가상 합성 전장 환경에서의 효율적인 전송 알고리즘 구현 방안 및 설계†

장의진, 윤미연, 김 산, 신용태
숭실대학교 컴퓨터학과
e-mail : fiatlux@cherry.ssu.ac.kr

Design and Implementation scheme of effective transmission algorithm in virtual battlefield environments

Uijin Jang, Miyoun Yoon, San Kim, Yongtae Shin
Dept. of Computing, Soongsil University

요 약

'분산 모의훈련 전장환경'은 실제 또는 가상전쟁, 전투 등 군사 상황을 묘사하기 위해, 둘 또는 그 이상의 적대 세력간에 발생하는 군사적 상황을 게임규칙, 자료 및 절차를 사용하여 모의하는 군사작전 게임(war game)을 의미한다. 이는 컴퓨터와 네트워크를 이용한 '합성 가상 전장환경'에서 구현되는데, 특성상 다수의 게이머와 게이머, 게이머와 객체, 객체들간에 네트워크를 통한 빈번한 메시지 송·수신을 통한 정보를 교환하고 실시간으로 상호동작 한다. 즉, '합성 가상 전장환경'의 구현을 위해서는 신뢰성과 실시간성을 보장하는 다자간 통신의 보장이 핵심을 이룬다. 그러나, 일반 인터넷상의 브로드캐스팅 기술을 적용할 경우, 가상전장환경에서 폭주하는 통신량을 실시간으로 처리하기엔 어려운 문제점이 있다. 따라서 '합성 가상전장가상환경'의 보다 효과적인 구현을 위해서는 고신뢰 실시간 다중 그룹통신 기술의 개발이 요구된다. 본 논문은 그룹통신 개념에 기반하여 가상 합성 전장환경에 적용할 수 있는 효율적인 전송 서비스를 위한 네 가지 전송 알고리즘을 제시하며, API 설계를 통해 서비스를 구현한다.

1. 서론

'분산 모의훈련 전장환경'은 컴퓨터를 이용한 가상의 전장환경을 제공하여, 군사작전의 분석 및 훈련용으로 사용된다. 환경적인 특성상 빈번한 메시지 송·수신을 고려했을 때, 메시지의 신뢰성과 실시간성을 보장하는 통신이 가장 큰 이슈라고 할 수 있다. 이에 따라서 다자간 통신의 기반 모델인 그룹통신의 전송 방식—원자성, 순서성, 실시간성—을 적용한다면, 메시지의 정확성과 실시간성을 동시에 만족 시킬 수 있을 것으로 기대된다.

본 논문에서는 그룹 통신의 전송 방식[1]을 적용시켜 전장환경에서의 효율적인 메시지 전송 서비스를 제공하기로 한다. 그에 따라 DoD(Department of Defense) 내에서 전장 시뮬레이션을 위해 모델링과 시뮬레이션의 기반 구조를 제공하는 HLA/RTI 와 본 논문의 전송 방식을 비교·분석하여 효율적인 전송 알고리즘을 제시한다.

† 본 연구는 2001 년 국방과학 연구소 기술 개발사업 과제의 지원으로 수행 중임.

2. 관련 연구

2.1 그룹통신의 개념

그룹이란 동일한 그룹 식별자나 멀티캐스트 주소뿐만 아니라, 공통의 어플리케이션 시멘틱스(Semantics)를 공유하는 객체들의 집합체이다. 여기서 객체들이란 프로세스나 호스트 등과 같은 컴퓨팅 자원을 뜻하며, 그룹은 내부 구조를 드러내지 않는 하나의 논리적 개체이다. 그룹통신은 이러한 그룹간의 통신을 네트워크 상으로 지원하는 것이다. 그룹통신은 크게 그룹 멤버쉽 서비스와 전송 서비스로 나눌 수 있다. 그룹 멤버쉽 서비스란, 그룹에 참여하는 객체의 동적인 가입 및 탈퇴를 보장하고 네트워크의 오류 및 객체의 오류로 인한 서비스 중단 및 복구 등을 가능하도록 제공하는 서비스이다. 즉, 동적인 그룹관리를 보장하는 것이다.

전송 서비스란, 개체간의 통신시 신뢰적인 전송 등을 보장하도록 제공하는 서비스이다. 그룹통신에서는 원자성과 순서성을 만족하도록 하여 전송의 신뢰성을 보장하고 있다. 원자성은'all-or-nothing' 의 개념으로서, 그룹에 참여하고 있는 모든 개체는 전송된 데이터를 모두 받거나 아니면, 모두 받지 않음을 의미한다. 즉, 그룹에 전송된 데이터는 그룹내

의 개체 모두가 수신해야 한다는 개념이다. 순서성은 전송시에 보낸 데이터 순서에 의거하여 수신해야 한다는 개념으로, 순서성에는 전순서성, FIFO순서성, 인과적 순서성이 있다. 첫째, 전순서성은 매우 광범위한 개념으로, 그룹내의 모든 수신 멤버가 동일한 순서로 데이터를 수신해야 한다는 것이다. 둘째, FIFO순서성은 데이터를 전송한 순서에 따라 수신해야 한다는 의미이다. 셋째, 인과적 순서성은 이벤트 사건에 기반한 것으로 '1'이라는 데이터 수신 이벤트가 일어난 후에, '2'라는 데이터 수신 이벤트가 일어나야 한다는 것을 정의한다[2].

