여, 결함 발생을 인식을 못하고 메시지를 보낸 결과이다. 이럴 경우 Processing 큐에 메모리 효율을 위해서 전체 메시지가 아닌 상태 간격 정보를 저장한다. 두 번째로 복구 메시지의 수신될 경우, Order 큐로 보내어 순서에 맞도록 정렬시키고, 결함 프로세스로 전송한다. 세 번째로 복구 요청 메시지가 수신 함수에 도착하였다면, 이 센서 노드는 non-RME 모델로 ALBP 계층에서 직접 복구 메시지를 전달해야 한다. 따라서, 송신 함수를 호출하여 필요한 참조 테이블을 사용하여 복구 메시지를 전달한다.

```
function Receive(M, SII_sender, s, p)
// p is a failure process

SII_processing ← 0
if (p is in failure state)
   if (M is a normal message)
      drop message
      save sender_state and the identification of a
      source process s in the
      Processing_Queue[SII_processing]
      SII_processing ++
   else if (M is a recovery message)
      save M in the Order_Queue
else if (M is a failure request)
        call the delivery function to transfer the
      recovery message by the Reference_Table[s,
      SII_sender]
```

(코드 4) Receive function의 복구 동작

```
function Integration(p)
// p is a failure process

while (SII_max > SII_recovery)
   if (the SII_sender and the id of s about next message
         in the Order_Queue are equal to σ_i^sj,dk in
      Antecedence Graph)
         convert a bit stream in the Order_Queue to
      a proper message
      deliver this message to application
SII_recovery++
```

(코드 5) Integration function의 복구 동작

(코드 5)는 Integration 함수의 복구 동작을 기술하고 있다. Integration 함수는 복구 메시지를 순서에 맞추어서 복구 동작을 수행중인 프로세스로 전달하는 역할 이외에도 RME를 통하여 수신 받은 비트 스트림 형태의 복구 메시지를 적절한 파라미터 타입으로 변환시키고 통합 과정을 거쳐 하나의 메시지로 만든다.

