가상 전장 환경에서의 효율적인 네트워크 트래픽 처리를 위한 액티브 네트워크 응용방안

Applied Research of Active Network to Control Network Traffic in Virtual Battlefield Environments

정창모
한남대학교 컴퓨터공학과
이원구
한남대학교 컴퓨터공학과
김성옥
한남대학교 컴퓨터공학과
이재광
한남대학교 컴퓨터공학과

중심어 : 가상전장환경, 워게임, 시뮬레이션, HLA

요 약

군사분야에서 컴퓨터 시뮬레이션의 활용은 이미 수십 년 전부터 이루어지고 있다. 컴퓨터 시뮬레이션을 통해서 실제 전투자산을 가동하지 않고 실전과 같은 전투경험을 부여하고 있다. 이러한 시뮬레이션이 실제와 똑같은 환경을 구축하기 위해서는 페더레이트(federate)간의 연동(federation)이 네트워크 상에서 잘 수행되어야 한다. 이에 본 논문에서는 전장 데이터(이하 액티브 패킷)의 신속한 전달을 필요로 하는 긴급한 실제상황과 유사한 전장공간을 구축할 수 있도록 액티브 네트워크 상에서의 동적기술(혹은 액티브 네트워크 기술)을 이용해 페더레이트(혹은 액티브 노드) 간의 효율적인 트래픽 처리가 가능한 가상 전장 환경을 구성하고, 이에 대한 유효성을 모의실험을 통하여 검증하였다.

Chang-Mo Jung (jjchmo@yahoo.co.kr)
Dept. of Computer Engineering, Hannam University
Won-Goo Lee (wglee@netwk.hannam.ac.kr)
Dept. of Computer Engineering, Hannam University
Sung-Ok Kim (cho5kim11@hotmail.com)
Dept. of Computer Engineering, Hannam University
Jae-Kwang Lee (jklee@netwk.hannam.ac.kr)
Dept. of Computer Engineering, Hannam University

Keyword : Virtual Combat, War Game, Simulation, HLA

Abstract

Computer simulation has used to a area of military training from about several years ago. War game model(or computer simulation) endow a military man with field training such as combat experience without operating combat strength or capabilities. To samely construct simulation environment against actual combat environment associate among federates on network. we construct virtual combat environment enabling to efficiently manage network traffic among federates(or active nodes) by using active network technique on virtual military training space such as urgent combat field needed to rapidly transfer combat information including image and video, verify its validity by the help of simulation

I. 서론

컴퓨터 시뮬레이션(혹은 가상 전장환경)처럼 미래전은 사람이 직접 전쟁터에서 전쟁을 하는 것이 아니라 첨단 과학기술로 만들어진 다양한 무기체계를 활용하여 상대의 정보를 입수하고, 목적과 용도에 따라 상대의 취약한 곳을 공격하게 될 것이다.

이는 최근 이라크전, 코소보전과 아프가니스탄전에서 무인 정찰기를 통해 목표물에 대한 정찰을 보면 더 쉽게 이해될 것이다. 이러한 컴퓨터 시뮬레이션의 활용은 이미 수십년 전부터 이루어지고 있다. 컴퓨터 시뮬레이션을 통해서 실제 전투자산을 가동하지 않고 실전과 같은 전투경험을 부여하고 있다. 따라서, 가상 전장환경을 실제 전장환경과 얼마나 유사하게 구성하느냐가 미래전을 수행하는데 가장 중요한 요소가 될 것이며, 이러한 컴퓨터 시뮬레이션은 첨단 무기체계의 복잡화,

* 본 연구는 한국과학재단 목적기초연구(R0a-2002-000-00127-0)지원으로 수행되었음.
접수번호 : #030904-001
접수일자 : 2003년 9월 4일, 심사완료일 : 2003년 9월 17일
*교신저자 : 이원구, e-mail : wglee@netwk.hannam.ac.kr

훈련장 및 국방예산 확보의 어려움 등 여러 가지 제한된 여건을 해결할 수 있는 대안으로 그 필요성이 점점 증대되고 있다.

가상 모의훈련 전장환경은 컴퓨터를 이용한 가상의 전장환경을 제공하여, 군사작전의 분석 및 훈련용으로 사용된다. 하지만 환경적인 특성상 다량의 전장 데이터(이하 액티브 패킷)의 송수신을 고려했을 때, 종단장치(시뮬레이션 서버와 IGI 호스트)간의 네트워크 트래픽 부하가 현저하게 증가하여 중간 노드(네트워크 서버, IGI 서버)들의 트래픽 처리율이 현저하게 저하될 수 있고, 이는 실제 전장환경과 유사한 환경을 구축하고 모의훈련하고자 하는 가상 모의훈련 전장환경의 구성에 대한 실패를 초래할 수도 있다. 이러한 문제점 들을 해결하고자 하는 연구가 아직은 미비하지만, 향후 다각적인 측면에서 연구될 것이다.

본 연구에서는 액티브 네트워크를 활용한 효율적인 멀티캐스트 기법, 액티브 패킷에 대한 동적 라우팅 기법, 액티브 노드(네트워크 서버내지는 IGI 서버) 상에서의 혼잡제어 기법, 액티브 호스트(혹은 IGI 호스트)에 대한 자원할당 기법 및 전체적인 액티브 네트워크 도메인(혹은 페더레이션) 구성에 대해 연구하고, 이를 기반으로 앞서 기술한 문제점들을 해결하며, 이러한 기법을 적용한 액티브 네트워크 환경에 가상 전장환경을 통합하여 모델링한 후, 모의실험을 통하여 그 성능을 평가하였다.

본 논문의 구성은 다음과 같다. 2장에서는 지금까지 알려진 분산 시뮬레이션 및 액티브 네트워크 구조에 대해 살펴보고, 3장에서는 모델링을 수행할 대상 네트워크(혹은 가상 전장환경)의 성능향상을 위한 다양한 액티브 네트워크 기법에 대해서 설명할 것이다. 4장에서는 새로이 제시한 액티브 노드 및 네트워크에 대한 모델링을 하며, 5장에서는 이를 적용한 가상 전장환경 모델링의 효율성 및 유효성을 모의실험을 통해 검증한다. 마지막으로 6장에서는 결론을 맺고 향후 연구방향에 대해 논의한다.

II. 관련연구

1. 분산 시뮬레이션 구조

분산 시뮬레이션의 어려움에도 불구하고 네트워크 기술을 통한 분산 시뮬레이션 기술은 끊임없이 발전해 왔다. 네트워크 기술을 이용한 분산 시뮬레이션은 개별 무기체계 수준의 장비조작 및 전투기술 훈련에 사용해 오던 각종 시뮬레이션들을 하나의 컴퓨터 네트워크를 통해 연결함으로써 동일한 전장공간에서 시뮬레이터들을 운용할 수 있도록 하기 위한 시도로 출발했다[1]. 그러나 기존의 시뮬레이션 모형들은 대부분 각종 모의기능을 단일 모형에 통합하여 묘사하는 중앙집중식 모의구조를 갖는다. 이러한 중앙집중식 모의구조를 갖는 단일 모형으로는 보다 복잡하고 다양한 현대전을 실전과 유사한 훈련환경으로 조성하는 것이 불가능하게 되었다. 따라서 다양한 유형의 시뮬레이션 모델들을 상호연동하고 통합하는 방법을 통해 보다 복잡한 새로운 목적을 달성할 수 있도록 하기 위한 방안들이 요구되었다. 이러한 시도의 하나로 미 국방성에서는 1995년부터 표준연동구조(HLA : High Level Architecture)를 개발하게 되었고, 2000년에는 IEEE1516 국제표준으로 등록되었다[2].

1.1. HLA(High Level Architecture)

과거의 분산 시뮬레이션 구조는 많은 발전에도 불구하고 여전히 모형간 또는 체계간 연동소요는 단위모델의 작은 변경에도 불구하고 복잡한 연동화 작업이 발생하는 문제점을 내포하고 있었다[3].

차세대 시뮬레이션 기술 구조인 HLA는 DIS를 이은 차세대 네트워크 시뮬레이션의 표준 구조로써 여러 종류의 다양한 시뮬레이션 컴포넌트들로 이루어진 복합 시뮬레이션을 지원하기 위해 제정된 표준 프레임워크(framework)이다. 즉, HLA는 모든 유형의 M&S에 대해 상호 운용성(interoperability)과 재사용성(reusability)을 보장하기 위한 공통 시뮬레이션 구조를 제공한다[4].

또한, HLA 분산 시뮬레이션 구조는 RTI라고 하는 브로커 체계를 기반으로 함에 따라 연합체 구성 모델간의 연동구조 복잡도를 현저히 감소시켰다[3].

1.2. RTI(Run_Time Infrastructure)

RTI는 분산 운용 체계가 어플리케이션에 서비스를 제공하는 것과 유사한 방법으로 페더레이션을 구성하고 있는 페더레이트들에게 상호 연동에 필요한 서비스를 제공한다. 그림 1은 페더레이션 1과 페더레이션 2라는 두 개의 페더레이션이 연동되는 형태로서 하나의 RTI에 여러 개의 페더레이트들로 구성된 다수의 페더레이션이 연동되는 상황을 보여주는 HLA 시스템의 예이다[4].

