MPICH-G2 상에서 안전한 송수신을 위한 확장 구조

박금례[O] 최시열 박성용 권오영[†] 박형우[‡]
서강대학교 컴퓨터학과 분산시스템 연구실 한국기술교육대학교[†] 한국과학기술정보연구원[‡]
{namul[O], adore}@sogang.ac.kr, parksy@ccs.sogang.ac.kr, oykwon@kut.ac.kr[†],
hwpark@kisti.re.kr[‡]

Extensions to the MPICH-G2 Architecture for Secure Communication

Kumrye Park[O] Siyoul Choi Sungyong Park Ohyoung Kwon[†], Hyoungwoo Park[‡]
Distributed Computing and Communication Lab., Dept. of Computer Science, Sogang Univ.
Korea University of Technology and Education[†]
Korea Institution of Science and Technology Information[‡]

요 약

그리드 상에서 사용되는 MPICH-G2는 효율적이고 확장이 용이한 구조를 지니고 있지만 최근 그리드 상에서 요구되는 병렬 입출력, 보안 및 QoS와 같은 기능을 제공하지 못하고 있다. 본 논문은 MPI의 attribute caching 방법을 이용하여, MPI의 표준 인터페이스를 훼손하지 않으면서 MPICH-G2상에서 메시지의 송수신 시에 암호화 및 복호화 등의 보안 기능을 수행할 수 있는 확장된 구조 및 구현 방안을 제시한다. 본 논문에서 제시된 구조 및 구현 방안은 MPICH-G2 상에 향후 추가될 수 있는 서비스에 의존적이지 않기 때문에, QoS나 메시지 압축과 같은 기능 추가를 위해 손쉽게 사용될 수 있다.

1. 서 론

그리드 컴퓨팅에서는 기본적인 고속통신 서비스 이외에도 지리적으로 떨어진 시스템 간의 통신상의 보안 서비스를 제공해야 하며, 응용 프로그램에 양질의 서비스를 제공하기 위해서 응용 프로그램이 요청하는 다양한 QoS를 제공하고, 응용 수행에 필요한 대규모의 데이터 처리 및 오류 허용을 제공할 수 있어야 한다. 하지만 현재 그리드에서 이용되는 MPICH-G2는 WAN상의 통신 효율에 중점을 두고 있으며 위에서 언급된 기능을 효과적으로 지원하지 못하고 있다. 그러므로 MPICH-G2에 보안, QoS, 대규모 데이터 등의 지원이 가능하도록 기능을 확장하는 것이 필요하다.

MPICH-G2에 QoS 기능을 제공하기 위한 방안으로 MPICH-GQ[2]가 제안되었지만, 상세한 구조 및 구현 방안이 자세히 소개되지 않아 실제로 기능을 추가하는데 어려움이 있다.

본 논문에서는 MPICH-G2에서 안전한 송수신을 위한 기능 확장 구조(본 논문에서는 MPICH-GS라 명명한다.) 및 구현 방안에 대하여 기술한다. MPICH-GS는 기존 MPICH-G2의 계층 구조에서 MPI 하위 계층에 Security 에이전트와 Security 라이브러리를 추가함으로써 물리 계층으로 메시지를 전송하기 전에 메시지를 암호화하고, 수신 시에 받은 메시지를 복호화 함으로써, 안전한 MPI 메시지 송수신을 보장하게 된다. 본 논문에서 소개하는 MPICH-GS의 구조 및 구현 방안은 QoS나 메시지 압축과 같은 기능을 추가하는데 쉽게 사용될 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 MPICH-G2를 확장한 MPICH-GS의 구조를 제시하고, MPI의 attribute caching에 대해 소개한다. 또한 attribute caching을 이용하여 실제적인 메시지 송수신 과정을 구현한 Security 에이전트와 응용 프로그램 예제를 살펴본다. 3장에서는 본 논문의 결론 및 향후 연구에 대하여 기술한다.

