논문 2004-41SD-3-9

패킷 방식 네트워크상의 적응적 경로 선정을 위한 군집체 특성 적용 하드웨어 구현

(Hardware Implementation of Social Insect Behavior for Adaptive Routing in Packet Switched Networks)

안 진 호*, 오 재 석*, 강 성 호*

(Jin-Ho Ahn, Jae Seuk Oh, and Sungho Kang)

요 약

생태계의 군집 특성을 네트워크 환경에 적용하여 급변하는 환경에 대한 자가 적응 및 생존 특성을 부여하는 연구가 최근 많은 주목을 받고 있다. 그 중 AntNet은 개미를 모델링한 모바일 에이전트를 사용하여 최적의 네트워크 경로를 선택하는 적응적 라우팅 알고리즘이다. 본 논문에서는 SoC 시스템에 적용 가능한 AntNet 기반 하드웨어 구조를 제안한다. 제안된 구조는 기존 알고리즘 수준의 AntNet을 하드웨어 레벨로 근사화 하여 설계되었으며, 기존 AntNet과 가상 네트워크 구조에서의 비교를 통하여 그 타당성을 검증하였다. 그리고 RTL 수준의 설계 및 합성 결과를 통하여 제안된 하드웨어 구조가 AntNet 기반 라우팅 구현에 효과적임을 확인할 수 있었다.

Abstract

Recently, network model inspired by social insect behavior attracts the public attention. The AntNet is an adaptive and distributed routing algorithm using mobile agents, called ants, that mimic the activities of social insect. In this paper, we present a new hardware architecture to realize an AntNet-based routing in practical system on a chip application. The modified AntNet algorithm for hardware implementation is compared with the original algorithm on the various traffic patterns and topologies. Implementation results show that the proposed architecture is suitable and efficient to realize adaptive routing based on the AntNet.

Keywords : AntNet, Adaptive Routing, 군집 특성, 모바일 에이전트

I. 서 론

현존하는 최신의 네크워크 기술 대부분은 고성능의 장비와 숙련된 관리자를 기반으로 하기 때문에 점차 거대해지는 네트워크 크기와 그 트래픽에 대응하려면 네크워크 설계 및 유지비용이 기하급수적으로 증가하게 된다. 이에 최근에는 고속의 트래픽 처리와 동시에 네트워크망의 효율적 관리 및 끊임없이 진화하는 네트워크 기술에 대응하여 기존 시설의 재활용을 통한 인프라 구축비용의 최소화를 동시에 만족시키고자 하는 다양한 방안이 대두되고 있다. 그 중 비교적 짧은 시간이었지만 많은 사람들의 관심 속에 연구, 실험되어 그 성능의 우수성을 인정받고 있는 것이 군집체를 형성하여 사회활동을 하는 곤충들의 다양한 행동 패턴을 연구하여 실제 상황에 적용하는 방법이다. 곤충들의 군집 특성을 간단히 정리하면 개체의 집단행동으로 말할 수 있다. 개체들은 그룹을 만들고, 마치 지능이 뛰어난 것처럼 둥지를 만들거나 먹이를 찾는 복잡한 업무를 수행한다. 각각의 개체는 비록 단순한 행위만을 하지만 이와 같은 일련의 작은 행위들이 모여서 비교적 크고 복잡한 문제를 해결하는 것이다. 군집을 형성하는 곤충 중에서도

* 정회원, 연세대학교 전기전자공학과
(Yonsei University, Electrical and Electronic Engineering)
※ 본 논문은 2003년도 두뇌한국21사업에 의하여 지원되었음
접수일자 : 2003년12월29일, 수정완료일 : 2004년2월27일

지구상에서 가장 많은 개체수를 가지고 있고, 군집의 역사 또한 오래된 개미의 행동 패턴을 분석하여 현존하는 여러 상황에 적용하여 최적화된 방법을 찾는 것이 개미 군집 최적화(Ant Colony Optimization : ACO) 이론이다 [1-2]. ACO는 개미의 여러 특성 중 주로 페로몬을 이용한 먹이 찾기 특성(Foraging)을 모델링하여 세일즈맨의 이동 문제(Travelling Salesman Problem)같은 상황에 대한 성공적인 해결 방법을 제시하였다[3-4]. 통신 네트워크는 지금까지 ACO가 가장 성공적으로 적용된 응용 분야이다. 네트워크 노드는 개미의 둥지이며 트래픽은 개미로 모델링 되어 개미의 행동 패턴이 거의 완벽하게 그대로 실제 네트워크 환경에 적용되기 때문이다. 따라서 ACO 메카니즘을 적용한 네트워크 환경은 예측 불가능한 상황에 대한 적응성 및 안정성, 자가 조직 기능을 통한 분산 처리 능력 등 차세대 네트워크에서 요구되어지는 특성을 갖게 된다. Schoonderwoerd는 ACO를 응용한 회선교환 방식 네트워크에서의 라우팅과 로드 밸런싱 기법을 제안하였고[5]. Di Caro와 Dorigo는 패킷 방식 네트워크 환경에서 AntNet이라는 적응적 라우팅 기법을 개발하였다[6]. AntNet은 Ant라고 정의한 모바일 에이젼트를 이용하여 가변적인 네트워크 환경에서 상대적으로 양호한 라우팅 경로를 스스로 찾는 알고리즘이다. 여기서 Ant는 개미 군집체의 먹이 찾기 습성을 모델링한 네트워크 패킷이다. 개미의 먹이 찾기 습성은 개미가 먹이를 발견하고 먹이가 놓인 위치에 이르기까지 움직인 경로에 페로몬을 분비하고, 이후에 다른 개미들이 페로몬 분비양이 많은 곳으로 움직이게 하여 자연스럽게 많은 개미들이 다니는 최적의 경로를 제공하는 방식이다. AntNet에서는 Ant 패킷을 사용하여 여러 라우팅 경로에 대한 정보를 수집, 각 경로에 대한 상대적인 평가를 한다. 이 결과를 기준으로 각 노드에서는 일반 네트워크 데이터를 상대적으로 우수한 네트워크 경로로 포워딩 한다. AntNet이 기존의 여러 적응적 라우팅 기법과 비교하여 우수하다는 실험 결과는 지속적으로 발표되고 있다[6-8]. 그러나, AntNet의 성능은 결국 Ant 패킷에서 얻어지는 네트워크 정보의 정확성에 의해 결정되므로 실제 적용을 위해서는 이러한 정보의 정확성을 높이는 것이 중요하다. Ant가 수집하는 네트워크 정보 중에서 가장 중요한 것은 라우팅 시간이다. 따라서 라우팅 시간의 정확성을 향상시키기 위해 각 노드에서 발생하는 Ant 패킷 처리 속도를 최소화하여 오로지 노드와 노드사이의 라우팅 시간 정보만을 포함하도록 하며, 또한 노드별 처리 속도를 균일화하여 라우팅 시간의 오차를 줄이는 것이 중요하다. 또한 이러한 라우팅 정보를 신속하게 적용하여 일반 패킷의 쓰루풋(Throughput)을 최대화하는 것 또한 중요하다. 따라서 Ant 패킷은 패킷 처리 디바이스에서 하드웨어적으로 처리하여 그 속도 및 시간을 최소화, 균일화하고 정보의 갱신 역시 같은 방식으로 하는 것이 그 효과를 극대화할 수 있을 것이다. 그러나 기존의 상용 네트워크 프로세서와 같은 패킷 처리 디바이스는 일반 네트워크 데이타를 기준으로 설계되었기 때문에 연산부가 상대적으로 간단하며, 복잡한 연산의 경우 대부분 내장된 프로세서를 이용, 소프트웨어적으로 처리한다. Ant는 처리 속도에 민감한 패킷이지만 Ant 정보의 처리를 위해서는 상대적으로 많은 연산량이 필요하여 별도의 연산부가 필요하다. 그러므로 Ant 처리부는 명령어 레벨의 범용 하드웨어를 사용하기 보다는 Ant 처리만을 위한 별도의 로직을 설계하는 것이 효율적이다. 본 논문에서는 AntNet을 시스템 온 칩 구조에 적용하기 위한 하드웨어 구조를 제안한다. 제안된 하드웨어 구조는 Ant 처리 시간의 최소화와 균일화를 목적으로 하여 하드웨어 설계에 적합하게 변형된 기존 AntNet 알고리즘을 사용하였다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 AntNet 알고리즘을 소개하며, 제안된 AntNet 하드웨어의 전체 구조를 3장에서 설명한다. 4장에서는 하드웨어 구현에 적합하게 변형된 AntNet 알고리즘을 기존 AntNet과 다양한 네트워크 환경 하에서 비교, 평가한 실험 결과와 하드웨어 구현 결과를 소개한다. 끝으로 5장에서는 본 논문의 결론을 언급한다.