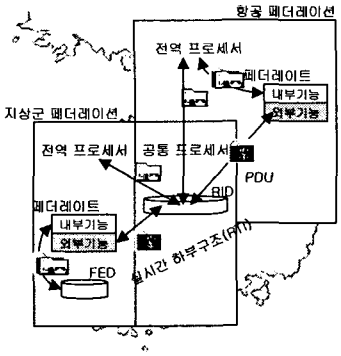

그림 1. HLA를 사용하는 시스템 응용

2. 액티브 네트워크 구조

2.1. 액티브 네트워크(Active Network)

액티브 네트워크는 노드에서 패킷에 포함된 프로그램의 실행이 가능하도록 한 네트워크이다. 따라서 기존 네트워크에서 노드는 단순히 데이터 전송을 목적으로 패킷의 헤더 정보만을 처리한데 비해, 액티브 네트워크에서는 사용자가 노드 기능을 특성화시킬 수 있으므로, 사용자 중심의 네트워크 구성이 가능하게 된다. 이에 따라 액티브 네트워크는 표준화 작업없이 빠른 속도로 새로운 서비스들이 전개되도록 한다.

아래 그림 2는 액티브 보안을 포함한 액티브 네트워크의 구성으로서 대응노드(ARN: Active Response Node)와 실행노드(AEN: Active Execution Node)를 기본 단위로 하는 기본 도메인(elementary domain)과 기본 도메인간 상호 연동된 ASM(Active Security Management) 도메인, 그리고 도메인을 관리하기 위한 중앙관제시스템(CCS: Central Coordinator System)으로 구성된 액티브 네트워크의 예이다.

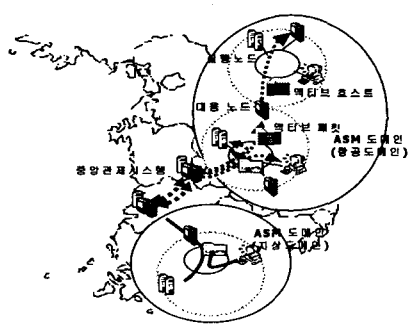

그림 2. 액티브 네트워크 응용

이전의 그림 1에서 보여주었던 분산 시뮬레이션 모델과 그림 2의 액티브 네트워크 모델 간에는 유사한 형태의 구조와 관리라는 측면을 갖고 있지만, HLA 모델에서의 문제점을 액티브 네트워크 모델을 통해 극복할 수 있다는 측면 또한 내포하고 있다. 이에 우리는 두 모형간의 통합방안과 성능향상 방안에 대하여 제시하고, 모의실험을 통해 검증하고자 한다.

2.2 액티브 라우팅(Actvie Routing)

현재 많은 연구소와 대학에서 액티브 네트워크 기술을 연구하고 있지만, 주로 액티브 노드 구조나 액티브 응용에 관한 연구를 하고 있으며, 액티브 네트워크 라우팅 프로토콜에 관한 연구는 미흡한 수준이다. 현재까지 제안된 액티브 노드들은 액티브 네트워크 토폴로지가 구성되어 있다고 가정하거나 혹은 정적으로 구성하는 방법을 사용한다. 정적인 구성 방법은 구현은 용이하지만 관리자의 많은 노력을 수반하며 특히 네트워크의 동적인 상황을 반영하지 못한다. 또한 액티브 네트워크 상의 특정 액티브 노드을 이용해 세션을 설정하는 액티브 응용[5]의 경우는 정확한 네트워크 토폴로지를 제공해 주어야 하며, 따라서 동적인 액티브 네트워크 라우팅 프로토콜이 요구된다.

현재까지 액티브 네트워크상에서 라우팅 프로토콜에 대한 연구로는 SLRP(Service Layer Routing Protocol)[6]와 TCOM 500 PLAN(Programming Language for Acive Networks)[7] 프로젝트 등이 있다. 이러한 프로토콜 모두 동적인 토폴로지 구성과 라우팅을 제고하기 위해 제안되었다. 그렇지만, 두 프로토콜 모두 정적인 토폴로지가 구성되어 있는 상태(즉, 이웃한 액티브 노드에 대한 정보가 정적으로 설정되어 있는 상태)를 기반으로 동작하기 때문에 완전하게 동적인 프로토콜이라고 볼 수가 없으며, 가상 전장공간이라는 특수한 환경을 고려할 때 동적 라우팅 방식은 반드시 요구된다[8].

III. 네트워크 트래픽 성능 향상 방안

현재 많은 연구소와 대학에서 액티브 네트워크 기술을 연구하고 있지만, 액티브 네트워크 상에서의 트래픽 제어, 액티브 라우터간의 동적 경로설정, 액티브 패킷에 대한 자원할당 및 효율적인 코드 캐싱과 같은 네트워크 트래픽 성능 향상에 관한 연구는 미흡한 수준에 있으며 반드시 해결해야 할 사항이다[8].

그림 3에서는 우리가 가상 전장공간 상에서 액티브 네트워크를 모델링하기 위한 우선적인 고려사항에 대해 보여주고 있다. 첫째, 네트워크 상의 모든 노드가 액티브 노드가

아니기 때문에 액티브 패킷을 다음 노드로 전달할 때 직접 다음 액티브 노드로 전달할 수 있느냐는 것. 둘째, 일반 IP 네트워크와 함께 구축되어 있는 액티브 네트워크 상에서 패킷을 어떠한 경로를 통하여 빠르게 전달시키며, 트래픽 제어를 어떻게 하느냐는 것. 셋째, 액티브 노드로 들어온 패킷에 대한 자원할당을 어떻게 효율적으로 할 수 있느냐는 것. 넷째, IGI 호스트로부터의 요청에 대한 응답으로 액티브 노드에서 멀티캐스트를 어떻게 효율적으로 처리할 수 있느냐는 것이다.

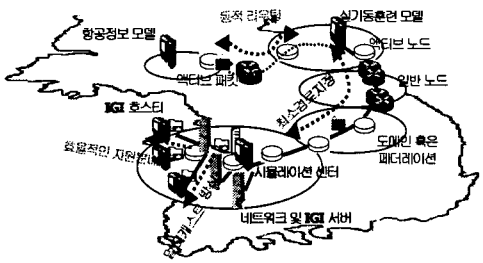

그림 3. 가상 전장환경 상에서의 액티브 네트워크 구성

본 연구에서는 최근에 제안된 다양한 기법이 어떻게 앞서 기술한 고려사항을 해결하고 있는지를 살펴보고, 이러한 기법을 가상 전장환경에 적용한 후, 모의실험을 통해 그 유효성을 검증하여 실제와 유사한 전장공간을 구성하고자 한다.

1. 네트워크 상의 트래픽 제어

기존의 네트워크의 내부 라우터(core router)들이 주로 패킷 전송과 경로할당을 처리하는 반면 액티브 네트워크에서는 패킷 내에 프로그램이나 처리에 필요한 정보를 담아 전송하게 되는데 네트워크 상에 있는 노드는 액티브 패킷을 읽어 들여 각 라우터의 실행환경(execution environment)에서 실행할 수 있고 필요한 프로그램(혹은 코드)을 다른 곳으로부터 다운로드하여 실행 가능하다. 또한 네트워크 내부 노드들은 서로에게 필요한 네트워크 상의 정보를 공유할 수 있다. 이 기술을 사용할 경우 가상 전장환경 상에서 혼잡이 발생하면 네트워크 내부에서 이를 감지하여 주변 라우터에 혼잡에 대한 정보를 알려줄 수 있으며 직접적으로 혼잡을 제어할 수 있다[9]. 액티브 네트워크 상에서 패킷이 액티브 노드(Active Node)에 수신되었을 경우 액티브 노드 상에 있는 혼잡정보(Congestion Information)와 액티브 패킷에 있는 혼잡 정보를 이용하여 패킷에 대한 처리를 수행하기 때문에 액티브 노드를 이용한 혼잡제어는 액티브 노드의 위치와 노드 간의 통신이 중요하다고 할 수 있다[10]. 하지만, 본 연구에서는 액티브 노드의 위치에 관해서는 고려하지 않는다.

만일 가상 전장환경 상에서 혼잡이 발생할 경우 액티브 노드의 패킷 처리방식에 대한 선택문제는 네트워크의 성능에 큰 영향을 끼칠 수 있으며, 이는 모의훈련용 시뮬레이션인 워게임 모델에도 커다란 영향을 미칠 수 있다.

1.1. 기존 혼잡제어기법

전장환경에서 혼잡이 발생했을 경우 라우터가 혼잡 발생시 트래픽에 대한 정보를 저장하고, 주변 라우터에 혼잡발생에 대한 정보를 전송하여 혼잡에 대한 처리를 해줄 수 있다면 전송자가 즉각적인 처리를 하는 것과 같은 효과를 얻을 수 있을 것이다. 내부 라우터는 자신과 연결된 라우터들에 대한 정보를 이미 갖고 있기 때문에 혼잡에 대한 정보를 패킷에 담아서 전달하는 것은 많은 처리가 필요치 않다. 이미 사용 가능한 정보와 혼잡에 대한 정보를 내부 라우터와 주변 라우터가 공유하기 때문에 이들 라우터들 사이의 통신이 일정한 규칙을 갖고 통신을 하게 된다면 혼잡제어에 있어 보다 빠르고 효율적인 처리가 가능할 것이다.

아래 그림 4에서 알 수 있듯이 창조21 모델상의 노드가 혼잡이 발생하는 내부 라우터로 설정되어 있고, 주변 라우터들은 혼잡 라우터로부터 혼잡정보를 받아 혼잡 발생시 트래픽을 필터링하게 된다. 이상적인 경우 HLA 연동모델에 의해 창조 21 모델과 연동되어 있는 항공정보 모델의 시뮬레이터(전송자)에서 창조 21 모델의 IGI 호스트(최종 목적지 혹은 수신자)까지의 RTT(Round Trip Time)가 100ms이고 항공 시뮬레이터에서 주변 라우터까지의 지연시간이 10ms라고 하면 혼잡 발생시에 기존의 단대단(end-to-end) 처리방식보다 80ms정도 빠르게 처리가 가능할 수도 있다.