5. 시뮬레이션

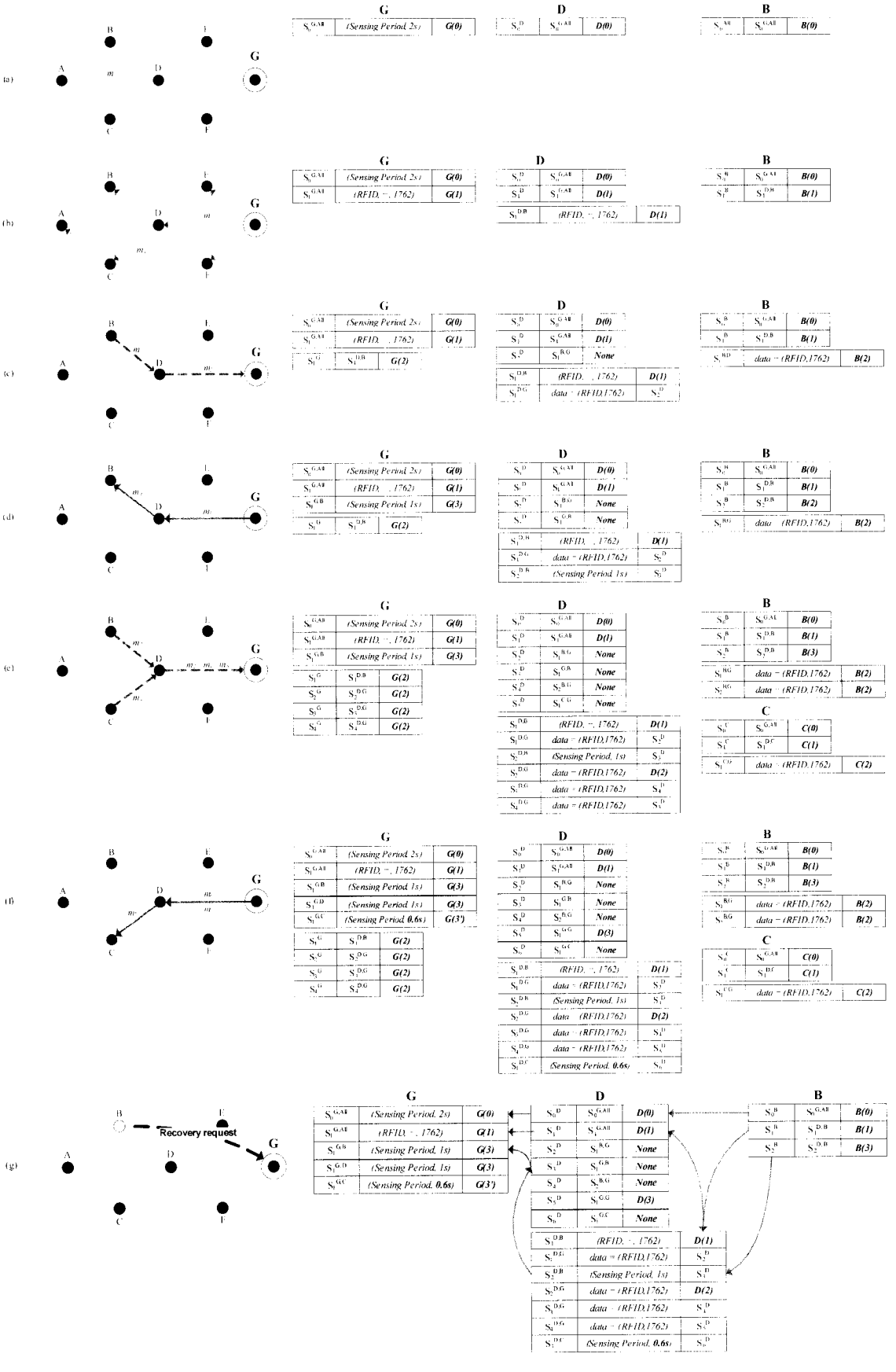
이 장에서는 ALBP의 시뮬레이션 시나리오 및 환경을 제시하며, 기존 프로토콜과의 성능을 비교·분석한다.

5.1 ALBP 시뮬레이션 시나리오

(그림 6)의 왼쪽 그림은 센서의 분포와 데이터 흐름도이며, 오른쪽 도표는 Antecedence에 저장되는 수신 상태 간격 σ_i^{dk} 와 송신 상태 간격 $\sigma_i^{sj,dk}$ 에 저장되는 값이다. σ_i^{dk} 에는 수신 프로세스의 *SII*, 수신 메시지에 포함된 송신자의 *SII*, 그리고 MMS 혹은 메모리에 저장된 수신 메시지의 주소값이 포함된다. 이와 유사하게, $\sigma_i^{sj,dk}$ 에는 송신 프로세스의 *SII*, 데이터의 내용 및 데이터가 저장된 MMS의 주소값을 포함하고 있다.