Ⅱ. AntNet

AntNet은 네트워크 환경을 조사하기 위해 Ant라 불리우는 에이전트 패킷을 사용한다. Ant는 크게 순방향(Forward) Ant와 역방향(Backward) Ant가 있다. 순방향 Ant는 일반 네트워크 데이터와 같은 우선순위를 가지며 최초 생성된 노드에서 목적지 노드까지 움직이면서 라우팅 정보를 각 노드에 저장한다. 순방향 Ant는 목적지에 도달하면 역방향 Ant가 된다. 역방향 Ant는 순방향 Ant가 지나온 길을 되돌아가면서 각 노드에 저장된 라우팅 정보를 이용하여 일반 네트워크 데이터를 위한 최선의 라우팅 경로에 관한 정보를 부여한다.

각 노드에서는 Ant의 라우팅 정보를 저장하기 위해 2종류의 저장 장소를 가지고 있다. 첫째는 로컬 트래픽

모델(Local Traffic Model)로서 네트워크 상황에 대한 통계 데이터를 저장하고 있다. 두번째는 라우팅 테이블(Routing Table)로서 현 노드에서 특정 목적지 노드로 가기위해 지나가야 하는 이웃 노드별 라우팅 선호도를 저장하고 있다. 선호도는 0부터 1까지의 정규화된 확률로 표현되며, 목적지 노드별 이웃 노드 선호도의 총 합은 1이다. 선호도가 높을수록 Ant가 해당 목적지 노드로 가기 위해 그 이웃 노드를 선택할 확률이 높아진다. 이웃 노드의 최종 결정은 선호도와 해당 이웃 노드로 가기위해 통과해야 하는 출력 큐의 길이를 참조하여 이루어진다. P'_{nd}를 목적지 노드 d로 가기 위하여 이웃 노드 n을 선택할 확률 값을 나타낸다고 가정하자. P'_{nd}는 라우팅 테이블에 저장되어 있는 이웃 노드 n에 대한 선호도 P_{nd}와 이웃 노드 n으로 가기 위한 출력 포트의 큐의 길이 l_n의 합으로 결정되며, l_n은 선험적인 수치로서 가중치 α값으로 그 비중을 결정한다. 결론적으로 $|N|$를 현재 노드에 연결되어 있는 이웃 노드의 총 수라고 할 때 P'_{nd}는 다음과 같이 표현할 수 있다.

$$P'_{nd} = \frac{P_{nd} + \alpha l_n}{1 + \alpha(|N| - 1)} \quad (1)$$

라우팅 테이블의 선호도는 역방향 Ant에 의해 갱신된다. 선호도의 갱신 폭은 강화값, r, 에 의해 결정되는데 강화값은 라우팅 정보들의 상대적인 우수함을 나타내는 파라미터이다. W_{best}는 순방향 Ant가 출발지 노드에서 목적지 노드에 이르기까지의 유효 기간 내 최단 라우팅 시간을 나타내며, T는 현재의 라우팅 시간을 가리킨다. I_{sup}와 I_{inf}는 평균 라우팅 시간 μ의 신뢰도를 나타내는 근사화된 파라미터들이다. c_1과 c_2를 각각의 가중치라 할 때 강화값은 다음과 같다.

$$r = c_1\left(\frac{W_{best}}{T}\right) + c_2\left(\frac{I_{sup} - I_{inf}}{(I_{sup} - I_{inf}) + (T - I_{inf})}\right) \quad (2)$$

강화값의 결정은 AntNet의 성능을 향상시킬 수 있는 중요한 요소이며 라우팅 정보의 상대적인 퀄리티를 제대로 평가하기 위해서는 강화값을 가변하는 방식이 필요하다. 최적의 강화값을 결정하는 방법은 지금도 많은 논문에서 끊임없이 제안되고 있다 [9]. 식 (2)에서 얻어진 강화값을 이용하여 현 노드의 라우팅 테이블 값들은 식 (3)과 같이 갱신된다.

$$\begin{aligned} &P_{fd} \leftarrow P_{fd} + r(1 - P_{fd}) \\ &P_{nd} \leftarrow P_{nd} - rP_{nd} \\ &r \in (0,1), n, f \in N, n \neq f \end{aligned} \quad (3)$$

식 (3)에서 강화값 r은 0에서 1보다 작은 값을 가지며 N은 현 노드에 연결된 모든 이웃 노드의 집합이다. 그리고 f는 현재의 Ant가 목적지에 가기위해 지나갔던 이웃 노드이며 n은 그 외 나머지 이웃 노드를 가리킨다. P_{fd}는 목적지 노드 d를 이웃 노드 f를 거쳐 갔을 경우의 라우팅 선호도이며, P_{nd}는 같은 목적지를 다른 이웃 노드를 이용했을 경우의 선호도이다. 따라서 식 (3)을 보면 현재의 역방향 Ant에 의해 지금의 Ant가 순방향으로 거쳐 갔던 이웃 노드 f에 대한 라우팅 선호도는 강화값에 비례하여 증가하고 f를 제외한 나머지 이웃 노드들을 거쳐 목적지 노드로 가는 라우팅 선호도는 역으로 감소하게 된다. 이러한 방식으로 현 노드에서 최종 목적지 노드로 가기 위해 기존의 Ant가 많이 선택했으며 또한 라우팅 퀄리티가 뛰어난 이웃 노드를 다음 경로로 선택한다.

III. 제안하는 하드웨어 구조

제안하는 하드웨어 구조는 2장에서 설명한 AntNet을 성능 열화 없이 하드웨어 부담을 최소로 할 수 있게 변환한 알고리즘을 적용하였다. 하드웨어 구조 설명에 앞서 먼저 제안된 구조에 최적화된 Ant 패킷 데이터를 정의한다.

1. Ant 패킷의 구조

The Total Size : 160 Bytes

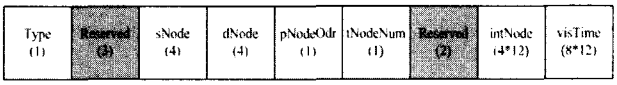


그림 1. Ant 패킷의 구조
Fig. 1. Ant Packet Structure.