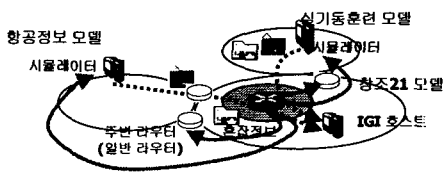

그림 4. 혼잡제어기법에서 사용한 라우터간의 통신

최초 혼잡 라우터에서 주변 라우터로 혼잡정보를 전송할 때를 제외한다면 주변 라우터는 그 이후에 들어오는 트래픽을 20ms의 지연시간으로 처리하는 것이 가능하다. 물론,

실제 네트워크 상에서 액티브 노드(일반적인 경우 라우터에 해당)의 설치에 있어, 어떤 노드가 액티브 노드로 동작하고 어떤 노드가 일반 노드로 동작할 것인가에 대한 문제점을 갖고 있다. 이상적인 설치에서는 혼잡을 처리할 때까지 걸리는 제어시간을 줄일 수 있지만 실제 네트워크 상에서 노드 모두를 액티브 노드로 설정할 수는 없는 것이다. 모든 액티브 노드에서 피드백된 정보를 전송자가 모두 수신하고 분석하여 처리하는 것은 비효율적이기 때문이다. 기존의 워게임 모델에서는 창조21 모델의 시뮬레이션 센터에서 모든 요청에 처리를 수행한다. 혼잡제어기법은 비록 이러한 액티브 네트워크 설치상의 문제점은 고려하지 않았지만 혼잡제어에 있어 새로운 방향을 제시했다는데 의의가 있다.

1.2 혼잡제어기법 확장

혼잡제어기법에서는 내부 라우터(액티브 노드)에서 상태정보를 저장해야 하며 혼잡이 발생했을 경우 혼잡에 필요한 처리도 해야 한다. 최근에 제안된 기법(Enhanced ACC, 이하 혼잡제어기법 확장)에서는 그림 5에서와 같이 내부 라우터에서 필요한 상태정보를 주변 라우터로 옮겨 처리하고 피드백된 정보를 각 전송자로 전달하게 된다. 주변 라우터는 내부 라우터에서 피드백된 패킷을 수신할 경우 자신에게 연결된 전송자에게 패킷을 복제하여 전송하게 된다. 전송자가 보내는 정보는 패킷 전송 시간과 내부에서 혼잡이 발생할 때 패킷 처리 프로그램이 전송된다. 그림 4와 그림 5의 차이점은 주변 라우터에 전송된 내부 라우터 정보가 특정 전송자에만 전송되는 것이 아니고 각 전송자에게 전송되도록 한다는 것이다. 또한 전송자에서 온 프로그램(새로운 어플리케이션 포함)이나 전장정보를 처리하여 내부 라우터에 전송하도록 한다. 이 프로그램과 전송시간은 CCE(Congestion Control Engine)에서 처리된다.

주변 라우터는 TCP 패킷이 수신되면 전송자의 개수를 지정하게 된다. 주변 라우터에서의 상태정보는 내부에서 상태정보를 유지하는 것에 비교하면 상대적으로 부담을 덜어주게 된다. 각 주변 라우터에서 수집된 정보는 내부 라우터에서 처리하게 되며 주변 라우터에서 보낸 정보에 따라서 처리 방법은 달라진다. 그 밖에 주변 라우터는 UDP 패킷에 대해서 대기 리스트(Zombie list)를 두어 수신되는 패킷을 처리한다. UDP 패킷은 지속적이며 일련적(bursty)인 특성을 가지므로 혼잡이 발생할 경우 내부 라우터에 부담을 주는 요인이며 네트워크 자원을 낭비하는 요인이 된다. 이와 같은 비반응(Unresponsive) 트래픽이나 미전송(Undelivered) 패킷을 주변 라우터에서 처리함으로써 전장환경 상에서 전장정보 및 네트워크 자원을 효과적이고 빠르게 관리할 수 있다[10].

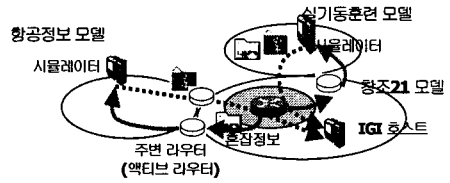

그림 5. 혼잡제어기법 확장에서 사용한 라우터간의 통신

2. 액티브 노드간의 동적경로 설정

현재 워게임 모델의 발전 과정으로 미루어 볼 때 기술적, 상업적인 이유로 가상 모의훈련 전장환경 상의 모든 노드가 액티브 노드로서 동작할 수 없고, 가상 전장공간의 모든 네트워크가 액티브 네트워크로 구축될 수 없다. 따라서 하나의 어플리케이션 작업 및 액티브 패킷 전송을 수행하기 위한 여러 개의 액티브 패킷들이 가상 전장공간상의 같은 액티브 노드를 경유한다는 것은 매우 중요한 문제이다.

즉, 송신지에서 목적지까지의 라우팅이 이루어진다면 액티브 패킷의 경로는 패킷마다 서로 다른 경로를 통해 전달될 가능성이 생긴다. 이 상황은 다음과 같은 액티브 어플리케이션에 있어서 심각한 문제를 유발한다. 액티브 네트워크 토폴로지 정보를 기반으로 몇몇 특정 액티브 노드를 세션에 포함하는 액티브 어플리케이션의 경우, 반드시 같은 세션에 속한 액티브 패킷은 해당 액티브 노드를 순서대로 거쳐가야 한다. 혹은 이전 액티브 패킷의 결과가 다음 액티브 패킷의 처리과정에 영향을 주는 액티브 어플리케이션의 경우도 이전 액티브 패킷의 경로를 통해 전달되어야 한다. 또한 같은 세션에 속한 액티브 패킷이 서로 다른 액티브 노드를 거쳐 간다면, 각 액티브 노드마다 액티브 코드 요청 과정을 수반하게 되는 상황이 발생하게 되는 것이다[8].

따라서 송신 노드가 중간의 액티브 노드 위치에 대하여 모르더라도 액티브 패킷이 모두 같은 노드를 거치도록 하는 정적 및 동적 경로설정 기법의 적용을 통해 전송지연을 최소화하였다.

2.1. 초기 경로 설정

송신자는 액티브 패킷을 보내기 전에 먼저 다음 액티브 노드 주소를 찾기 위한 경로 요청 패킷(Req)을 보낸다. 경로 요청 패킷은 액티브 패킷으로서 IP 라우터 경보 옵션(Router

Alert Option)을 포함하고, 전송해야 할 액티브 패킷들의 송신지 주소와 최종 목적지 주소, 초기 일련번호, 전송할 데이터의 양, 플로우 식별자, 패킷 일련번호를 포함한다. IP의 라우터 경보 옵션은 목적지가 아닌 중간의 액티브 노드에서 실행 가능하도록 해준다. 중간의 액티브 노드(AN(1))가 경로 요청 패킷을 받으면 경로 요청 패킷을 보낸 바로 전 단계의 액티브 노드로 자신의 주소를 실은 경로 응답 패킷(Rep(1))을 보낸다. 경로 응답 패킷은 다음 액티브 노드의 주소(AN(2)의 addr), 플로우 식별자, 패킷 일련 번호를 포함해야 한다. 동시에 이 노드는 경로 요청 패킷을 다음 액티브 노드를 찾기 위하여 최종 목적지를 향하여 전송한다. 경로 요청 패킷을 보낸 액티브 노드가 경로 요청 패킷을 받으면 이 패킷에 담긴 다음 액티브 노드의 주소로 액티브 패킷들을 전송하기 시작한다. 액티브 패킷을 받은 액티브 노드는 이미 다음 액티브 노드에 대한 경로 응답 패킷이 도착했다면 다음 액티브 노드를 향해 액티브 패킷을 계속 전송하고, 만약 경로 응답 패킷이 아직 도착하지 않았다면 경로 응답 패킷이 도착할 때까지 기다려야 한다. 위와 같은 과정을 액티브 패킷이 최종 목적지에 도착할 때 까지 반복한다. 그림 6은 위의 과정을 그림으로 나타낸 것이다.

이 방식은 경로 요청 패킷이 다음 액티브 노드에 도착하고 경로 응답 패킷이 돌아올 때까지 기다려야 하는 오버헤드가 있다. 그러나 네트워크가 커지고 전송량이 늘어남에 따라 초기 경로 설정을 위한 지연이 차지하는 비율은 작아진다. 중간의 액티브 라우터에서 소요되는 지연을 줄이기 위하여 복사 후 전송(Copy-and-Forward) 이라는 간단한 액티브 패킷 전송방식을 기술한다.

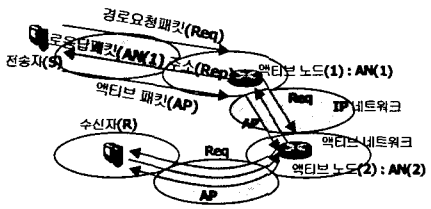

그림 6. 액티브 패킷의 경로 설정 과정

2.2 정적 패킷 라우팅 방식

최종 목적지(가상 전장공간상에서 IGI 호스트)까지의 중간 액티브 노드(가상 전장공간상에서 시뮬레이션 서버와 네트워크 서버, 네트워크 서버와 IGI 서버 혹은 시뮬레이션 서버와 다른 시뮬레이션 서버사이의 액티브 노드)는 IP 패킷이 들어오면 특별한 처리없이 단순히 다음 노드로 전달한다. 그러나 처리할 수 있는 액티브 패킷이 들어오면 패킷의 내용을 실행한 후 다음 노드로 전송한다. 따라서 중간 액티브 노드에서 지연이 발생한다. 이 지연은 중간의 액티브 노드의 개수가 늘어나고 전송하는 액티브 패킷의 양이 증가할수록 더욱 커지게 된다. 복사 후 전송은 액티브 패킷이 모두 들어올 때까지 기다리지 않고 들어오는 대로 사본은 남기고 원본은 다음 노드로 전송시키는 방식이다. 이 때 다음 노드로 전송하기 위해서는 경로 응답 패킷이 도착해 있어야 한다. 만약 경로 응답 패킷이 아직 도착하지 않았다면, 경로 응답 패킷이 도착할 때까지 기다린 후 나머지 액티브 패킷이 모두 도착하지 않았더라도 저장해 놓은 액티브 패킷들을 전송하기 시작한다[13]. 그러나 이러한 기법은 기존 연구(OSPF, SLRP 등등)에서와 같이 네트워크 토폴로지가 정적으로 설정되어 있음을 가정하였다. 하지만 실제 우리가 적용하고자 하는 가상 전장환경은 이웃한 노드에 대한 정보가 없는 상태에서부터 시작할 수도 있으며, 이와 같은 목적을 위해 설계한 액티브 네트워크 라우팅 프로토콜(Active OSPF, 이하 OSPF 확장 프로토콜)이며, 우리는 이 동적 패킷 라우팅 방식을 적용하였다[8].