2. MPICH-GS 구조

MPICH-GS 설계의 목표는 표준 기반의 프레임워크 안에서 메시지의 안전한 송수신을 보장하는 것이다. 이러한 목표에 따라 MPI를 임의로 확장시키지 않고, MPI가 제공하는 attribute caching을 이용하여 응용 프로그램의 요구조건을 하부의 메커니즘과 연결하는 방법을 통해서 MPICH-G2에 보안 측면의 기능을 강화한 MPICH-GS를 구현한다.

MPICH-GS는 그림 1와 같은 구조를 가진다. 기존 MPICH-G2에 MPI Security 에이전트와 Security 라이브러리가 globus device와 globus i/o 사이에 추가되었다.

MPI 응용 프로그램에서 *MPI_Send()/MPI_Recv()*를 통해 전달되는 메시지는 MPI Security 에이전트를 통해서 사용자가 지정한 알고리즘에 따라 송신 시에 암호화 과정과 수신 시에 복호화 과정을 거쳐 통신 프로세스에 전달 된다.

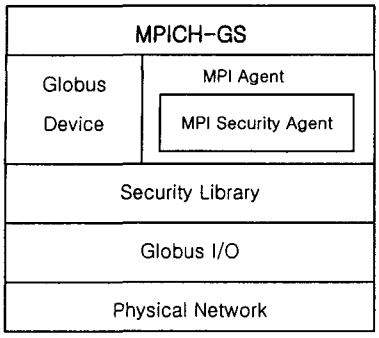

그림 1 MPICH-GS 구조

2.1 MPI Attribute Caching

MPI 표준은 MPI 응용 프로그램의 이식성을 훼손하지 않으면서 응용 프로그램 수준에서의 튜닝을 가능하게 하는 솔루션을 가지고 있다. 이것이 바로 attribute caching이다.

MPI 프로그래밍 모델에서 모든 통신은 하나의 커뮤니케이터 내에서 일어난다. 커뮤니케이터는 단순히 프로세스의 그룹으로, A란 커뮤니케이터에서 보내진 메시지는 B란 커뮤니케이터에 속한 프로세스에서는 받을 수 없다. 커뮤니케이터는 " caching" 메커니즘을 제공하는데, 이것은 한 커뮤니케이터에 새로운 attribute를 연계시킨다. Attribute는 사용자나 라이브러리가 나중의 참조를 위하여 커뮤니케이터에 추가되는 로컬 정보로 정의될 수 있다. Attribute는 key와 attribute 값의 쌍으로 구성되며 key에 의해서 식별이 된다. MPI 응용 프로그램 개발자는 *MPI_Attr_put()*과 *MPI_Attr_get()*을 이용해서 attribute에 대한 값을 새로 설정하거나 기존에 설정된 값을 얻어 올 수 있다. 이는 응용 프로그램 개발자 입장에서의 인터페이스의 변화없이 쉽게 MPI의 확장된 기능을 이용할 수 있도록 한다.

```
struct securecomm_params {
  crypto_algorithm_t    crypto_algm;
  key_size_t            key_size;
  gka_algorithm_type_t gka_algm;
  securecomm_crypto_policy_t crypto_policy;
  securecomm_domain_policy_t domain_policy;
} scp;
MPI_Attr_put(MPI_COMM_WORLD,
        MPICHX_SECURECOMM_PARAMETERS, &scp);
```

그림 2 Security parameter set

그림 2는 MPICH-GS의 Security 기능을 확장하기 위한 parameter의 구조를 보여준다.

2.2 Security 에이전트

MPICH-GS의 핵심은 Security 에이전트이다. MPICH-GS에 Security 기능을 추가하기 위해서는 그 기능을 식별할 수 있는 key를 정의해야 한다. 그림 2에서 보는 것과 같이 Security 에이전트를 구동하기 위한 key는 MPICHX_SECURECOMM_PARAMETERS이고, attribute는 암호화 알고리즘(crypt_algm), 그룹키 동의 알고리즘(gka_algm), 암호화 정책(crypto_policy) 및 도메인 정책(domain_policy) 등을 설정할 수 있다.