제안된 구조에 최적화된 Ant의 구조는 그림 1과 같다. 하나의 Ant는 160바이트의 크기를 가지며 QoS 서비스와 같이 추가되는 정보의 종류에 따라 그 길이는 가변될 수 있다. 현재까지 정의된 Ant는 라우팅 시간 정보만을 가지고 있다. 그림 1에서 나타낸 Ant의 세부 설명은 다음과 같다

- *Type* : 순방향 Ant인지 역방향 Ant인지를 표시
- *sNode* : 현 Ant가 최초 생성된 노드의 IP 주소
- *dNode* : 현 Ant가 향하는 최종 목적지의 IP 주소
- *pNodeOdr* : 현 Ant가 출발지에서 목적지까지 이동하면서 거쳐간 노드의 순서
- *tNodeNum* : 현 Ant가 출발지에서 목적지까지 이동하면서 거쳐간 노드의 전체 수

- *intNode* : 현 Ant가 출발지에서 목적지까지 이동하면서 거쳐간 노드의 IP 주소
- *visTime* : 현 Ant가 각 노드에 방문한 시간

효율적인 하드웨어 설계를 위해 Ant의 길이는 고정되었다. 따라서 하나의 Ant가 방문할 수 있는 노드의 총 수는 12개로 한정되었다. 그림 1의 Ant 구조에서 *sNode*와 *dNode*를 제외한 나머지 데이터는 Ant의 이동과정 중 각 노드에서 자동적으로 입력되며 *sNode*와 *dNode*는 최초 Ant를 생성한 노드에서 결정된다. *dNode*의 경우 사용자가 직접 결정하는 사용자 지정 모드와 노드를 거쳐 가는 일반 데이터 패킷의 목적지를 랜덤 추출하여 사용하는 자동 지정 모드가 있다. Ant의 라우팅 시간 정보는 *visTime*에 저장되는데 *visTime*은 SNTP(Simple Network Time Protocol)와 같이 현존하는 네트워크 시간 프로토콜을 사용한다. 현재 구조는 SNTP Ver. 4 [10]를 기준으로 하여 노드 당 시간 정보는 64비트 크기로 구성되고 200 ps의 해상도를 가진다.

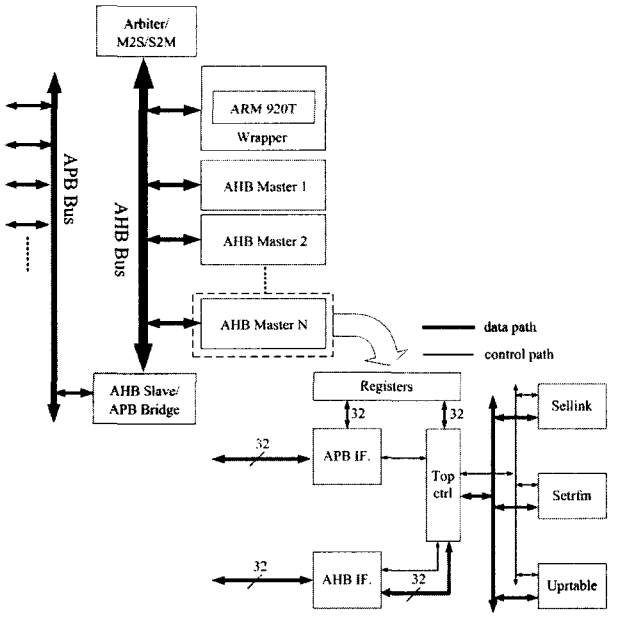


그림 2. 전체 하드웨어 구조
Fig. 2. Top-Block Diagram.

2. 전체 하드웨어 구조

전체 구조는 그림 2에서 나타내었다. 3-1장에서 설명한 Ant 패킷을 처리하기 위해 설계된 구조로 외부 연결 블록을 제외하고 총 4개의 주요 Ant 처리 블록으로 구성되어 있다. 제안된 구조는 ARM 920T [11-12]를 메인 프로세서로 하는 시스템 온 칩 구조를 목적으로 설계되었으며, AMBA 버스 Ver. 2 [13]를 기준으로 한다. 또한 상기 버스의 주(Master) 블록이며 AHB 버스에 연결되어 있다. 따라서 모든 주소와 데이터의 폭은 32비트이다. 사용자는 레지스터 제어를 통해 세부 기능을 통제할 수 있으며, 레지스터 제어는 APB 인터페이스 기능을 지원하는 UART와 같은 콘트롤러를 사용하면 된다 [14]. 세부 블록별 설명은 다음과 같다.

A. 라우팅 테이블과 트래픽 모델

기본 구조는 2장에서 설명한 라우팅 테이블과 트래픽 모델 구조와 동일하다. 예를 들어 현 노드에 연결된 이웃 노드의 수가 10개이며 목적지 노드의 수를 20개로 가정하면 라우팅 테이블은 (10, 20)의 2차원 구조로 구성되며, 트래픽 모델은 목적지 노드에 따라 분리되므로 20개의 목적지 노드별로 데이터를 가지게 된다. 트래픽 모델에 저장되는 데이타는 기존 AntNet의 경우 라우팅 최단 시간, 평균 라우팅 시간, 라우팅 시간의 분산도 등의 정보를 저장하지만 제안된 구조에서는 유효 시간 내 라우팅 시간의 최저값만을 저장한다. 그리고 하드웨어 연산량 및 구조를 단순하게 하기 위해 라우팅 테이블에 저장되는 선호도를 1바이트로 설정하여 0부터 255까지의 정수값으로 저장한다.

B. Sellink

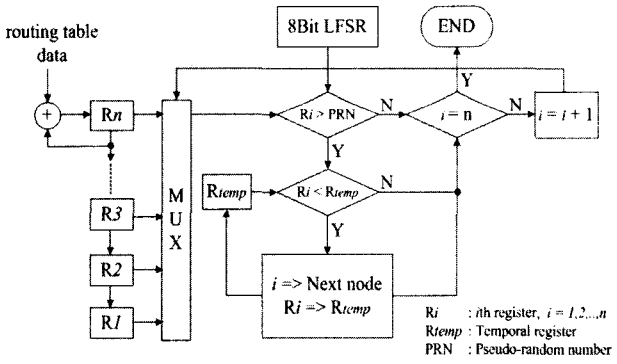


그림 3. 이웃 노드의 선택
Fig. 3. Selection of a Next Node.

Sellink는 현 노드에서 이동할 이웃 노드를 선택하는 블록으로서 현재는 라우팅 테이블에 있는 선호도만을 참조하여 결정한다. 결정하는 순서는 먼저 외부 메모리에 저장되어 있는 라우팅 테이블 중에서 현재 Ant와 같은 목적지 노드를 갖는 이웃 노드별 선호도 값들을 가져온다. 가져온 값들은 미리 정해진 노드별 순서대로 내부 레지스터에 누적된다. 따라서 현 노드에 연결된 이웃 노드가 10개이면 필요한 내부 레지스터의 수 또한 10개가 된다. 같은 목적지를 갖는 이웃 노드들의 라우팅 선호도 값의 총 합은 255이므로 각각의 내부 레지스

터 크기는 1바이트면 된다. 내부 레지스터들은 1부터 255까지의 유사 랜덤 값을 생성하는 LFSR (Linear Feedback Shift Register)에 의해 만들어진 값과 순차적으로 비교된다. 모든 내부 레지스터 값들과 비교가 끝나면 몇 번째 노드가 다음 노드가 되는지 결정되고, 선택된 다음 노드는 레지스터에 미리 정의되어 있는 노드 순서별 IP 주소 테이블을 참조하여 Ant가 이동할 실제 IP 주소로 변경된다. 블록의 상세 동작을 그림 3에서 나타내었다.