2.3. 동적 패킷 라우팅 방식

액티브 패킷의 라우팅은 액티브 네트워크 토폴로지 데이터베이스와 액티브 라우팅 테이블을 통해 이루어진다. 그리고 가상 전장환경상의 모든 노드가 액티브 노드가 아니기 때문에 액티브 노드간의 데이터 전송은 터널링(tunneling)을 통해 이루어져야 하며, 앞서 정적인 터널링 전송기법에 대해서는 설명하였다.

동적 패킷 라우팅 방식을 적용하려는 액티브 네트워크상의 액티브 노드의 경우 직접적으로 액티브 네트워크 토폴로지의 구성에 참여하며, 라우터로서의 역할과 송·수신자로서의 역할을 같이 하기 때문에 더 복잡한 처리과정을 수행한다.

그림 7은 송신지와 목적지가 모두 액티브 호스트인 경우에 있어서, 각각 액티브 패킷이 전달되는 과정을 보여준다.

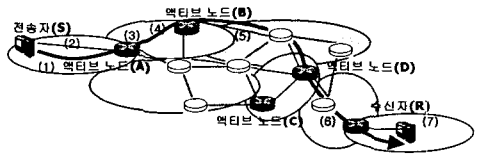

그림 7. 동적 패킷 라우팅 전달 과정

(1) 송신지 액티브 호스트(S)(일반적으로 시뮬레이션 서버에 해당)에서 패킷을 생성한다. (2)액티브 패킷을 기본 액티

브 노드(A)로 전달한다. 이때 패킷은 목적지 주소가 기본 액티브 노드의 주소를 포함한 IP 헤더로 캡슐화되어 전송된다. (3) 기본 액티브 노드는 액티브 패킷의 목적지 액티브 호스트에 대한 목적지-인접 액티브 노드로서 D를 선택하고, 액티브 라우팅 테이블에서 다음 액티브 노드(B)를 선택한다. (4) 다음 액티브 노드로 전송되는 패킷은 우선 목적지-인접 액티브 노드로의 터널링을 위해 캡슐화된 후, 다음 액티브 노드로의 터널링을 위한 캡슐화를 해서 전달한다. (5) 중간 액티브 노드로 동작하는 B는 액티브 패킷을 받고 자신의 액티브 노드로서 D를 선택하고 다시 캡슐화한 후 전송한다. (6) 목적지-인접 액티브 노드 D는 캡슐화된 패킷을 추출하는 과정에서 자신이 목적지-인접 액티브 노드임을 인식하고 원래의 액티브 패킷을 추출한 후 원래의 목적지인 액티브 호스트로 터널링을 통해 패킷을 전송한다. (7) 목적지 액티브 호스트(R)는 패킷의 목적지가 자신임을 인식하고 패킷을 처리한 후, 더 이상 패킷을 다른 곳으로 전달하지 않는다.

이러한 동적 라우팅 방식을 통해서 액티브 패킷은 시뮬레이션 서버에서 각 노드를 거쳐 IGI 호스트에 전달되게 되며, 지금까지 액티브 패킷의 라우팅 과정을 살펴보았다. 위에서 기술한 라우팅 과정은 액티브 패킷에 추가적인 라우팅 정보를 포함시키지 않은 경우, 즉, 패킷이 각 노드에 전재하는 액티브 라우팅 테이블 기반으로 전달되는 경우에 해당된다. 이러한 라우팅 방식을 홉당(hop-by-hop) 라우팅이라 하며, 액티브 패킷 라우팅 방식들 중 기본 방식이다. 만약 액티브 패킷에 발신지(source) 라우팅으로 명시되었다면 위의 라우팅 과정은 달라지게 된다.

액티브 토폴로지 정보를 기반으로 몇몇 특정 액티브 노드를 이용해 세션을 설정하는 경우, 패킷은 세션에 참여하는 액티브 노드를 경유해야 하며 이 경우에는 발신지 라우팅(source routing)이 적합하다. 발신지 라우팅 경로가 패킷에 명시된 경우, 혹은 실행 환경에서 해당 패킷 플로우에 대한 추가적인 라우팅 정보를 유지하는 경우는 명시된 경로를 통해 패킷이 전달된다[8].

3. 액티브 노드에서의 효율적인 코드 캐싱 기법

액티브 네트워크 상에서 패킷 처리를 위한 실행 코드 요청이 어떻게 이루어지는지에 대해서는 이미 MIT의 ANTS (Active Node Transfer System)[14]를 통해 제안되었으며, 실제코드를 요구 및 전송할 수 있는 코드 분배 기법 또한 제공되고 있다. 하지만 가상 전장환경의 IGI 서버와 동일한 서브넷에 있는 액티브 노드에서는 IGI 호스트로부터의 요청에 대한 액티브 패킷 처리를 위한 코드 요청은 빈번하게 그리고 지속적으로 발생하며, 매번 이러한 요청이 있을 때마다 요청에 맞는 코드를 전송해 주기 위한 코드 캐싱 작업이 수행되고 있다. 이러한 코드 캐싱 기법은 가상 전장환경이라는 특수한 특성을 갖는 액티브 네트워크 상에서는 적용하기가 어렵다. 따라서 네트워크의 성능향상을 위해서는 효율적인 캐싱 기법이 요구된다.

본 논문에서는 무수히 발생하게 되는 코드 요청에 대한 캐시 적중률(hit-ratio)을 높이고 실행 코드를 효율적으로 캐시(cache)에 유지함으로써 코드 요청 횟수를 줄이기 위한 효율적인 캐싱 기법을 제안하였다.

기존의 캐싱 기법에 응용되는 기본적인 알고리즘으로는 FIFO(First-In-First-Out), LFU(Least Frequently Used), LRU (Least Recently Used) 기법 등이 있다. FIFO 기법은 교체될 대상을 선택함에 있어 가장 먼저 들어온 것 즉, 가장 오래된 것을 선택하는 기법이다. 이 기법은 이해하기 쉽고 프로그램하기 쉽지만 성능이 항상 좋은 것은 아니다. 예를 들어 코드 적재 공간이 부족하여 코드 중에 하나를 삭제하여야 할 때 사용 빈도수가 높음에도 불구하고 가장 먼저 적재된 코드라면 사용 빈도수가 높음에도 불구하고 삭제되며 똑같은 실행이 요구될 때 코드를 다시 적재해야하는 비효율성이 있다. LFU 기법은 얼마나 많이 참조되어 졌는지에 착안한다. 이 기법은 교체가 필요한 경우에 참조 횟수가 가장 적은 것을 교체한다. 이 기법은 자주 사용되어 지는 코드를 계속 유지함으로써 좋은 성능을 나타낼 수 있지만 단기간에 많이 사용되고 오랜 시간동안 사용되지 않을 경우에도 캐시에 코드가 유지되는 단점이 있다. LRU 기법은 가장 오랫동안 참조되지 않은 것을 교체하는 기법이다. 이 기법은 참조의 최근성만을 기준으로 교체하기 때문에 참조 횟수를 반영하지 못한다.

액티브 노드에서는 다양한 패킷을 처리하기 위한 다양한 실행 코드가 필요하다. 이러한 실행 코드를 효율적으로 캐시에 유지함으로써 코드 요청 횟수를 줄임으로써 실행 코드 요청에 대한 패킷 처리 대기 시간을 감소시키고 패킷 처리 속도를 향상 시킬 수 있을 것으로 생각된다.

본 논문에서 제안하는 캐싱 기법은 캐시 관리자가 코드의 참조 횟수, 즉 코드의 실행 횟수에 대한 필드와 시간 제약(time limit)에 대한 필드를 참조하는 캐시를 관리하는 기법이다. 코드 정보 테이블(Code Information Table)에는 각 코드에 대한 식별자를 나타내는 필드(Code ID)와 코드 실행 횟수에 대한 정보를 나타내는 필드(Execution No), 시간 제약을

나타내는 필드(Time Limit No)로 구성 된다. 테이블의 구성은 표 1에 있는 구조를 따른다

표 1. 코드 정보 테이블의 구성

Code ID	Execution NO	Time Limit No
A	5	8
B	1	3
C	7	1

코드 실행 횟수 필드는 코드가 최초 실행될 때 '1' 값을 가지며, 재실행 될 때마다 값을 '1'씩 증가 시킨다. 이렇게 함으로서 코드 실행 횟수에 대한 정보를 유지한다. 시간 제약 필드는 시간 제약 값에 대한 정보를 주기적으로 갱신하여 유지한다. 코드가 최초로 실행된 후에 코드는 시간 제약에 대한 일정한 값(예를 들어 10)을 가지게 된다. 캐시 관리자는 코드가 실행된 후 일정 시간 마다(예를 들어 10시간) 시간 제약 값을 '1' 씩 감소시킨다. 시간 제약 값은 코드가 재실행 될 때 다시 초기 값(10)을 가진다. 제안한 알고리즘의 참조 횟수에 관한 알고리즘은 LFU 기법을 따른다.