2.3 HORUS System

ISIS시스템의 후계자격인 호러스 시스템은 융통성과 확장성을 제공하는 프로세스 그룹 통신 시스템이다. 이는 개발하고자 하는 여러 어플리케이션의 필요에 따라, 편리한 인터페이스를 제공한다. 또한, 전송시의 신뢰성을 보장한다. 호러스 시스템은 고성능의 그룹웨어 어플리케이션뿐만 아니라, 실시간 어플리케이션에도 적당하다. 현재 존재하는 호러스 프로토콜의 레이어들은 가상의 동기화 프로세스 그룹의 구현을 포함한다. 이는 연속적이고, 오류를 허용하는 데이터 전송을 할 수 있도록 한다. 또한 병렬적인 멀티미디어 어플리케이션의 구현도 가능하다. 이러한 호러스 시스템은 원자성 및 메시지의 순서성을 보장한다. 현재는 실시간 어플리케이션에 관한 것과 보안적인 측면에 대한 프로토콜 개발이 진행되고 있다. 더욱이, 호러스는 현 시점에서 통신 프로토콜이 어플리케이션의 재시작과 멈춤이 필요 없는 환경에서도 업그레이드 될 수 있도록 한다[3].

2.4 HLA/RTI

HLA(High Level Architecture)는 DoD(Department of Defense)내에서 모델링과 시뮬레이션을 위한 기반구조를 제공한다. 이 구조는 시뮬레이션의 상호운용성과 시뮬레이션 컴포넌트의 재사용을 장려하기 위한 것이다. HLA는 다음의 세가지 개념—(1) Object Model Templates (2) Runtime Infrastructure (3) HLA compliance rules—으로 정의된다.

RTI는 시뮬레이션 시스템에 일반적으로 요구되는 서비스를 제공하기 위한 소프트웨어의 집합으로, 컴퓨팅 플랫폼, 운용체제, 그리고 통신 시스템 상에서 portability 의 방식을 제공한다. 본 논문에서는 DoD에서 사용되는 HLA/RTI의 전송 API를 분석하여 가상전장환경에 보다 효과적으로 접근할 수 있는 새로운 전송 API를 설계한다. [표 1-1]은 HLA/RTI와 본 논문의 전송 측면을 비교한 표이다.

[표 1-1] HLA/RTI와 본 논문의 전송 비교

	HLA/RTI	본 논문
전송	• Using selectively Reliable Transport • support Dual mode 0/1 • Best effort: Mode 0 • Minimum Rate: Mode 0 with heartbeat • State Consistent: Mode 0/1 data stream • Reliable : Provided by TCP	atomicDelivery()
		causalDelivery()
		reliableDelivery()
		realtimeDelivery()
		causalatomicDelivery()

HLA/RTI 는 SRTP(Selective Reliable Transport Protocol)프로토콜이 전송을 담당하며, 이는 듀얼모드(Dual Mode) 전송을 제공한다. '합성 가상 전장환경' 'mode 0'에서는 'best effort' 서비스를 제공하며, 신뢰적인 전송의 경우는 'mode 1'로서 TCP 프로토콜을 이용하여 전송한다.

본 논문에서는 각각의 데이터 특성에 맞는 알고리즘을 API 화하여 제공함으로써 기본적인 네트워크 레벨에서의 신뢰성에 더하여 어플리케이션적인 신뢰성을 제공한다. 또한, 실시간성을 제공함으로써 신속성을 요하는 데이터의 처리를 가능하게 한다. 즉, 가상전장환경의 가장 큰 이슈인 신뢰성과 실시간성을 보장하는 통신에 적합한 전송 API 를 설계하도록 한다[4].

3. 전송 알고리즘의 설계[5]

3.1 Causal delivery

인과적 순서성을 보장하기 위해서, 전달될 메시지에 선행하는 모든 메시지가 수신되었을 경우에만 그 메시지를 전달하고 그렇지 않으면, 선행 메시지의 수신 완료 시까지 메시지의 전달을 지연한다.

Sender

```
Step 1: Sender Mi (1≤i≤n) initializes message
check vector MCi
Step 2: Increase 1 to MCi[i] and Include MCi[i] in
a message
Step 3: Mi sends a message and waits ACKs for 2T
Step 4:

If (Mi received ACK from Mj) then
{
   Check MCi[j];
   If (Mi received all ACKs from receivers)
      Send OK
   Else
   {
      Retransmission to MjNACK
          Wait ACK for T
   }
}
Else
{
   Retransmission to MjNACK
   Wait ACK for T
}
```

Receiver

```
Step 1 : Receiver Mj(∀j∈{1,2,…,n}-{i})
initializes message check vector MCj
Step 2 : Mj receives a message and waits for T
Step 3:
    If (MCj[i] = MCi[i]-1)
    {
        If MCj[k] ≥ MCi[k], ∀k∈{1,2,…,n}-{i}
            Transmission end
        Else
        {
            Request the lost packet to retransmit
            Mj processes retransmitted and stored
            messages in a queue
            Update MCj
        }
    }
    Else
    {
        Request the lost packet to retransmit
        Mj processes retransmitted and stored
        messages in a queue
        Update MCj
    }
```

3.2 Atomic delivery

수신 멤버들은 선행 메시지 벡터를 이용하여 분실된 메시지를 탐지하고 재전송을 요청함으로써, 동작 중인 모든 멤버들은 메시지를 수신하게 된다.