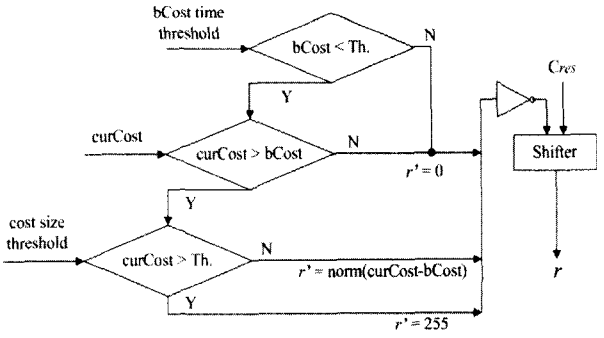


그림 4. 강화값의 계산 과정
Fig. 4. Calculation Flow of a Reinforcement Value.

C. Setrfm

강화값은 라우팅 정보의 퀄리티를 계산하여 라우팅 테이블에 저장되어 있는 값들의 갱신 크기를 결정하는 파라미터이다. 기존 AntNet에서는 강화값을 결정하기 위해 여러 요소들을 도입했지만 제안된 구조에서는 Ant의 라우팅 시간만을 고려한다. *bCost*를 Ant에 의해 결정된 이웃 노드 n에서 목적지 노드 d까지의 가장 짧은 라우팅 시간이라 하고 *curCost*를 현재의 라우팅 시간으로 할 때, 이 두 값의 차이를 1바이트 크기로 정규화한 것을 r'라 한다. r'은 단순히 두 값의 절대적 차이만을 의미하므로 실제 강화값은 라우팅 시간의 퀄리티에 따라 가중치 C_{res}를 이용하여 결정되며 이를 정리하면 다음과 같다.

$$r' = norm\,(curCost - bCost) \tag{4}$$

$$r = (255 - r')/C_{res} \tag{5}$$

식 (5)의 C_{res}는 식 (4)에서 사용된 *bCost*가 증가할수록 감소하며, *bCost*가 감소하면 반대로 증가한다. 최적의 C_{res}는 네트워크 환경에 따라 가변적이므로 제안된 구조에서는 C_{res}를 파라미터화 하여 외부에서 입력할 수 있게 설계하였으며, 또한 하드웨어 부담을 고려하여 C_{res} 동작을 쉬프터로 구현하였다. *bCost*는 일정 유효 시간이 지나면 다시 초기화되는데 이것은 최신 라우팅 정보를 이용하여 강화값의 신뢰도를 높이기 위함이다. 또한 연산 시간과 하드웨어 크기를 줄이기 위해 *curCost* 크기의 제한을 적용한다. 강화값의 계산 과정은 그림 4에서 나타내었다.

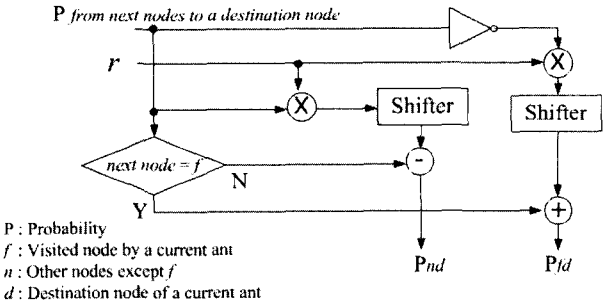


그림 5. 라우팅 테이블 값의 갱신
Fig. 5. Updating Procedure of a Routing Table.

D. Uprtable

라우팅 테이블에 저장된 값들은 강화값과 수집된 라우팅 정보에 의해 Uprtable 블록에서 갱신된다. 갱신 하는 과정은 강화값의 범위를 0에서 1사이의 값으로 환산하는 부분을 제외하고는 기존의 AntNet과 동일하다. 식 (6)과 그림 5에서 라우팅 테이블에 저장된 이웃 노드별 선호도를 갱신하는 식과 과정을 나타내었다.

$$\begin{aligned} &P_{fd} \leftarrow P_{fd} + (r(255 - P_{fd})/256) \\ &P_{nd} \leftarrow P_{nd} - (rP_{nd}/256) \\ &n, f \in N,\ n \neq f \end{aligned} \tag{6}$$

E. Topctrl

Topctrl 블록은 세부 기능 블록들과 외부 연결을 위한 로직들을 제어하고, 도착한 Ant 패킷들을 분석하는 역할을 한다. 제어 과정은 다음의 5단계 상태도로 구성된다.

- Start : Ant 기능이 시작되면 순방향 Ant를 생성하여 저장된 라우팅 선호도를 기준으로 다음 이웃 노드로 전송한다
- Forwarding : 현 노드가 목적지 노드가 아닌 경우 Ant를 계속 순방향으로 보내는 단계로 노드를 지날 때마다 *pNodeOdr*과 *tNodeNum*이 하나씩 증가하며, 방문한 노드의 IP 주소와 시간 정보도 함께 Ant에 저장된다
- Destination : 현재의 노드가 목적지 노드인 경우 일단 현 노드에 대한 정보를 Ant에 추가한 후 역방

향 Ant로 변환하여 바로 이전에 방문했던 노드로 되돌아간다

- Backwarding : 출발지 노드에 도달할 때까지 순방향 Ant가 지나왔던 노드들을 거슬러 올라가면서 가지고 있는 라우팅 정보를 이용하여 각 노드에 저장된 라우팅 선호도와 트래픽 모델 데이터를 갱신한다. 하나의 노드를 지날 때마다 *pNodeOdr* 값을 하나씩 감소시킨다
- Source : 출발지 노드에 도달하면 노드의 라우팅 선호도와 트래픽 모델 데이터를 갱신하고 소멸된다

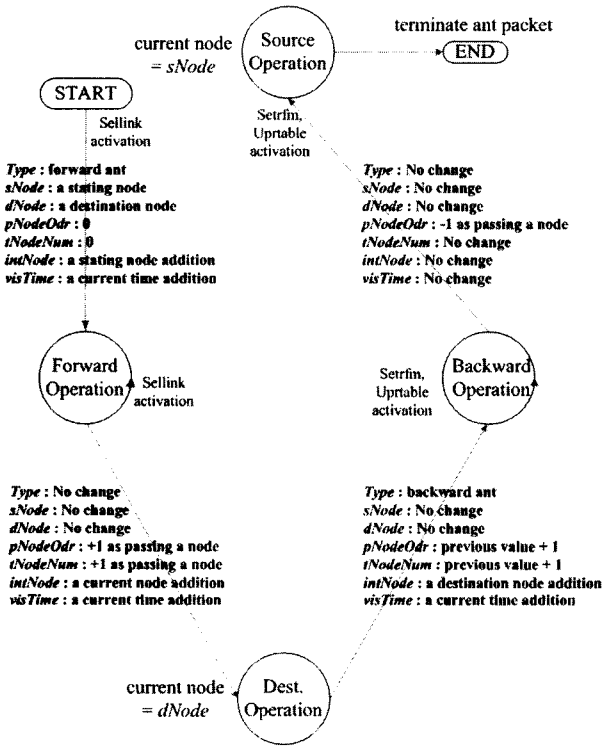
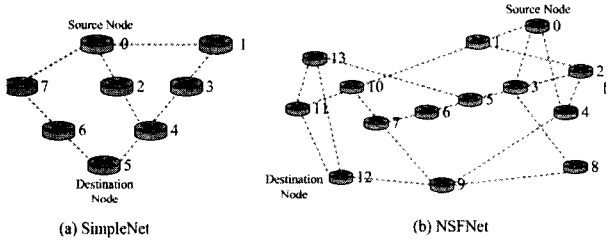


그림 6. Topctrl 블록의 상태도
Fig. 6. FSM of Topctrl.