그림 8은 제안한 기법의 동작순서를 순서도로 나타내었다. 순서도는 Execution No 필드를 참조하여 해당 코드를 삭제하는 과정과 Time Limit No 필드를 참조하여 해당 코드를 삭제하는 두 개의 과정이다.

제안한 알고리즘의 동작 과정을 살펴보면 다음과 같다. 이때의 동작 과정은 패킷 실행에 필요한 코드 요청에 대한 응답을 받은 상황으로 가정한다.

캐시 관리자는 코드를 캐시에 넣기 위해 캐시에 공간이 있는지를 검사한다. 캐시에 공간이 있으면 코드를 캐시에 적재한다. 만약 필요한 공간이 부족하면, 캐시 관리자는 실행 횟수 필드를 검사한다. 검사된 코드 중에서 실행 횟수가 가장 적은 코드를 삭제한다. 캐시 관리자는 코드 적재를 위해 필요한 공간이 생길 때 까지 이 과정을 반복한다. 이 과정에서는 단지 코드의 실행 횟수만을 참조하기 때문에 단시간에 여러 번 실행되고 그 이후에 장시간 동안 참조되지 않는 코드도 계속 캐시에 남아서 공간을 차지하게 된다. 이러한 점을 개선하기 위해 캐시 관리자는 시간 제약 테이블을 주기적으로 감소시킨다. 이때 만약 시간 제약 값이 0이 되는 코드가 있으면, 캐시 관리자는 실행 횟수에 상관없이 해당 코드를 삭제함으로써 캐시 공간을 확보한다.

예를 들어 코드 정보 테이블에서 코드 B와 C의 현재 상태에서 다음 시간 제약 값에 대한 갱신 때까지 둘 다 사용되지 않았다면, B의 시간 제약 값은 '2' 가되고 C의 시간 제약 값은 0이 된다. 이때 캐시 관리자는 코드 C를 캐시에서 삭제함으로써 캐시 공간을 확보한다. 이때 C의 참조 실행 횟수는 B보다 훨씬 많지만 단시간에 여러 번 사용되고 그 후에 사용되지 않는 코드로 간주할 수 있다.

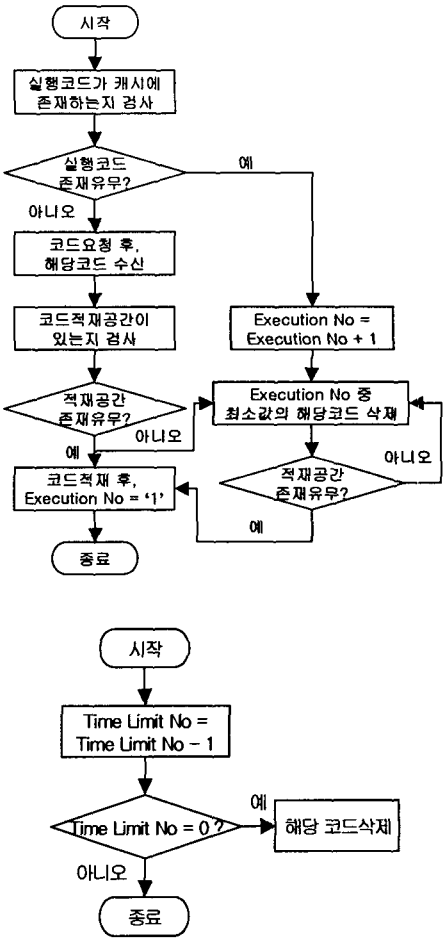

그림 8. 코드 캐싱 기법의 동작 과정

제안한 코드 캐싱 기법을 통해 자주 실행되는 코드를 캐시에 유지하고 장시간 실행되지 않는 코드를 삭제함으로써 코드 요청 횟수를 줄이고 캐시 공간의 비효율적인 사용을 줄임으로써, 코드 캐싱으로 인한 트래픽 부하를 상당부분 감소시킬 수 있다.

IV. 가상 전장환경에서의 액티브 네트워크 모델링

가상 전장환경상에서의 액티브 노드는 앞서 기술된 RED 확장 기법, 동적 라우팅 기법, 동적 멀티캐스팅 기법 및 동적

경로설정 기법을 적용하여 모델링한 후, 액티브 노드를 기본 노드로 하는 액티브 네트워크를 모델링하였다.

1. 액티브 노드 모델링

현재 인터넷의 발전 과정으로 미루어 볼 때 기술적, 상업적인 이유로 인터넷의 모든 라우터가 액티브 라우터로서 동작할 수 없기 때문에 초기 단계에서는 액티브 네트워크 상의 액티브 노드들은 서로 물리적으로 직접 연결되지 못할 가능성이 크다. 이는 액티브 네트워크 상에서의 패킷 전달 방법과 라우팅 프로토콜을 복잡하게 만드는 가장 큰 요인이 된다.

패킷 전달 방법에 있어서는 물리적으로 연결되어 있지 않은 다음 액티브 노드로 패킷이 전달되는 것을 보장하기 위한 추가적인 방법이 필요하다. 본 논문에서는 이를 위해 액티브 패킷에 추가적인 IP 헤더를 캡슐화하여 전달하는 IP 터널링 방법을 사용하도록 제안하다. IP 터널링 방법은 현재 Mobile IP와 MBONE 등에서 사용되고 있는 방법이다. 다음으로, 다음 액티브 노드를 선택하는 과정에 있어서는 액티브 패킷의 처리 결과와 액티브 네트워크 토폴로지 정보를 고려해야 한다. 만약 처리 결과를 통해 라우팅 정보를 얻을 수 있다면, 이 라우팅 정보를 통해 다음 액티브 노드를 선택할 수 있다. 그렇지 않다면, 액티브 노드 자신이 유지하고 있는 라우팅 정보를 기반으로 다음 액티브 노드를 결정해야 한다. 만약, 인터넷의 모든 라우터가 액티브 라우터로서 동작한다면 기존 인터넷 라우팅 프로토콜에 의해서 만들어진 라우팅 정보를 그대로 사용할 수 있을 것이다. 그러나 액티브 노드들을 물리적으로 서로 연결되어 있지 못할 수도 있기 때문에 액티브 노드들로만 구성된 액티브 네트워크 토폴로지 정보를 유지하고 액티브 패킷의 라우팅을 위한 액티브 라우팅 테이블을 별도로 구성해야한다. 즉, 새로운 액티브 네트워크 라우팅 프로토콜이 요구된다. 이웃한 액티브 노드들이 물리적으로 직접 연결되어 있지 못하기 때문에 이웃한 노드에 대한 최신의 정보(이웃한 노드들의 리스트, 이웃한 노드와 링크 상태)를 유지하고 그에 따른 대응이 어렵기 때문에, 액티브 네트워크 라우팅 프로토콜은 일반적인 인터넷 라우팅 프로토콜보다 좀더 복잡할 것이다[8].

기존의 액티브 노드에 있어, 패킷이 수신되면 우선 패킷 분류기가 액티브 패킷과 일반 패킷을 분류하여 액티브 패킷일 경우 패킷 내에 들어 있는 전장정보나 프로그램을 실행환경으로 보내어 처리하게 한다. 실행환경에서 처리 후 얻어진 정보는 상태 정보 표(State Information Table : SIT)에 저장하며 패킷 생성기를 이용하여 주변 라우터에 혼잡에 대한 정보를 제공할 수 있다.

본 논문에서 제안한 액티브 노드는 MIT의 ANTS를 기반으로 하여 그림 9와 같이 기존의 액티브 노드에 하나의 작업을 수행하기 위해 필요한 일련의 액티브 패킷들이 같은 액티브 노드를 경유하도록 하기 위한 경로 설정기법을 제공하는 경로설정 모듈, 네트워크 상의 트래픽을 제어하기 위한 혼잡제어모듈 등을 포함하고 있는 구조를 가지고 있으며, 각각의 모듈에 대해 구체적으로 기술하면 다음과 같다.

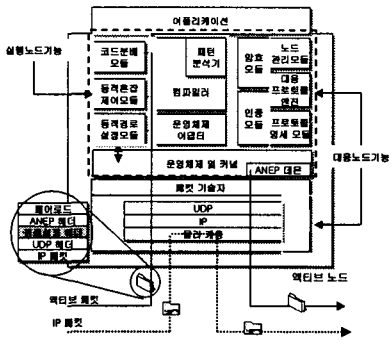

그림 9. 액티브 노드 모델링

1.1. 동적 경로설정 모듈

중간의 액티브 노드에 설치될 경로설정모듈은 UDP와 ANEP(Active Network Encapsulation Protocol) 사이에서 그 기능을 수행한다. UDP와 ANEP와는 포트를 통하여 액티브 패킷을 주고받는다. 경로설정모듈을 사용자 레벨에서 어플리케이션처럼 구현하고 포트를 할당하는 것은 구현을 용이하게 하고 액티브 노드에 이식을 쉽게 하기 위해서이다. 또한 액티브 패킷을 특별한 표시없이 포트번호로 구별할 수 있다. 만약 전송 계층에서 액티브 엔진을 구현하면 속도는 좀 더 빨라질지 모르나 커널 레벨에서 구현해야 하는 어려움이 있고 액티브 노드에 이식도 쉽지 않을 뿐 아니라 보안상의 위협도 존재하게 된다. 그림 9는 액티브 노드에서의 액티브 엔진의 위치와 패킷의 흐름을 나타낸 것이다.