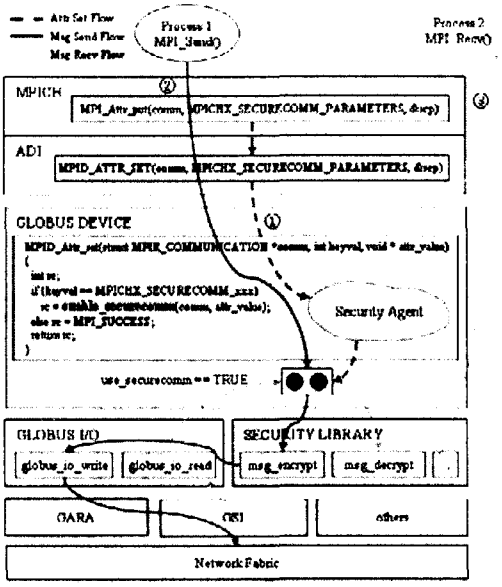

그림 3 Security 에이전트의 동작

그림 3은 Security 에이전트의 동작을 보여준다.

① Attribute 설정 과정

응용 프로그램이 *MPI_Attr_put()*을 통하여 Security를 위한 key와 attribute 구조체에 대한 주소를 넘겨주면, globus device의 *MPID_Attr_set()*에서 key를 식별하여 Security 에이전트를 구동시킨다. 이미 언급하였듯이 MPICH는 계층적인 구조로 이루어져 있고, MPICH의 아래엔 ADI 계층, 그 아래에는 MPICH-G2 구현의 경우는 globus device가 존재한다. *MPI_Attr_put()*은 *MPID_ATTR_SET()*으로 맵핑되고, MPICH-G2에서는 globus device의 *MPID_Attr_set()*이 호출된다. 그림 3에서 보듯 key가 MPICHX_SECURECOMM_PARAMETERS인 경우는 Security 에이전트인 *enable_securecomm()*을 호출한다.

Security 에이전트의 실질적인 역할을 담당하는 *enable_securecomm()*은 사용자가 설정한 attribute 값으로 채널 테이블의 암호화 핸들러를 설정하게 된다. 이때 사용되는 테이블은 MPICH-G2 본래의 채널 테이블이 아닌 MPICH-GS를 위해 확장된 구조체이다. MPICH-G2는 *MPI_Init()* 과정에서 기본적으로 통신에 필요한 정보들을 수집하여 채널 테이블에 유지한다. channel_t의 구조체인 채널 테이블은 여러 구조체에 대한 포인터를 참조함으로써 통신에 필요한 모든 정보들을 유지하고 있

다. 채널 테이블의 구조를 수정하여 암호화 핸들러(sec_handle)를 추가한다(그림 4). 이 핸들러는 그룹키와 암호화 및 복호화 함수 포인터 등과 같은 암호화 통신에 필요한 정보들을 유지하게 된다. 그림 4는 MPICH-GS를 위해 확장된 구조체이다. use_securecomm은 *enable_securecomm()* 이 실행되면 true가 되어, 이후의 통신에서 플래그 역할을 담당하도록 추가되었다.

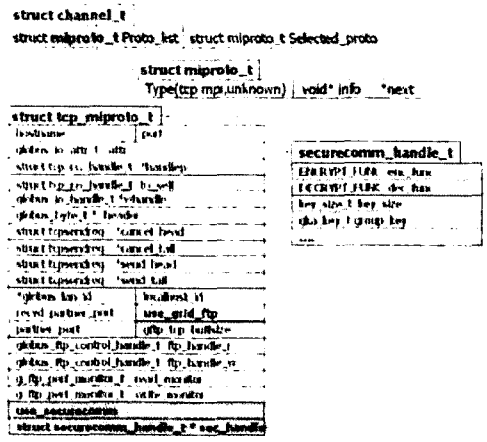

그림 4 channel_t 구조체

② 메시지 보내기 과정

일단, *MPI_Attr_put()*을 통해 attribute가 설정되면, 이후의 *MPI_Send()*는 메시지를 보낼 때마다 이 통신 플래그를 확인하여 true이면, 그림 4의 tcp_miproto_t 구조체의 암호화 핸들러가 지정하는(즉, 사용자가 선택한) 암호화 알고리즘으로 메시지를 암호화 하여 globus-io 라이브러리를 이용하여 물리적인 네트워크로 전송하게 된다.