순방향 라우팅 과정 중 각 노드에서 단계를 결정하기 전에 항상 검토해야할 2가지 포인트가 있다. 첫번째는 순환(Circle) 발생 여부이다. 순환이란 출발지에서 목적지 노드로 이동하는 도중 이미 방문했던 노드에 다시 들리는 것을 의미한다. 순환은 Ant로 하여금 잘못된 라우팅 정보를 저장하게 하므로 반드시 제거되어야 한다. 순환의 발생 여부는 Ant가 저장하고 있는 정보 중 *intNode*를 검사하여 현 노드 IP 주소와의 중복 여부를 조사하면 된다. 두번째 검토 요소는 Ant가 방문한 노드의 총 수이다. 3-1장에서 정의된 Ant는 최대 12개의 노드 정보까지 저장할 수 있다. 따라서 목적지 노드에 도달하기 전에 최대값에 이르게 되면 그 때의 노드를 목적지 노드로 판단하여 역방향 Ant로 변환한다. 네트워크 안에 여러 노드에서 주기적으로 Ant가 발생되므로 이처럼 일부 Ant의 소실 및 라우팅 중단으로 인한 성능 열화는 미미할 것으로 판단된다. Topctrl의 제어 상태도는 그림 6과 같다.

VI. 성능 평가 및 실험 결과

하드웨어 구현을 위해 수정된 AntNet 알고리즘은 기존의 AntNet과 다양한 네트워크 환경 하에서 비교 실험되어 그 성능을 평가하였다. 실험 환경으로 사용된 네트워크 구조는 그림 7에서 나타내었다. 그림 7의 구조는 기존 AntNet의 성능 검증을 위해 사용된 여러 네트워크 구조 중 일부이다. SimpleNet은 8개의 네트워크 노드와 9개의 양방향 링크로 구성된 네트워크 구조이며, NSFNet은 14개의 네트워크 노드와 21개의 양방향 링크로 구성된 구조이다. 노드에 매겨진 숫자는 노드 식별자이다.



(a) SimpleNet (b) NSFNet

그림 7. 실험용 네트워크 노드 구조
Fig. 7. Test Topology.

실험의 평가 기준은 각 노드의 라우팅 테이블 내 라우팅 선호도의 변화 추이며, 그 중에서도 출발지 노드에서의 갱신 횟수에 비례한 선호도 변화를 관측하였다. 여기서 갱신 횟수란 선호도의 갱신 횟수 혹은, 해당 노드를 방문한 Ant의 수를 의미한다. 비교 실험을 위해 다음과 같은 4가지의 네트워크 상황을 가정하였다.

- Normal : 네트워크 내 모든 트래픽이 균일하게 분포되어 각 노드사이의 라우팅 시간이 거의 차이가 없는 상태
- Biasing : 특정 노드들 사이의 트래픽 양이 다른 곳보다 작아서 소요되는 라우팅 시간이 현저하게 작은 상태
- Heavy : 모든 트래픽이 고루 분포되어 있는 것은 normal 상태와 유사하지만 전체적인 트래픽 양이

매우 많아서 노드간 평균 라우팅 시간이 normal과 비교하여 최소 10배 이상 더 늘어난 상태

- Dynamic : 주기적으로 네트워크 트래픽 패턴이 변하는 상태. 실험에서는 normal, biasing, heavy 상태가 일정한 주기로 반복되는 경우를 가정하였다

실험은 normal, biasing, heavy 상태와 같이 고정된 트래픽 패턴의 경우 출발지 노드의 라우팅 선호도 중 어느 하나가 다른 선호도에 비해 미리 정해놓은 차이 이상으로 우세해지면 마치게 하였고, dynamic 상황에서는 일정한 시간이 지나면 멈추게 하였다. Ant는 출발지 노드에서 주기적으로 발생하며, 전체 네트워크 내 노드 수를 고려하여 동시에 활성화된 Ant의 수는 30개로 제한하였다.

1. SimpleNet

SimpleNet 구조에서는 출발지 노드가 0이고 목적지 노드가 5인 경우를 가정하여 실험하였다. 이러한 경우 출발지에서 목적지 노드까지 갈 수 있는 가용 경로는 총 3가지가 있다. 하나는 노드 0-1-3-4-5이며, 두 번째로는 노드 0-2-4-5, 또 다른 하나는 노드 0-7-6-5이다. 따라서 출발지 노드 0의 라우팅 테이블 내 목적지 노드 5에 대하여 활성화된 라우팅 선호도는 P_{15}, P_{25} 그리고 P_{75} 세 가지가 있다. 이후 실험 결과는 여러 네트워크 상황 하에 상기 세 가지 라우팅 선호도의 변화도를 갱신 횟수에 비례하여 나타낸 것이다. Normal 상태에서는 노드 사이의 라우팅 시간 차이가 없으므로 목적지 노드까지의 거리, 즉 방문한 노드의 수에 따라 라우팅 선호도가 영향을 받는다. 따라서 상대적으로 라우팅 경로가 짧은 P_{25} 와 P_{75}가 P_{15}에 비해 우세해진다. P_{25}와 P_{75}사이는 랜덤하게 결정되는 두 경로의 선택 횟수에 따라 그 차이가 벌어질 수 있지만 어느 한 쪽의 선택 자체가 전체 라우팅 퀄리티에 영향을 미치지는 않는다. 그림 8에서 normal 상황에서의 실험 결과를 나타내었다. 기존 AntNet의 경우는 P_{25}와 P_{75}가 우열을 가리지 못하고 계속해서 교번하는 형태이고, 수정된 AntNet은 일정 시간 이후에 둘 중 하나가 선택되는 경향을 보인다. 이러한 차이가 발생하는 이유는 LFSR에 의해 생성되는 유사 랜덤 수에 의한 영향이 수정된 알고리즘에서 더 크기 때문이다. 실험 결과에서 선호도의 진동이 심해지면 그만큼 해당 경로 사이의 우열이 뚜렷하지 못해 선택의 변화가 많았다는 것을 의미한다. 하지만 기존 AntNet이나 수정된 AntNet 모두 P_{25}와 P_{75}가 P_{15}에 비해 우세함을 알 수 있다. Biasing 모드에서는 경로 0-1-3-4-5의 라우팅 시간이 다른 2개의 경로보다 작은 경우를 가정하였다. 이러한 경우 2개의 알고리즘 모두 normal 모드와 달리 상대적으로 진동 없이 작은 갱신 횟수에서 P_{15}가 확연한 우위를 보여준다. 실험 결과는 그림 9에서 나타내었다. 전체 네트워크 트래픽이 현저하게 증가한 heavy 상태에서는 각 경로의 차이도 같은 비율로 증가하지 않는 이상 경로별 차이가 normal에 비해 더욱 더 줄어들게 된다. 따라서 그림 10을 보면 기존 AntNet의 경우 진동의 횟수가 증가하며 그 폭은 줄어들었다. 수정된 AntNet은 식 (5)의 C_{res}를 변경하지 않는 한 기존 AntNet과 마찬가지로 어느 한 쪽의 우세가 확실하게 고정되기 보다는 계속된 진동이 발생한다. 그럼에도 불구하고 normal과 마찬가지로 P_{25}와 P_{75}가 P_{15}에 비해 우세한 경향은 그대로 유지된다. 마지막으로 가변하는 네트워크 환경을 만들기 위해 라우팅 선호도 갱신 횟수 500회를 기준으로 네트워크 환경을 normal, biasing, heavy 모드 순서대로 변경하였다. 기존 AntNet은 적합한 경로를 선택하기 위해 고려해야 하는 요소들이 많으므로 새로운 환경에 적응하기 위해서는 많은 시간을 필요로 한다. 그러나 수정된 알고리즘은 Ant의 라우팅 시간만을 고려하는 비교적 간단한 구조여서 트래픽 변화에 대한 적응 시간이 매우 짧다. 그림 11의 결과에서 볼 수 있듯이 기존 AntNet의 전체적인 라우팅 경로 선택 경향은 고정된 트래픽 패턴의 경우와 유사하나 트래픽 패턴이 바뀌면 다시 새로운 경로를 선택하기까지는 상당한 시간이 필요하고, 진동 또한 크게 증가함을 확인할 수 있다. 그에 비해 수정된 AntNet은 트래픽 변화에 대한 적응력이 우수함을 볼 수 있다.