경로 설정을 요구하는 액티브 패킷이 들어오면 UDP 포트번호를 보고 액티브 엔진으로 액티브 패킷을 보내준다. 액티브 엔진은 다음 액티브 노드를 결정한 후 사본은 ANEP를 전달하여 액티브 패킷의 내용을 실행하도록 하고, 원본은 UDP를 통하여 다음 노드로 전송한다. 경로 설정을 요구하지 않는 액티브 패킷이 들어오면 UDP는 바론 ANEP 데몬으로 액티브 패킷을 넘겨주고 IP 패킷이 들어오면 액티브 노드는 기존의 라우터에서와 같이 단순히 다음 노드를 결정해서 전달하다.

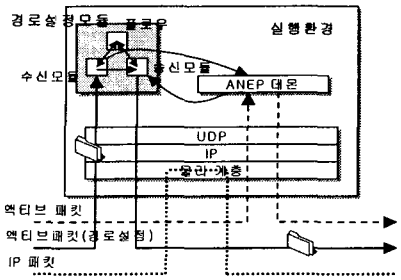

그림 10. 액티브 노드의 경로설정모듈 구조

그림 10과 같은 구조에서는 다른 패킷들에게 영향을 끼치지 않고 경로 설정이 필요한 액티브 패킷들에 대해서만 경로설정모듈에서 필요한 작업을 수행할 수 있다. 또한 기본 전송계층인 UDP와 IP에 전혀 수정이 필요하지 않으므로 기존의 네트워크에 쉽게 적용할 수 있다.

경로설정모듈은 전 노드로부터 액티브 패킷을 수신했을 때 동작하는 수신 블록과 액티브 패킷을 다음 노드로 전송하기 위한 동작을 수행하는 송신 블록으로 나눌 수 있다. 플로우 정보 저장소는 액티브 노드로 들어오는 액티브 패킷들의 정보를 유지한다. 경로 설정 프로세스는 다음 액티브 노드를 결정하여 액티브 패킷들이 모두 같은 액티브 노드르 거쳐 실행될 수 있도록 한다[13].

1.2. 동적 혼잡제어 모듈

단지 패킷 속에 필요한 정보와 프로그램을 담아 전송 시키면 실행환경에서 그 목적에 맞게 동작 동작하기 때문이다. 이러한 실행환경이 없는 일반 라우터의 경우에는 네트워크 관리자가 라우터를 직접 설정해 주어야 할 뿐만 아니라 새로운 장비의 교체까지 해야 한다. 그림 11은 액티브 노드가 그 목적에 따라 구조의 변 없이 동작함을 보여준다.

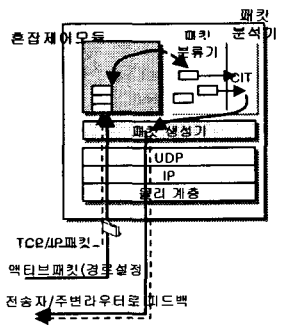

그림 11. 동적 혼잡제어 확장 모듈의 구조

내부 라우터는 큐 관리 방식에 있어 앞서 기술된 RED 확장을 사용하였다. 전반적인 구조는 RED와 거의 유사하지만 문턱값(threshold) 값에 따라 패킷을 누락시키는 RED와는 다르게 RED 확장은 피드백을 사용한다. ECN(Expl-icit Congestion Notification)을 사용한 RED(이하 RED ECN)와의 차이점은 혼잡지점에서 바로 전송자에게 혼잡신호를 줄 수 있다는데 있다. RED ECN 또한 전송자가 혼잡 신호를 받게 되는데 까지 걸리는 시간이 RTT에 영향을 받으므로 EACC와 비교해서 신속한 처리가 어렵다[10].

2. 액티브 네트워크 모델링

워게임 모델에 적용할 액티브 네트워크 구조는 당분간 전통적인 일반 네트워크 구조와 통합되어 사용되어져야 할 것이다. 따라서 기존의 네트워크와의 효율적인 연동을 위하여 액티브 네트워크의 구조도 이를 고려한 형태로 구성되어야 하며, 향후 완전한 액티브 네트워크 구조(Compl-ete Active Network)로 구현되더라도 최소의 관리노력으로 업그레이드 될 수 있도록 하여야 할 것이다.

따라서 본 논문에서는 가상 전장공간 상에서 기존의 네트워크와 효율적으로 연동되며, 네트워크 트래픽을 최소화할 수 있는 액티브 네트워크 기반구조를 구성하고자 하며, 기본 단위는 도메인이며, 액티브 네트워크는 기본적으로 도메인으로 구성하였다. 단, 도메인 내의 모드 노드가 액티브 노드인 경우는 고려하지 않았다.

2.1. 가상 전장환경상의 액티브 네트워크 구성시 고려사항

가상 전장공간 상에서 액티브 네트워크를 구성하기 위해서는 다음의 두 가지 상황을 고려해 볼 필요가 있다. 첫째는 도메인 내부의 노드 중 일부가 일반 노드(Non-Active Node)인 경우, 둘째는 도메인과 도메인간의 노드가 일반 노드인 경우이다.

첫 번째 상황인 도메인 내부의 노드 중 일부가 일반 노드일 경우는 기존의 액티브 네트워크 기술에서 해결책을 다루고 있다. 즉, 캡슐이 액티브 노드를 지날 때는 해당 노드에서 필요한 처리를 하지만, 일반 노드를 통과할 때는 forward 시키도록 구성하고 있다.

두 번째 상황인 도메인과 도메인 사이의 중간 노드들이 일반 노드인 경우는 앞서 기술된 동적기법 들에서 이미 가정하고, 해결책을 제시하였으며, 앞의 상황보다 좀 더 신중한 모델링이 필요하다. 액티브 네트워크의 도메인 구성에서 액티브 노드를 관리 대상으로 편입하였다. 따라서 중간 노드가 액티

브 노드일지라도 자신이 속한 도메인의 관리 제어를 받지 않는 액티브 노드는 정책상 일반 노드로 처리한다. 시뮬레이션 서버를 포함한 도메인(서버 도메인)에서 IGI 호스트를 포함한 도메인(이하 호스트 도메인)으로 캡슐이 전송될 때, 서버 도메인의 게이트웨이 역할인 노드는 서버 도메인이 관리하는 마지막 노드가 된다. 서버 도메인에서 마지막 노드를 지나는 순간부터 호스트 도메인 영역의 게이트웨이 역할의 노드에 도달하기까지 거치는 중간 노드는 모두 일반 노드로 인식을 하여 전송(forward) 처리만 하도록 한다. 이 경우는 서버 도메인과 호스트 도메인은 신뢰 관계(Trust)가 선행되어 있어야 하며, 노드의 신뢰 정보가 각각의 도메인에 소속된 노드 운영체제 컨트롤러 사이에 교환이 되어져야 한다. 도메인의 노드에서는 대상 도메인으로 캡슐을 전송하기 전에 액티브 노드간에 신뢰 관계를 확인하여 이와 같은 캡슐 전송 처리를 한다.

2.2. 가상 전장환경에서의 액티브 네트워크 구성 예

위에서 기술한 고려사항을 포함한 하부구조를 기반으로 창조 21 모델(하나의 커다란 도메인)이라는 가상 전장환경 상에서 액티브 네트워크 모델을 그림 12와 같이 구성하여 보았다.

그림 12. 도메인(혹은 시뮬레이션 모델) 안에서의 액티브 네트워크 구성

예를 들어, A라는 서브넷 안에 있는 IGI 호스트가 하나의 전장자료 혹은 서비스를 시뮬레이션 서버에 요청(최초의 Req)하게 될 경우에 기존의 워게임 모델에서는 시뮬레이션 서버가 모든 응답(최초의 Rep)을 해야만 했다. 이것은 B라는 서브넷에 있는 IGI 호스트로부터 똑같은 서비스 요청(현재의 Req)을 받게 될 경우 시뮬레이션 서버가 똑같은 응답을 호스트에 전달해야만 한다. 이것은 시뮬레이션 서버를 포함한 전체 도메인에 커다란 부하를 가져오게 되며, 워게임 모델 자체에 운영되지 못하게 된다. 이에 그림 12에서와 같이 B라는 서브넷에 있는 IGI 호스트로부터 똑같은 서비스 요청(현재의 Req)이 있을 때, A 서브넷의 액티브 노드에서 이미 자료를 저장하고 있으므로 B 서브넷의 액티브 노드에 서비스(코드)를 전송하고, 그 서비스를 전달 받은 노드에서 서비스를 요청한 IGI 호스트에 전달하면 된다. 이것은 요청이 작을때에는 커다란 효율성을 갖지 못하지만, 요청(액티브 패킷)의 양일 많아질수록 가상 전장환경의 네트워크 트래픽을 현저하게 감소시킬 수 있을 것이다.

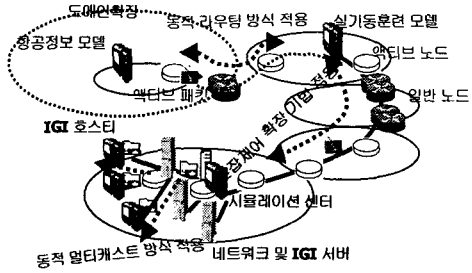

그림 13. 도메인간의 액티브 네트워크 구성

위의 그림 13은 도메인 안에서의 액티브 네트워크를 확장하여 액티브 네트워크 도메인과 일반 네트워크 도메인간의 구성을 보여주고 있다.

앞서 서로 다른 네트워크(액티브 네트워크와 일반 네트워크)간의 액티브 패킷전송방식에 대해서는 동적경로설정 및 동적혼잡제어방식을 통해 제시하였다. 이를 기반으로 하여 그림 13에서는 창조 21 모델(주 시뮬레이션 센터에서 사용)과 새로운 모델(예, 실기동훈련모델, 가상모의훈련모델, 항공정보모델 등등)과의 연동을 보여주고 있다.