센서 노드들이 (그림 6(a))와 같이 분포되었다고 가정하자. 그 후, (그림 6(b))처럼 싱크 노드(G)에서 특정 RFID를 센싱하도록 쿼리(m_1)를 다른 센서들로 전송할 때, m_1 대한 백업 메시지의 참조 테이블[G(1)]을 생성하도록 Delivery 함수가 호출된다. 그 후, m_1을 수신하는 노드들은, Receive 함수를 통하여 새로운 인과 관계를 수신 Antecedence 그래프에 추가하는 한편 m_1을 자신의 프로세싱에 반영한다. 이 때, 각각의 수신 노드들은 m_1을 가장 빨리 송신한 노드에 대해서 그래디언트 패스(gradient path)를 설정하게 된다. 따라서, (그림 6(c))에서처럼 노드 B가 쿼리에 적합한 데이터를 센싱하고 싱크 노드 G로 해당 데이터(m_2)를 송신 하기 위해서는, 그래디언트 패스가 설정된 노드 D로 m_2를 전송해야 한다. 이 때, 중간 전달 노드 D는 단순히 이것의 참조 테이블 주소[B(1)] 만을 자신의 메모리에 저장한다. (그림 6(d))에서는 노드 B의 센싱 데이터를 좀 더 빈번히 수신 받기 위해서, 센싱 주기 수정 메시지(m_3)를 Delivery 함수를 통해 전송하는 한편 이것의 참조 테이블[G(2)]을 생성한다. 하지만, (그림 6(e))처럼 B이외의 다른 노드들(C, D) 또한 쿼리에 적합한 데이터(m_4, m_5)를 센싱하여 싱크 노드 G로 전송할 경우에는, 노드 D로 노드 B에게 전송했던 똑같은 센싱 주기 수정 메시지(m_6)를 전송할 필요성이 있다. 이 경우 (그림 6(f))처럼, 싱크 노드 G에서 호출된 Delivery 함수는 m_6를 송신할 때 새로운 메시지를 백업하지 않고, m_6 에 대한 참조 테이블은 명시 주소(G3)를 가리키도록 한다. 하지만, 노드 C의 쿼리처럼 메시지의 파라미터가 수정되었을 경우, 싱크 노드 G의 MMS는 새로운 메시지(m_7)를 MMS(G3')에 저장하는 한편 기존에 저장되어 있던 참조 테이블의 명시 주소값을 적절히 수정하게 된다. 이와 유사하게 ALBP를 사용함으로써, 노드 B가 싱크 노드 G에게 이전에 전송했던 데이터(m_3)를 동일하게 전송하고 있는 상황에서는 추가적인 메시지 백업이 일어나지 않는다. 이러한 일련의 과정 후, (그림 6(g))는 노드 B가 결함이 발생하여 이전에 수신 받았던 메시지들을 재수신 받기 위해서, 호출된 Recovery 함수는 수신 Antecedence의 메시지 송신자인 노드 D에 복구 요청을 하는 모습을 보여주고 있다. 만약, 복구 요청을 받은 노드 D의 메모리에 해당 데이터가 유효(valid)한 형태로 존재할 경우, 이 값을 노드 B로 전송한다. 하지만, 유효하지 못하다면(invalid), 노드 D는 노드 G로 해당 노드 B의 복구 요청을 전송하여 노드 B에게 복구 메시지를 전송할 수 있도록 한다. 이 때, 이러한 복구 요청 및 메시지 재전송은 각각의 센서에 탑재된 RME를 통하여 이루어지기 때문에, 결함 복구에 필요로 하는 자원의 소모를 최소화할 수 있다[15,16].

따라서, 본 논문에서 제안한 Antecedence 그래프를 사용함으로써 개개 센서의 consistency를 맞추기 위한 SII의 피기백을 막는 한편, ALBP 구조를 사용함으로써 메시지 백업에 필요한 메모리의 양을 줄였다. 특히, ALBP 구조의 사용은 동일한 쿼리를 수 많은 노드에 전파하는 무선 센서 네트워크와 같은 환경에서 효율적이라 판단된다.