③ 메시지 받기 과정

암호화된 메시지는 globus-io 라이브러리 함수들에 의해 수신된다. 이렇게 수신된 메시지는 암호화 되어 있는 상태이므로, 암호화 핸들러의 복호화 함수를 가지고 복호화하여 상위의 응용 프로그램으로 올려 보내게 된다.

MPI Security 에이전트의 구현과정을 소개하는 것은 이러한 과정이 MPI의 기능 확장을 위한 모델이 되기 때문이다.

2.3 응용 프로그램 예제

MPICH-GS 상에서 응용 프로그램 개발자는 Security 에이전트의 동작에 대하여 고려할 필요가 없다. 메시지를 암호화하여 통신을 하고자 한다면, 그림 5에서와 같이 응용 프로그램 개발자는 이미 정의되어 있는 Security attribute에 대한 key를 명시하고, 암호화 알고리즘, 그룹키 생성 알고리즘과 암호화 정책, 도메인 정책을 선택하여 *MPI_Attr_put()*을 통해서 attribute를 설정하면 된다. 단순히 이렇게 함으로써 이후의 메시지는 암호화 복호화 과정을 거쳐서 전달된다. 기존의 Security 라이브러리에 새로운 암호화 알고리즘을 추가하는 것은 Security 에이전트의 동작에 대한 고려 없이도 가능하다.

```
#include <mpi.h>
int main(int argc, char **argv)
{   int numprocs, my_id;
    struct securecomm_params  scp;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &my_id);
    if (my_id == 0 || my_id == 1) {
     scp.partner_rank  = (my_id ? 0 : 1);
     scp.crypto_algm = DES_EDE3_CBC; // 암호화 알고리즘
     scp.gka_algm = CLIQUES_GKA; // 그룹키 생성 알고리즘
     scp.key_size = DEFAULT_KEY_SIZE; // 키 크기
     scp.crypto_policy = DEFAULT_crypto_policy; //암호화 정책
     scp.domain_policy = DEFAULT_domain_policy; //도메인 정책
      MPI_Attr_put(MPI_COMM_WORLD,
                  MPICHX_SECURECOMM_PARAMETERS, &scp);
    }
    ...
    MPI_Finalize();
}
```

그림 5 응용 프로그램 예제

3. 결 론

MPICH-GS는 MPI 수준에서의 메시지 암호화를 제공한다. 이를 통해서 통신에 있어서 메시지의 암호화, 통신 라인 상에서의 암호화를 이중으로 구축함으로써, 변화하는 그리드 환경에서의 보안 요구사항을 만족시킬 수 있다. 또한 본 논문에서 제안하는 메커니즘은 MPICH-GS 뿐만 아니라, MPI를 이용한 프로세스 사이에 효율적인 통신을 위한 메시지 압축이나 QoS 기능을 지원하도록 MPICH-G2를 확장하기 위한 참조 모델이 된다. 실제 MPICH-GS 구조를 바탕으로 하여 메시지 압축 및 QoS 기능 등을 구현하기 위한 연구를 진행 중이다.

4. 참고 자료

[1] N. Karonis, B. Toonen, I. Foster. MPICH-G2 : A Grid-enabled implementation of the Message Passing Interface. In the **Journal of Parallel and Distributed Computing, 2003.**

[2] A. Roy, I. Foster, W. Gropp, N. Karonis, V. Sander, and B. Toonen. MPICH-GQ : Quality-of-Service for Message Passing Programs. In the Proceedings of the IEEE/ACM SC2000 Conference. Nov 2000.

[3] I. Foster C. Kesselman, The Globus Project: A status report, In the Proceedings of the Heterogeneous Computing Work-shop, IEEE Computer Society Press, Silver Spring, MD, 1998, pp.4-18.

[4] W. Gropp, E. Lusk, N. Doss, A. Skellum, A high-performance, portable implementation of the MPI message-Passing Interface standard, Parallel Comput. 22(6) (1996) 789-828