Receiver

```
Step 1: Wait a message from Mi
Step 2:
If (Receiver Mj received a message)
    Store a message at queue and send ACK to Mi
    If (Received OK)
        Processing a message
    Else
    {
        If (timer expired)
            Processing a message
        Else
            Wait a message from Mi
    }
Else
{
     If (timer expired)
        Process a message
     Else
        Wait a message from Mi
}
```

3.3 CausalAtomic delivery

모든 멤버에 의한 메시지의 수신을 보장하는 반면 메시지의 수신 시점을 알 수 없다는 원자성의 단점을 보완하기 위하여, 메시지의 수신 시점에 따른 인과적 순서성을 보장한다.

Receiver

```
Step 1: Receiver Mj initializes message check
vector MCj
Step 2: Mj receives a message and waits for T
Step 3:
If (MCj[i] = MCi[i] -1)
{
   If (MCj[k] ≥ MCi[k])
   {
      If(MCj[i] = msgNum-1)
         Transmission end
      Else
      {
         Request the lost packet to retransmit
         Mj processes retransmitted and stored
         messages in a queue
         Update MCj
      }
    }
    Else
    {
         Request the lost packet to retransmit
         Mj processes retransmitted and stored
         messages in a queue
         Update MCj
    }
}
Else
{
     Request the lost packet to retransmit
     Mj processes retransmitted and stored messages
     in a queue
     Update MCj
}
```

3.3 Realtime delivery[6]

실시간성을 보장하도록 큐의 스케줄링을 조정하도록 하며, 손상된 데이터의 경우 FEC(Forward Error Check)기법을 사용함으로써 신뢰성을 보장할 수 있다.

Receiver

```
Step 1: Add redundancy information for
data repair
Step 2: Sender Mi sends a message and
start Δ timer
Step 3:
If (Any members received NACK)
    Retransmit a message
Else
    Transmission end
```

4. 분석 및 향후 연구방향

본 논문에서는 가상 합성 전장환경에서 다자간 통신의 기반 모델인 그룹통신의 전송 방식을 적용시켜 효율적으로 데이터를 전송하기 위한 전송 알고리즘을 제안했다. 다양한 전송 서비스를 제공하기 위하여 원자성, 순서성 및 실시간성 알고리즘을 제안하였으며, 제안한 알고리즘은 메시지 분실 및 프로세스 오류시에도 신뢰성 있는 서비스를 보장한다. 본 논문과 HLA/RTI는 유니캐스트 전송을 이용한 응용 프로그램으로써 멀티캐스트를 제공한다는 공통점이 있다. 이는 현재, 실제적인 멀티캐스트 통신이 어렵기 때문이다. 그러나, 본 논문은 각각의 데이터 특성에 맞는 알고리즘을 API화하여 제공함으로써 기본적인 네트워크 레벨에서의 신뢰성에 더하여 어플리케이션적인 신뢰성을 제공한다. 또한, 실시간성을 제공함으로써 신속성을 요하는 데이터의 처리를 가능하게 한다. 향후에는 개발된 API를 가상 합성 전장환경을 기반으로 한 프로토타입 모델에 적용하여 전송 서비스를 테스트 하게 될 것이다. 마지막으로, 기존에 가상 합성 전장환경에서 사용된 HLA/RTI 기반의 전송 서비스와 비교 테스트를 거친 후, 본 논문의 전송 알고리즘의 효율성과 성능 향상의 효과에 대한 분석이 요구된다.

참고문헌

[1] L. E. Moser, P. M. Melliar-Smith, D. A, Agarwal, P. K. Budhia, and C. A. Lingley-Papadopoulos, "Totem : A Fault-Tolerant Multicast Group Communication System," *Communications of ACM*, Vol. 36, No. 12, Dec. 1993, pp.54-63.

[2] M. Hofmann, "Enabling Group Communication in Global Networks", *Proceedings of Global Networking 97*, Canada, June 1997.

[3] Robbert van Renesse, Kenneth P. Birman, and Silvano Meffeis, "Horus : A Flexible Group Communication System," *Communications of the ACM*, Vol.39, No.4, Apr. 1996, pp.76-83.

[4] http://www.dmso.mil/public/

[5] Sape mullender, "Distributed Systems", ACM Press New York

[6] H. Schuzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: a transport protocol for real-time applications," *Internet` RFC 1889*, January 1996.