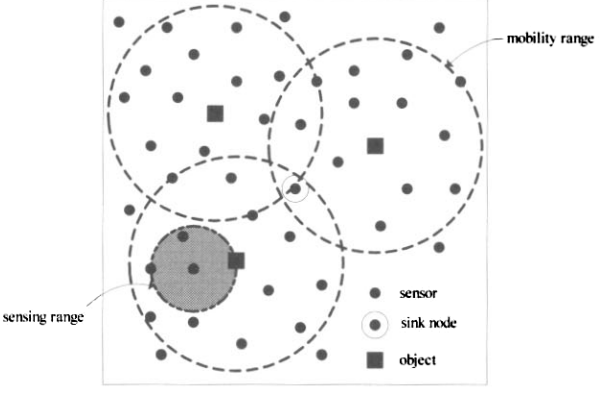


(그림 6) ALBP 동작

5.2 시뮬레이션 결과

이 절에서는 5.1절에 제시된ALBP 동작을 기반으로 수행된 시뮬레이션 결과를 인과 기반 메시지 저장 프로토콜[6]의 성능과 비교·분석한다. 시뮬레이션 환경은 (그림 7)처럼 일정 범위 안에 센서들을 랜덤하게 분포시키고, 3개의 서로 다른 RFID를 보유한 물체를 일정한 영역 안에서 이동하도록 설정하였다. 따라서, 각각의 고정된 위치에 있는 센서들은 물체가 이동함에 따라 RFID를 지속적으로 센싱하지 못하게 된다. 이와 같은 환경에서, 싱크 노드는 효율적으로 송신횟수를 조절하기 위해, 각각의 노드 별로 센싱 사거리 안의 물체의 수에 따라 센싱주기를 가변적으로 변경하도록 설정하였다. 시뮬레이션을 위한 기본 파라미터 설정값 [18, 19]은 <표 1>과 같다.

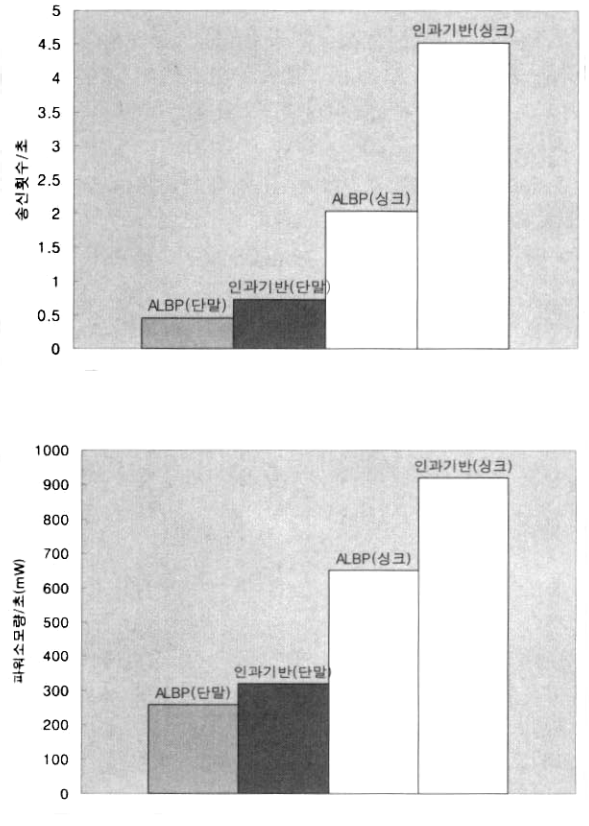
(그림 8)과 (그림 9)는 ALBP와 인과 기반 메시지 저장 프로토콜의 성능을 비교·분석한 그래프이다. 그래프에서 나타나듯이ALBP를 사용하였을 때 송신 횟수, 파워소모량, 그리고 메모리 오버헤드가 낮은 것으로 나타났다. 파워 소모량이 낮은 이유는, 본 논문에서 제안된 Antecedence 그래프를 사용하여 노드의 송신횟수가 낮아지므로 센서의 활동 상태(active state)의 시간을 줄일 수 있고, 이로 인하여 수신 노드에서의 메시지 수신 횟수 또한 적어져 듣기 상태(listen state)의 시간도 감소하였기 때문이다. 따라서, 센서의 전체 활동 시간 중 파워 소모량이 낮은 유휴한 상태(idle state)가 차지하는 시간이 늘어나므로 단위 시간 동안의 파워 소모량은 줄어든다. 뿐만 아니라, 싱크 노드에서는 동일한 쿼리를 여러 개의 단말 노드로 전송함에 따라 필요한 메시지 백업의 양을 줄일 수 있고, 단말 노드와 같이 일정 시간 동안 동일한 데이터를 지속적으로 센싱하여 싱크 노드로 전송하는 경우, 여러 번의 송신에 대해서 하나의 메시지만을 백업함으로 메모리 오버헤드를 크게 줄일 수 있다.