이와 같이 실제 전장상황과 똑같은 상황을 구축하기 위해 계속해서 새로운 모델은 개발될 것이며, 이러한 모델에 대한 기존의 모델과의 연동은 반드시 필요할 것이다. 이에 본 논문에서는 액티브 네트워크를 이용한 가상 전장환경을 모델링하였다.

V. 모의 실험 및 성능 평가

본 장에서는 모의실험 환경을 설명하고 모의실험 결과를 분석한다. 본 모의실험에서는 기존의 실험을 토대로 액티브 패킷의 실행 및 그 실행을 통한 네트워크 트래픽 감소를 위해 적용된 기법들을 액티브 노드에 추가함으로써 액티브 노드의 성능과 액티브 네트워크 트래픽에 어떠한 영향을 미치는지 성능을 평가한다.

1. 모의실험 환경

본 논문에서는 가상 전장환경상에 적용한 RED 확장 기법, 동적 라우팅 기법, 액티브 멀티캐스팅 기법 및 동적 경로설정 기법의 성능 평가를 위하여 LBNL(Lawrence Berkely National Laboratory)의 NS-2(Network Simulator-II) 네트워크 시뮬레이터를 이용하였으며, 액티브 노드를 시뮬레이션하기 위한 토폴로지 생성을 위해 Georgia Technology에서 개발한 GT-ITM으로 생성한 트랜짓-스텁(Transit-Stub) 구조를 사용함으로써 그림 14에서와 같이 현재의 국방네트워크 환경과 가장 유사한 환경을 설정하였고[13], 추가적으로 일반 노드와 액티브 노드를 각각 설치하여 액티브 노드와 일반 노드가 서로 공유하도록 하여 좀더 실질적인 네트워크 구조를 설정하였다.

주변 라우터와 내부 라우터 사이의 지연시간을 변화시켜 가면서 성능을 측정하였다.

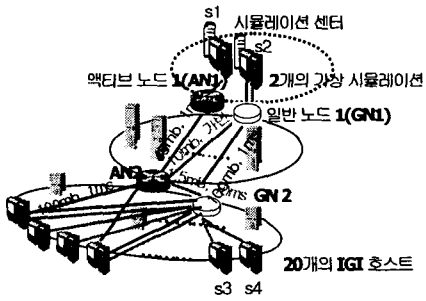

그림 14. 네트워크 토폴로지

전송자에서 주변 라우터까지의 지연시간은 10ms이고 대역폭은 각각 1Mb로 고정시키고 각 전송자는 실험 시작부터 일련적으로 패킷을 전송한다. 실험에서 각 라우터의 큐 길이는 50이고, 모의실험시간은 200초로 설정하였다. 동적혼잡제어와 동일한 조건에서 모의 실험하기 위해 주변 라우터와는 TCP만 연결되며 UDP 패킷에 대한 주변 라우터의 필터링(zombie list)기능은 사용하지 않았다[10].

2. 네트워크 트래픽 모의실험

2.1. 동적 라우팅 확장 모의실험 결과

동적 라우팅 확장기법은 주변 라우터에서 각 전송자로 피드백된 정보로 전송자는 윈도우 크기를 조절하게 된다. 동적 혼잡제어 기법과 비교할 때 주변 라우터에서 전송자까지의 지연시간이 작더라도 기존의 TCP와 같은 처리량을 나타냈다. 동적 혼잡제어 기법은 혼잡이 발생할 경우 특정 전송자의 트래픽만 필터링하게 된다. 이는 어떤 전송자는 지속적으로 처리량에 영향을 받는 반면, 어떤 전송자는 지속적으로 높은 윈도우 크기로 전송 가능하기 때문에 공정성(fairness)에 문제점을 갖는다. 동적 라우팅 기법은 이를 개선하기 위해서 주변 라우터와 내부 라우터 간의 통신을 사용하였다. 표 2는 TCP에 동적 라우팅 기법을 적용했을 경우와 혼잡제어기법을 적용했을 경우의 결과이다. 지연(bandwidth*delay)의 경우에도 동적 라우팅 기법은 기존의 방식과 거의 비슷한 결과를 얻었음을 알 수 있다. 지연이 클 경우 기존의 방식뿐만 아니라 혼잡제어방식보다 성능이 개선되었다.

동적 라우팅 기법은 혼잡이 발생하여 내부 라우터에서 피드백 될 경우 주변 라우터에서 각 전송자에게 네트워크에 전송 가능한 윈도우 크기를 전송해 주기 때문에 특정 전송자가 불이익을 받지 않게 된다.

표 2. 동적 라우팅 방식과 혼잡제어방식의 처리율 비교

방식 / 지연	동적라우팅 (ms)	혼잡제어 (ms)	처리율 증가
10	4.63	4.40	-4.9%
50	4.49	4.35	-3.1%
100	4.41	4.19	-4.9%
150	4.18	4.30	2.8%
200	4.19	4.23	0.9%
250	3.95	4.29	8.6%
300	3.83	4.17	8.8%
350	3.85	4.11	6.7%
400	3.82	4.10	7.3%
450	3.83	4.08	6.5%
500	3.75	4.13	10%

또한 RTT에 따른 처리가 가능하기 때문에 지연의 크기에 무관하게 표 2에서와 같이 처리량이 높게 나타났다[10].

2.2. 동적 멀티캐스팅 모의실험

동적 멀티캐스팅 모의실험에서는 기존의 시뮬레이션 환경에 액티브 네트워크를 설치했을 때와 설치하지 않았을 경우에 각 데이터의 한 패킷 양을 각각 512바이트, 1024바이트를 전송했을 때 데이터 처리량(전송속도)을 측정하였다. 시뮬레이션의 공정성을 기하기 위해 임의의 값을 이용하여 각 방식별 100번을 반복 수행한 후, 평균을 결과 값으로 채택하였다. 표 3, 4는 각각의 전장 데이터 패킷인 512바이트, 1204바이트를 전송했을 때의 데이터 전송속도를 나타내었다. 표 3의 512바이트는 데이터 패킷이 수신 측으로 전달되는 전송속도가 액티브 네트워크를 적용했을 때 속도가 약간은 증가함을 나타내고 있다.

표 3. 512바이트의 패킷 전송속도

유형 / 단계수	액티브 네트워크	일반 네트워크	전송율 증가
1	138	137	0.73%
5	142	142	0.00%
10	137	136	0.73%
15	135	136	-0.73%
20	139	140	-0.71%
25	135	136	-0.73%
30	149	144	3.47%

표 4는 데이터 패킷 크기가 커짐에 따라 액티브 네트워크를 적용했을 때 데이터의 계층적 특성에 대한 액티브 노드의 처리로 인해 전송속도가 오히려 PIM-DM 방식을 이용한 일반적인 네트워크보다 저하되었다. 향후 두 방식간의 전송속도의 차이를 최소화하기 위한 액티브 노드 구현에 대한 연구가 필요하다.

표 4. 1024바이트의 패킷 전송속도

유형 / 단계수	액티브 네트워크	일반 네트워크	전송율 증가
1	112	108	3.70%
5	113	113	0.00%
10	106	118	-10.16%
15	110	117	-5.98%
20	108	116	-6.89%
25	105	116	-9.48%
30	115	124	-7.26%

2.3. 동적 경로설정 모의실험

복사 후 전송방식의 성능을 실험하여 보았다. 표 5는 복사 후 전송방식의 전송지연시간을 측정한 결과를 표로 나타낸 것이다. 액티브 패킷을 모두 받을 때까지 기다려서 실행한 후 다음 노드로 전송하는 기존의 방식과 TCP, UDP를 비교하였다. 2개의 송신자가 20개의 수신자로 트래픽을 1M바이트를 보냈을 때의 평균 전송지연시간을 나타낸 것이다.

표 5. 복사 후 전송 방식의 전송지연시간

방식 / 노드수	TCP (ms)	UDP (ms)	SCF (ms)	CF (ms)	SCF 대비 전송증가(배수)
1	37	10	10	10	1.0
2	37	10	20	10	2.0
3	37	10	40	10	4.0
all	37	10	60	10	6.0

TCP와 UDP는 종단간의 전송에만 관여하므로 TCP와 UDP는 액티브 노드의 개수가 늘어나도 일정한 값을 유지한다. 복사 후 전송 방식은 액티브 노드의 개수가 증가할수록 전송지연시간이 점점 짧아지는 반면, Store-Compute-Forward 방식은 전송지연시간이 점점 증가해서 나중에는 부하가 큰 TCP의 전송지연시간보다 길어지는 것을 볼 수 있다. 따라서 액티브 네트워크에서 복사 후 전송 방식을 사용하는 것이 효율적이다.

표 6은 중간의 액티브 노드의 수를 증가시키면서 액티브 패킷을 보냈을 때의 전송지연시간을 나타낸다. 패킷은 1M바이트를 전송했다.

표 6. 액티브 노드의 수에 따른 전송지연시간

방식 / 노드수	TCP (ms)	UDP (ms)	CF (ms)	TCP 대비 전송증가(배수)
1	32	10.1	10.5	3.05
2	32	10.1	10.3	3.11
3	32	10.1	10.2	3.14
all	32	10.1	10.2	3.14

TCP와 UDP는 종단간의 전송만을 담당하므로 중간의 액티브 노드의 수와 상관없이 일정한 값을 갖는다. 표 6을 보면 액티브 노드의 개수에 상관없이 액티브 엔진이 UDP의 전송지연시간과 거의 비슷한 결과를 나타내고 있으며, UDP와 액티브 엔진의 전송지연시간만을 비교하면, 액티브 엔진의 전송지연시간이 액티브 노드의 수가 증가할수록 짧아지는 것을 볼 수 있다. 이는 액티브 노드의 수가 증가할수록 경로 요청 패킷에 대한 경로 응답 패킷이 돌아오는데 걸리는 시간이 줄어들기 때문이다[13].