2. NSFNet

NSFNet에서는 출발지 노드가 0이며 목적지 노드가 12인 경우를 가정하였다. NSFNet은 중간에 분기되는 노드가 많기 때문에 출발지에서 목적지 노드에 이르기 위한 경로 선택 방법 또한 SimpleNet에 비해 크게 증가한다. 본 실험에서의 비교 대상은 출발지 노드에서 목적지 노드 12에 대한 라우팅 선호도 $P_{1,12}$, $P_{3,12}$, 그리고 $P_{4,12}$의 변화도이다. 트래픽이 고루 분포되어있는 normal과 heavy 모드에서는 목적지 노드까지 이르는 최단 거리, 경로 0-4-9-12,일 경우 가장 라우팅 효과가 좋아지므로 $P_{4,12}$가 기존 AntNet과 수정된 AntNet 모두에서 가장 우세해진다. 다만 최단 거리 경로가 대부분의 다른 경로에 비해 아주 짧으므로 우열의 차이가 명확하여 수정된 AntNet의 경우 큰 진동 없이 아주 빠르게 수렴

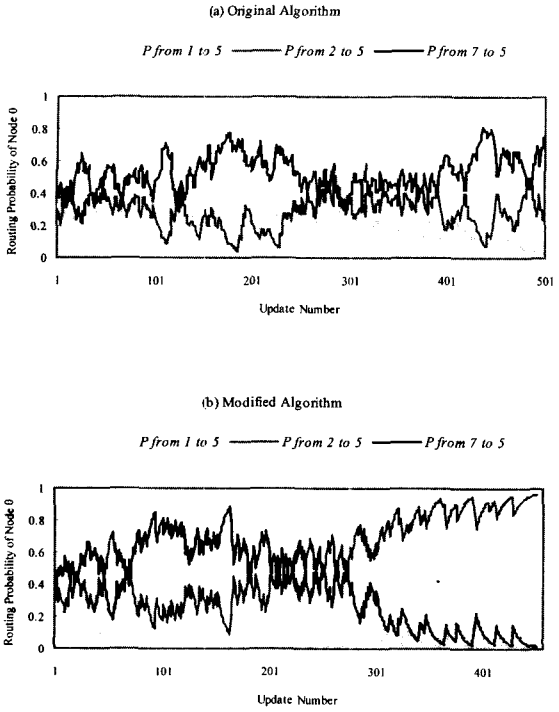


그림 8. SimpleNet 실험 결과 : Normal 상태
Fig. 8. SimpleNet Simulation Result : Normal State.

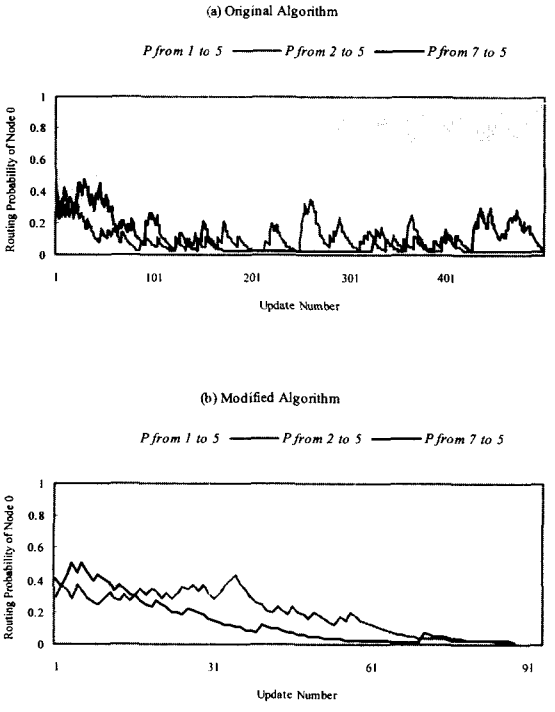


그림 9. SimpleNet 실험 결과 : Biasing 상태
Fig. 9. SimpleNet Simulation Result : Biasing State.

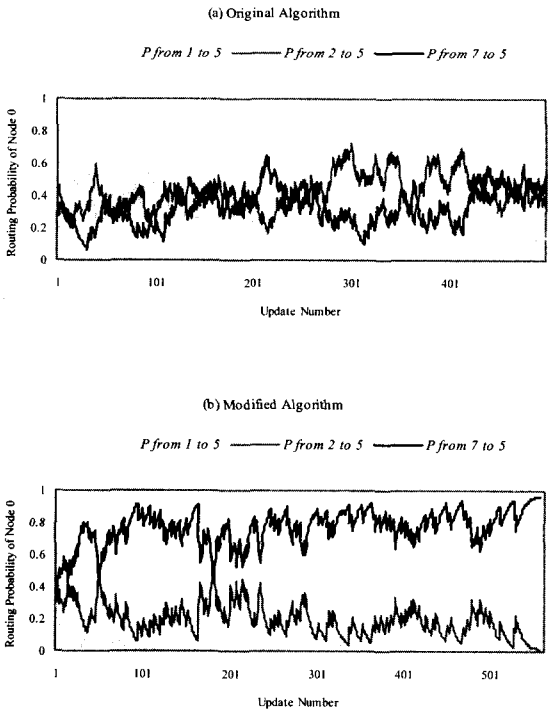


그림 10. SimpleNet 실험 결과 : Heavy상태
Fig. 10. SimpleNet Simulation Result : Heavy State.