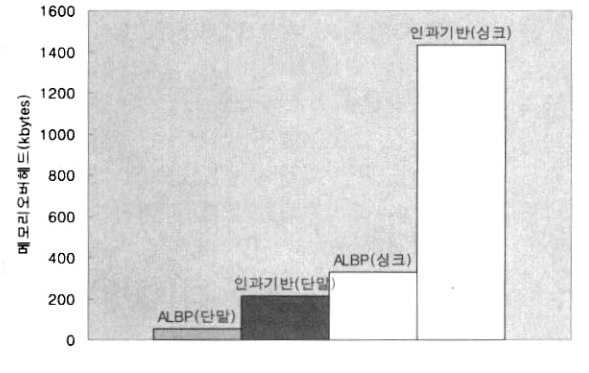
(그림 7) 시뮬레이션 환경

(표 1) 시뮬레이션 파라미터

sensing range	10 m
the number of sensor nodes	50
the number of sink nodes	2
the size of network	200 * 200 m^2
sensing period(the number of object)	0.3 s (3), 0.6 s (2), 1 s(1), 2 s(0)
the size of a packet	32 kbytes
checkpointing period	10 s
power consumption (state)	1040 mW (active), 400 mW (listen), 10 mW (idle)



(그림 8) 송신횟수 및 파워소모량



(그림 9) 메모리 오버헤드

6. Correctness

이 장에서는 ALBP 프로토콜의 정당성에 대해서 증명하도록 한다.

Lemma 1. 본 논문에서 제시된 한 쌍의 Antecedence 그래프는 기존 Antecedence 그래프의 모든 정보를 담고 있다.
(증명) 다른 말로, 모든 종속관계가 한 쌍의 Antecedence 그래프 안에 포함 되어있는 여부로 바꾸어 말할 수 있다. 수신 Antecedence 그래프의 상태 간격을 α_i^k, 송신 Antecedence 그래프의 상태 간격을 α_i^{ij} 라고 했을 때 (그림 2)에서 노드 단위의 종속관계를 표현한 하나의 선은 $j=k$일 때 연결될 수 있다. 이것은 송신자와 수신자 사이에 모든 종속관계가 표현된 것을 의미하며, 모든 프로세스들에 대해서 한 쌍의 Antecedence 그래프를 연결하면 (그림 1)에서와 같이 모든 종속 관계를 나타낼 수 있다.

Lemma 2. 통신 계층에 이상이 없는 한, 복구 요청으로 인한 복구 메시지의 손실은 없다.
(증명) 이 문제는 한 쌍의 Antecedence 그래프가 모든 종속관계를 담고 있는 여부를 판별하는 것이다. 기본적으로 ALBP 기본 동작 중에는 새로운 종속 관계를 Antecedence 그래프에 추가한다. ALBP복구 동작 중에 정상적인 메시지를 처리하지 못해 발생할 수 있는 종속 관계의 부재를 막기 위해서, Processing 큐에 저장된 상태 정보를 통해 해당 메시지를 순서대로 수신 받고, receive 함수에서 이 메시지들에 대한 종속관계를 기록한다.

Lemma 3. ALBP 프로토콜은 비동기 방식으로 동작이 가능하다.
(증명) 이 문제는 동기를 맞추지 않은 상황에서, 한 쌍의 Antecedence 그래프를 사용한 비동기 방식의 메시지 로그는 불일치 상태를 발생시키지 않음을 판별하는 것이다. 불일치가 발생하는 경우는 다음의 2가지이다

경우 1 : 정상 프로세스가 비정상적인 메시지 혹은 중복 메시지를 결함이 발생한 프로세스로부터 받았을 경우 발생할 수 있다. 하지만 복구 단계 중의 송신 함수는 ALBP 프로토콜에 의해 활동하지 못하게 되고 따라서 비정상적인 메시지를 발생시킬 우려도 없다.

경우 2 : 결함이 발생한 프로세스가 복구 메시지를 정확하게 받지 못했을 경우 발생할 수 있다. 하지만 lemma 1에 의해서 불완전한 한 쌍의 Antecedence 그래프는 없다. 따라서, 순차적으로 필요한 복구 메시지를 빠짐 없이 전달 가능하기 때문에 성공적으로 일치 상태를 회복할 수 있다.