VI. 결론 및 향후 연구 과제

가상 모의훈련 전장환경은 컴퓨터를 이용한 가상의 전장환경을 제공하여, 군사작전의 분석 및 훈련용으로 사용된다. 하지만 환경적인 특성상 다량의 전장 데이터(이하 PDU, 액티브 패킷 혹은 코드라 함)의 송수신을 고려했을 때, 종단장치(시뮬레이션 서버와 IGI 호스트)간의 네트워크 트래픽 부하가 현저하게 증가하여 중간 노드들의 트래픽 처리율이 현저하게 저하될 수 있고, 이는 실제 전장환경과 유사한 환경을 구축하고 모의훈련하고자 하는 가상 '모의훈련 전장환경'의 구성에 대한 실패를 초래할 수도 있다.

최근의 가상 모의훈련용 워게임 모델에서의 다양한 액티브

응용을 지원하기 위해서는 액티브 네트워크의 링크, 실시간 전송, 보다 효과적인 멀티캐스트 프로토콜이 요구된다. 전장 화상회의, 동영상 및 고용량 이미지와 같은 큰 용량의 데이터를 실시간으로 전송하기 위해서는 데이터의 특성을 고려해야 한다. 최근에는 데이터의 특성을 보다 효과적으로 적용하기 위하여 액티브 패킷을 사용해서 멀티미디어 데이터의 특성에 따라 대역폭과 전송지연을 미리 예측 처리하여 데이터를 전송함으로서 데이터 트래픽의 공정한 분배와 전송속도를 증가시킬 수 있게 되었다. 이에 본 논문에서는 기존 액티브 네트워크 기술들에 대한 통합을 통해 이러한 문제점들을 해결하고 가상 전장환경에 적용한 후, 아래와 같은 모의실험을 통해 그 유효성을 검증하였다.

우선, 기존의 혼잡제어 방식이 혼잡 발생 시점에서 혼잡에 대한 처리를 할 수 없다는 문제점을 가지고 있다. 즉 혼잡이 발생한 후 패킷 누락에 대한 신호가 전송자에게 도착 했을 경우에 혼잡을 감지하게 된다. 혼잡을 감지하는데 걸리는 이런 제어시간이 길어지면 길어질수록 혼잡은 더욱 악화된다. 혼잡 발생과 동시에 혼잡에 대한 처리를 할 수 있다면 전송자는 보다 빠르게 혼잡에 대처 할 수 있을 것이다. 따라서 네트워크의 내부 정보를 이용하여 혼잡 발생 시점에서 혼잡을 제어하는 ACC 확장기법을 적용함으로써, 기존의 혼잡제어의 비효율성을 제거하여 성능이 향상됨을 알 수 있었다. 또한, 동적으로 변화하는 네트워크 트래픽을 단지 평균 큐 크기 정보만을 기초로 혼잡상황을 제어하려는 기존의 RED 알고리즘을 개선한 RED 확장 기법을 적용하였다. RED 확장 기법은 기존의 네트워크 혼잡상황시 RED 알고리즘이 계산하는 패킷 폐기 확률 방법을 개선하여 휴리스틱한 방법을 통하여 패킷 폐기 확률을 계산한다. 시뮬레이터를 이용한 실험을 통하여 공정성과 전송 성능 및 전송 지연 시간에 대한 성능을 비교 분석하여 보았다.

끝으로, 액티브 패킷이 같은 액티브 노드를 지나도록 하기 위하여 액티브 노드를 찾기 위해 패킷을 보내는 방법을 적용하여 노드들이 중간 노드들의 위치나 상태에 대한 정보를 가지고 있지 않아도 되고 기존의 IP 라우팅 프로토콜을 사용하며 IP 계층에서 조각화가 발생해도 같은 액티브 노드를 경유할 수 있도록 하였다.

향후 연구로는 지금까지의 개념을 좀더 확장하여 액티브 라우터와 일반 라우터간의 상호작용을 통해서 가능한 대역폭을 최대한 활용할 수 있는 방안을 연구하는 것이 필요하고, 광대역 네트워크를 구축했을 때의 네트워크 트래픽을 제어할 수 있는 연구가 요구되며. 모의 실험결과를 토대로 실제적인 네트워크 상의 모든 트래픽에 대해서 범용적으로 혼잡제어가 가능한 라우터 구성에 대한 연구가 이루어져야 할 것이다. 또한 본 연구에서 적용한 각각의 확장 기법을 확장하여 네트워크 레벨의 Qos를 지원하는 라우팅 기법과 네트워크 안정성을 위한 보안 라우팅 기법에 대하여 연구해야 할 것이다.

참 고 문 헌

[1] 장상철, 손미애, 정상윤, 장동옥, "차세대 시뮬레이션 연동체계 기술연구", 한국국방연구원, 2001.

[2] 김대석, 양병희, 류제철, "합성전장훈련체계 개발방안 연구 I, II", 국방과 기술, 258호, 2000.

[3] 김대석, 양병희, 류제철, "HLA 페더레이트 개발을 위한 ROM 프레임워크 설계 및 구현", 한국정보처리학회논문지 D, 제 9-D권, 제 6호, 2002.12.

[4] 서혜숙, 김태윤, "HLA 기반의 시뮬레이션 모형 개발 및 확장 연구", 한국정보처리학회 추계 학술발표논문집, 제 8권, 제 2호, 2001.

[5] Sumi Choi et al., "Configuration Sessions in Programming Networks," Proc. IEEE INFOCOM, 2001.

[6] Jonathan T. Moore and Michael Hick, "A Service Layer Routing Protocol for PLAN," PLAN Project, November 1997. http:///www.cis.upenn.edu/~switchware/PLAN>

[7] Javier Alvarez and Jessica Kornbium, and Erick Messing, "Simulating Link State Routing in an Active Network." TCO M500 PLAN Project. http://www.cis.upenn.edu/~messing/academic/tcomm.html.

[8] 안상현, 김경춘, 한민호, 나중찬, "액티브 라우터를 가진 IP 네트워크를 위한 OSPF 프로토콜의 확장 및 액티브 패킷 전달 방식", 정보과학회논문지, 제 30권, 제1호, 2003.

[9] S.Bhanttachariee, K. Calvert, and E. Zegura, "On Active Networking and Congestion," Technical Report GIT-CC-96-02, College of Computing, Georgia Tech., Atlanta, GA, 1996.

[10] 최기현, 장경수, 신호진, 신동렬, "액티브 라우터의 피드백 메커니즘을 이용한 혼잡제어 기법", 정보처리학회논문지 C, 제 9-C권, 제 4호, 2002.8.

[11] 구자헌, 송병훈, 정광수, 오승준, "라우터에서의 동적인 혼잡제어를 위한 새로운 큐 관리 알고리즘", 정보과학회논문지, 제 28권, 제 4호, 2001,12.

[12] Floyd, S., and Jacobson, v., "Random Early Detection Gatewqys for Congestion Avoidence," IEEE/ACM Transac tion on Networking, August 1993.

[13] 윤보영, 채기준, 남택용, "IP 망에서 액티브 패킷의 경로 설정 및 신뢰성 전송", 정보처리학회논문지 C, 제 9-C권, 제 5호, 2002.10

[13] 안상현, 김경춘, 한민호, 나중찬, "액티브 라우터를 가진 I P 네트워크를 위한 OSPF 프로토콜의 확장 및 액티브 패 킷 전달 방식", 정보과학회논문지, 제 30권, 제 1호, 2003.

[14] David J. Wetherall, John V. Guttang and David L. Tenn enhouse, "ANTS : A Toolkit for Building and Dynamically Deploying Network Protocols," IEEE OPENARCH, 1998.

정 창 모(Chang-Mo Jung) 준회원

1983년 : 육군사관학교 전산학과 (이학사)
1988년 : 국방대학원 전산학과 (이학석사)
2000년 : 한남대학교 대학원 컴퓨터공학과 박사과정

<관심분야> : 컴퓨터 네트워크, 시뮬레이션, SW공학

이 원 구(Won-Goo Lee) 준회원

2000년 : 한남대학교 컴퓨터공학과 (공학사)
2002년 : 한남대학교 컴퓨터공학과 (공학석사)
2002년 : 한남대학교 대학원 컴퓨터공학과 박사과정

<관심분야> : 컴퓨터 네트워크, 정보통신 정보보호

김 성 옥(Sung-Ok Kim) 준회원

1966년 : 연세대학교 수학과 (이학사)
1976년 : Univ. Minn 전산학과 (이학석사)
1989년 : 연세대학교 대학원 수학과(이학박사)
1976년 ~ 1983년 : 국방과학연구소 선임연구원
1995년 ~ 1996년 : 한국정보과학회 지부장
1983년 ~ 현재 : 한남대학교 컴퓨터공학과 정교수

<관심분야> : 시뮬레이션/병렬시스템, 수치해석

이 재 광(Jae-Kwang Lee) 종신회원

1984년 : 광운대학교 전자계산학과 (이학사)
1986년 : 광운대학교 대학원 전자계산학과(이학석사)
1993년 : 광운대학교 대학원 전자계산학과(이학박사)
1986년 ~ 1993년 : 군산전문대학 전자계산학과 부교수
1997년 ~ 1998년 : University of Alabama 객원교수
2002년 ~ 현재 : 한남대학교 컴퓨터공학과 정교수

<관심분야> : 컴퓨터 네트워크, 정보통신 정보보호