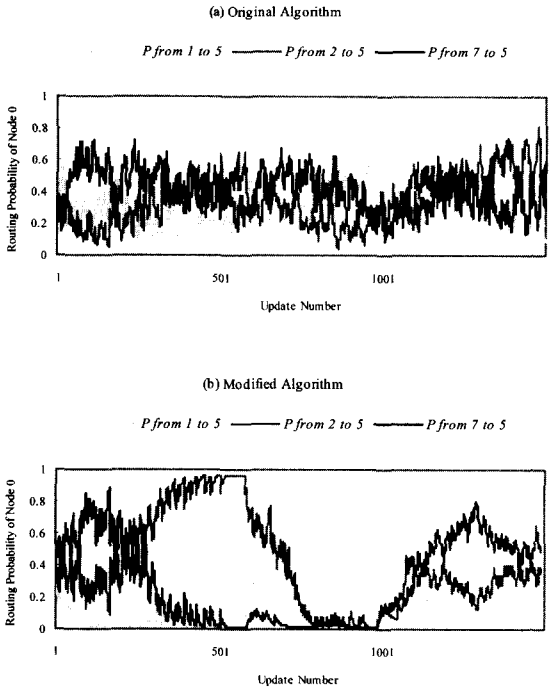


그림 11. SimpleNet 실험 결과 : Dynamic 상태
Fig. 11. SimpleNet Simulation Result : Dynamic State.

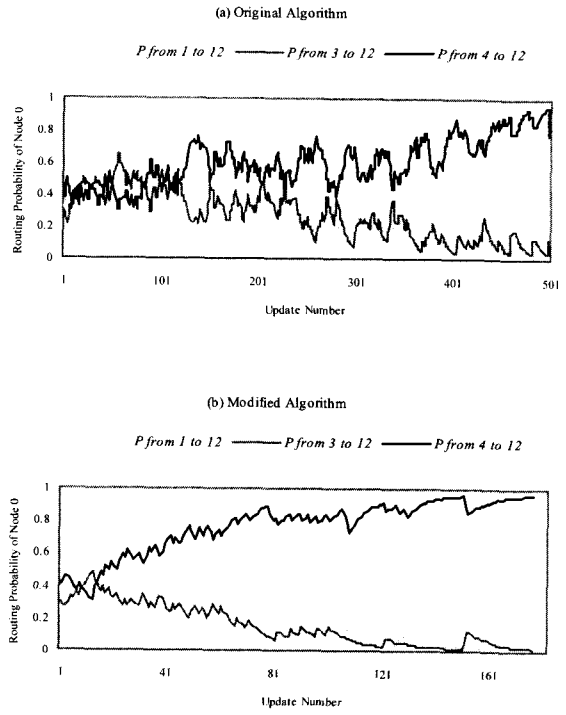


그림 12. NSFNet 실험 결과 : Normal 상태
Fig. 12. NSFNet Simulation Result : Normal State.

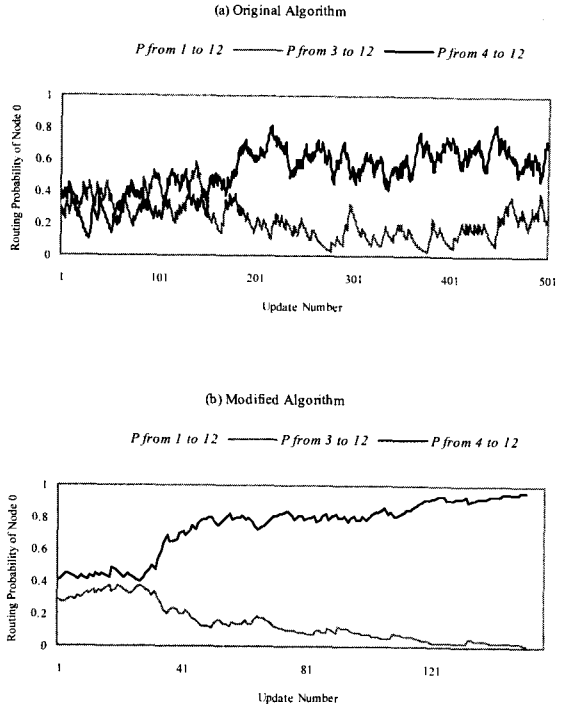


그림 14. NSFNet 실험 결과 : Heavy 상태
Fig. 14. NSFNet Simulation Result : Heavy State.

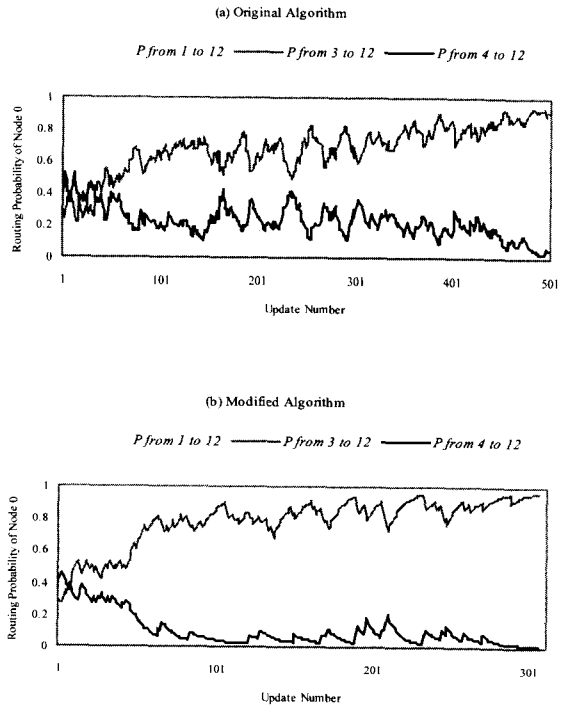


그림 13. NSFNet 실험 결과 : Biasing 상태
Fig. 13. NSFNet Simulation Result : Biasing State.

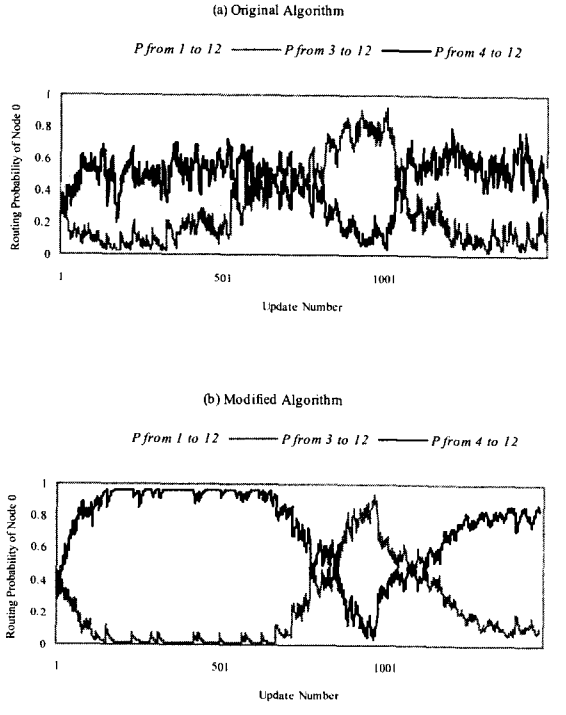


그림 15. NSFNet 실험 결과 : Dynamic 상태
Fig. 15. NSFNet Simulation Result : Dynamic State.