Lemma 4. 결함이 발생한 프로세스의 상태 간격 인덱스 SII는 복구 동작이 종료된 후, 이 값을 정상적으로 회복시킬 수 있다.
(증명) Lemma 2,3에서 증명한 바와 같이 결함이 발생한 프로세스는 성공적으로 복구 메시지를 받을 수 있다. 따라서, 다른 프로세스들과의 일치 상태 SII값인 $SII_{max+processing}$ 의 값은 체크포인트 상태 SII_0 로부터 복구 메시지를 받음으로써 SII_{max} 되고, Processing 큐 안의 정보를 사용하여 추가적인 메시지를 수신 받으므로써 $SII_{processing}$ 가 될 수 있다.

Theorem 1. ALBP 프로토콜을 사용함으로써 백업 서버가 살아 있는 한 전체 시스템의 프로세스의 실행 환경은 일치 상태를 보장받을 수 있다.
(증명) 기본적으로 결함이 발생한 프로세스는 복구 메시지를 한 쌍의 Antecedence 그래프를 이용하여 받을 수 있다(lemma 1,2,3) 단, MMS프로세스의 결함 발생 처리를 위해 명시 데이터를 보존할 백업 서버가 살아있다면 해당 프로세스는 성공적으로 일치 상태를 보장받을 수 있다(lemma 4)

7. 결 론

본 논문에서는, 무선 센서 네트워크가 가지고 있는 제한 조건을 고려하면서 기존에 개발된 체크 포인트 프로토콜들의 문제점들을 제시하였다. 이 문제점들을 해결하기 위해 본 논문에서 제안한ALBP프로토콜은 인과 분석 프로토콜의 기본 모델을 수정하여 전송 횟수를 줄임으로써 전력 소모를 줄이는 한편 주소 기반 기록 개념 및 RME의 탑재로 메모리 오버헤드 문제를 해결하였다. 특히, RME 스스로 센서의 자원 소모 없이 센서 메모리에 접근이 가능하기 때문에 타 프로토콜과 비교하여 성능 하락의 여지는 없다.

이러한 모든 면을 고찰해 볼 때, 무선 센서 네트워크에서 중요한 고려사항인 전력 소모, 메모리 사용량, 그리고 마감시간을 맞추기 위한 빠른 응답시간을 만족할 수가 있도록 ALBP 는 설계되었기 때문에 이 프로토콜은 무선 센서 네트워크에서 실용적인 체크포인트 프로토콜이 될 수 있다.

참 고 문 헌

[1] M. Burrows, et al., "Compressed Differences: An Algorithm for Fast Incremental Checkpoint," Proceedings of Architectural Support for Programming Languages and Operating Systems, pp. 2-9, Oct. 1992.

[2] M. Prvulovic, Z. Zhang, and J. Torrellas, "ReVive: Cost-Effective Architectural Support for Rollback Recovery in Shared-Memory Multiprocessors," Proceedings of Computer Architecture, pp. 111-122, May 2002.

[3] A. Bouteiller, et al., "Coordinated Checkpoint versus Message Log for Fault Tolerant MPI," Proceedings of IEEE International Conference on Cluster Computing, pp. 242-250, Dec. 2003.

[4] M. Elnozahy and L. Alvisi, "A Survey of Rollback-recovery Protocols in Message-passing System," ACM Computing Surveys, Vol. 34, pp. 375-409, Sep. 2002.

[5] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," Proceedings of Mobile Computing, pp.56-67, Aug., 2000.

[6] E. Elnozahy and W. Zwaenepoel, "Manetho: Transparent Rollback-recovery with Low Overhead, Limited Rollback and Fast Output," IEEE Transactions on Computers, Vol.41, No.5, pp.526-531, May, 1992.