한다. Biasing 모드에서는 경로 0-3-5-6-7- 9-12가 가장 라우팅 시간이 작아지도록 변경했다. 이 경우에서도 $P_{3,12}$의 선호도가 비록 거쳐야하는 노드 수는 많아졌지만 비교적 이른 시간 내 수렴하였다. 단 분기점이 많은 관계로 경로당 Ant의 라우팅 시간 차이가 크지 않으면 기존의 AntNet이나 수정된 AntNet 모두 수렴 시간이 증가하며, 라우팅 시간 차이가 별로 없는 차선의 라우팅 경로를 선택할 수 있다. Dynamic 모드에서도 SimpleNet에서의 실험 결과와 유사한 현상을 보였으며 라우팅 시간의 차이가 확실한 만큼 선호도의 변화 추이가 네트워크 환경에 따라 좀 더 명확한 경향을 보였다. NSFNet의 실험결과는 그림 12 ~ 15에서 나타내었다.

상기 실험을 통하여 수정된 AntNet이 네트워크 환경이나 크기, 구조에 관계없이 기존 AntNet과 유사한 성능을 가질 수 있음을 확인하였다. 수정된 AntNet 알고리즘은 3장에서 설명한 하드웨어 구조 형태의 RTL 코드로 구현되었다. RTL 코드는 verilog HDL을 사용하였으며, 멘토사의 하드웨어/소프트웨어 통합 검증 환경인 Seamless를 사용하여 검증되었다[15]. Ant의 처리 시간은 실제 연산 시간보다 AHB 버스 사용권 획득 시간과 외부 메모리 사용을 위한 대기 시간에 의해 크게 좌우되었으며, 처리 시간의 균일화를 위해서는 전체 시스템 내에서의 Ant 처리의 우선순위를 증가시킬 필요가 있다. TSMC 0.25μm 라이브러리를 이용한 디자인(레지스터 세트 제외) 합성 결과는 100MHz 시스템 클럭 기준 약 60K 게이트 크기이며, 사용된 외부 메모리의 크기는 $(m*n+4*n)$ 바이트이다. 여기서 m은 이웃 노드의 총 합계이며, n은 목적지 노드의 총 합계이다.

V. 결 론

본 논문에서는 개미 군집체의 행동 패턴을 실제 네트워크에 적용하여 가변되는 환경 하에서 적응적으로 최적의 라우팅 경로를 선택할 수 있는 AntNet 알고리즘 기반의 하드웨어 구조를 제안하였다. AntNet이 네트워크 환경 정보를 수집하기 위해 사용되는 Ant는 반드시 하드웨어 레벨로 처리, 그 시간을 최소화, 균일화해야 라우팅 정보의 신뢰도를 증가시켜 실제 환경에서의 정확한 동작을 유도할 수 있다. 이에 제안된 하드웨어 구조는 AntNet 프로토콜을 지원하는 네트워크 프로세서 같은 네트워크 디바이스에 적용되어 미래 네트워크 환경에 적합한 기능을 제공할 수 있을 것으로 예상된다.

하드웨어 구조에 적합하게 변형된 AntNet 알고리즘은 기존 AntNet과 동일한 가상 네트워크 환경에서 그 성능을 비교하여 대부분의 경우 유사한 성능을 보임을 확인할 수 있었다. 그리고 제안된 구조의 하드웨어 설계 및 합성 결과는 본 구조가 AntNet 기반의 라우팅 기능을 실제 SoC 형태의 시스템으로 구현하는데 효과적임을 보여준다. 또한 AntNet의 일부 기능, 예를 들면 강화값의 계산,은 아직도 많은 연구가 진행되고 있기 때문에 하드웨어 설계도 많은 부분을 외부 파라미터에 의해 조정 가능하게 구현하여 차후 기능 확장 및 실험이 가능하게 하였다.

참 고 문 헌

[1] M. Dorigo, G. Di Caro and L. M. Gambardella, "Ant Algorithms for Discrete Optimization," *Artificial Life*, Vol. 5, No. 3, pp. 137-172, 1999.

[2] E. Bonabeau, M. Dorigo, and G. Theraulaz, "Inspiration for Optimization from Social Insect Behavior," *Nature*, Vol. 406, pp. 39-42, July 2000.

[3] M. Dorigo, V. Maniezzo, and A. Colorni, "The Ant System : Optimization by a Colony of Cooperating Agents," *IEEE Trans. on Systems, Man and Cybernetics-Part B*, Vol. 26, No. 1, pp. 1-13, 1996.

[4] M. Dorigo and L. M. Gambardella, "Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Trans. on Evolutionary Computation*, Vol. 1, No. 1, pp. 53-66, April 1997.

[5] R. Schoonderwored, O. Holland, J. Bruten, and L. Rothkrantz, "Ant-based Load Balancing in Telecommunications Networks," *Adaptive Behavior*, Vol. 5, No. 2, pp. 169-207, 1996.

[6] G. Di Caro and M. Dorigo, "AntNet : Distributed Stigmergetic Control for Communications Networks," *Journal of Artificial Intelligence Research 9*, pp. 317-365, December 1998.

[7] K. M. Sim and W. H. Sun, "Multiple Ant-Colony Optimization for Network Routing," *Proc. of the First International Symposium on Cyber Worlds*, pp. 277-281, November 2002.

[8] Y. Yang, A. N. Zincir-Heywood, M. I. Heywood, and S. Srinivas, "Agent-Based Routing

Algorithms on a LAN," *Proc. of the IEEE Canadian Conference on Electrical & Computer Eng.*, Vol. 3, pp. 1442-1447, May 2002.

[9] M. Dorigo, M. Zlochin, N. Meuleau, and M. Birattari, "Updating ACO Pheromones using Stochastic Gradient Ascent and Cross-Entropy Methods," *Proc. of the EvoWorkshops 2002*, LNCS 2279, Springer, pp. 21-30, 2002.

[10] RFC-2030 : *SNTPv4 for IPv4 and IPv6 and OSI*, October 1996.

[11] ARM DDI 0151C, *ARM 920T (Rev 1) Technical Reference Manual*, ARM Limited, April 2001.

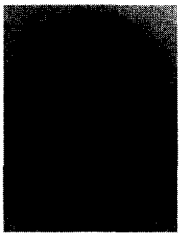
[12] S. Furber, *ARM System-on-Chip Architecture*, Addison-Wesly, Great Britain 2000.

[13] ARM IHI 0011A, *AMBA (Rev 2) Specification*, ARM Limited, 1999.

[14] ARM DDI 0183E, *Primecell UART (PL011) Technical Reference Manual*, ARM Limited, December 2001.

[15] Seamless CVE User's and Reference Manual, v4.3, Mentor Graphics Corp., 2002

저 자 소 개



안 진 호(정회원)

1995년 2월 연세대학교 전기공학과 (공학사).

1997년 2월 연세대학교 전기공학과 (공학석사).

2002년 8월 LG전자 DTV연구소 선임 연구원.

2002년 9월~현재 연세대학교 전기전자공학과 박사과정.

<주관심분야 : SoC 설계 및 응용, DFT>

오 재 석(정회원)

2001년 5월 The University of Bridgeport 컴퓨터공학과(B.S.).

2004년 2월 연세대학교 전기전자공학과(공학석사)

<주관심분야 : SoC 설계 및 응용>

강 성 호(정회원)

1986년 2월 서울대학교 제어계측공학과(공학사).

1988년 5월 The University of Texas at Austin 전기 및 컴퓨터 공학과(공학석사).

1992년 5월 The University of Texas at Austin 전기 및 컴퓨터공학과(공학박사).

1992년 미국 Schlumberger 연구원.

1994년 Motorola 선임 연구원.

1994년~현재 연세대학교 전기전자공학과 부교수.

<주관심분야 : SoC 설계 및 응용, DFT, SoC Test, CAD for SoC >