[7] L. Alvisi and K. Marzullo, "MessageLogging: Pessimistic, Optimistic, Causal, and Optimal," IEEE Transactions on Software Engineering, Vol.24, No.2, pp.149-159, Feb., 1998.

[8] T. Juang and S. Venkatesan, "Crash Recovery with Little Overhead," Proceedings of IEEE International Conference on Distributed Computing Systems, pp.454-461, May, 1991.

[9] J. Cao, Y. Li, and M. Guo, "Process Migration MPI Applications based on Coordinated Checkpoint," Proceedings of International Conference on Parallel Distributed Systems, pp.306-312, July, 2005.

[10] J. Tasi, "An Efficient Index-based Checkpointing Protocol with Constant-size Control Information on Messages," IEEE Transactions on Dependable and Secure Computing, Vol.2, No.4, pp.287-296, Oct., 2005.

[11] J. Helary, et al., "Communication-based Prevention of Useless Checkpoints in Distributed Computations," Journal of Distributed Computing, Vol.13, No.1, pp.29-43, Jan., 2000.

[12] A. Krohn, et al., "TOMAC - Real-time Message Ordering in Wireless Sensor Networks Using The MAC Layer," Proceedings of International Conference on Networked Sensing Systems, pp.377-381, June, 2005.

[13] L. Alvisi, et al., "An Analysis of Communication Induced Checkpointing," Proceedings of IEEE International Conference on Fault-Tolerant Computing, pp.242-249, June, 1999.

[14] D. Johnson and W. Zwaenepoel, "Sender-based Message Logging," Proceedings of International Symposium on Fault-Tolerant Computing, pp.14-19, July, 1987.

[15] A. Bohra, et al., "Remote Repair of Operating System State using Backdoors," Proceedings of IEEE International Conference on Autonomic Computing, pp.256-263, May, 2004.

[16] F. Sultan, et al., "Nonintrusive Remote Healing Using Backdoors," Proceedings of Algorithms and Architectures for Self-Managing Systems, pp.69-74. June, 2003.

[17] C. Rao, L. Alvisi, and H. Vin, "The Cost of Recovery in Message Logging Protocols," Proceedings of IEEE Symposium on Reliable Distributed Systems, pp.10-18, Oct., 1998.

[18] Y. Wang and H. Wu, "DFT-MSN: The Delay Fault Tolerant Mobile Sensor Network for Pervasive Information Gathering," Proceedings of IEEE International Conference on Computer Communications, pp.25-38, Apr., 2006.

[19] E. Shih, et al., "Physical Layer Driven Protocol and Algorithm Design for Energy-Efficient Wireless Sensor Network," Proceedings of Mobile Computing and Networking, pp.272-287, July, 2001.

정 동 원

e-mail : windsage@ajou.ac.kr
2005년 아주대학교 정보통신공학과(공학사)
2005년~현재 아주대학교 정보통신전문대학원 석사과정
관심분야 : 유비쿼터스 컴퓨팅, 분산 알고리즘, 결함허용 시스템 등

최 창 열

e-mail : clchoi@ajou.ac.kr
1999년 아주대학교 정보통신공학과 (공학사)
2000년 아주대학교 정보통신전문대학원 (공학석사)
2002년~현재 아주대학교 정보통신전문대학원 박사과정
관심분야 : 유비쿼터스 컴퓨팅, 오토노믹 컴퓨팅, 고가용성 시스템, 미들웨어 등

김 성 수

e-mail : sskim@ajou.ac.kr

1982년 서강대학교 전자공학과(공학사)

1984년 서강대학교 전자공학과(공학석사)

1995년 Texas A&M University 전산학과 (공학박사)

1983년~1996년 삼성전자/삼성종합기술원 수석연구원

2002년~2003년 Texas A&M University 교환교수

1996년~현재 아주대학교 정보통신전문대학원 정교수

2006년~현재 아주대학교 정보통신전문대학원장

관심분야 : 유비쿼터스 컴퓨팅 및 네트워크, 오토노믹 컴퓨팅, 컴퓨팅 및 시스템, dependable 시스템 및 네트워