울산대학교 연구논문집 제18권(자연과학·공학편)제2호 pp.43~59. 1987.
UOU Report Vol.18(Natural Science and Engineering)No.2 pp.43~59. 1987.
University of Ulsan

통신네트워크에서의 정보전달순서에 관한 모의 실험*

고 재 문

산업공학과

(1987. 4. 30 접수)

<要 約>

본 연구에서는 통신네트워크에서 로컬브로드캐스팅 방식에 따른 정보전달 방법을 다루었다. 나무형 구조의 경우 최적기법에 대해 고찰한 다음 퍼스널 컴퓨터용 패키지를 개발하였고 일반적 구조의 경우 발견적 해법을 제시하고 전산모의실험을 하였다.

Computational Experimentation of Information Dissemination in a Communication Network

Koh, Jae-moon
Dept. of Industrial Engineering

(Received April 30, 1987)

〈Abstract〉

Information dissemination by local broadcasting in a communication network is studied. In the case of a tree-type network, an optimal technique is reviewed and a program package for a personal computer is developed. In the case of a general-type network, a heuristic algorithm is suggested and computational experimentation is performed

Ⅰ. 서 론

최근 들어 여러 분야에 걸쳐 광범위하게 커다란 영향을 미치기 시작한 데이타 통신시스템은 우리나라에서도 설치, 운용 단계에 이르렀다. 이에 대한 하드웨어의 설치 및 구성은 어느 정도의 수준에 올라 있으나, 이를 운용, 관리하는데 아직도 미비한 점이 많으며, 이런 소프트웨어 분야에 대한 심층적

*이 논문은 1986년도 문교부 학술연구조성비에 의하여 연구되었음.

인 연구가 필요하다. 그중에서도 특히 정보전달방법에 대한 연구는 통신시스템 전체의 운용효율을 높인다는 관점에서 매우 중요하다. 즉, 통신시스템에서 어떤 정보가 발생했을 때 이를 어떤 순서로 전달할 것인가에 따라 통신의 효율이 크게 좌우될 수 있다.

통신네트워크에서 정보를 전달하는 방식에는 여러 가지가 있으나 본 연구에서는 다음과 같은 방식을 다룬다.

a. 정보를 이미 전달 받은 vertex만이 다른 vertex에 정보를 전달할 수 있다.

b. 정보는 바로 이웃하는(직접 연결된) vertex에만 전달한다.

c. 한 vertex는 동시에 여러 곳으로 정보를 보낼 수 없고 또 동시에 여러 곳으로부터 정보를 받을 수도 없다. 즉 한 vertex는 한번에 한 곳으로 보내기만 하거나 또는 한번에 한 곳으로부터 받기만 할 수 있다.

d. 각 edge에서의 전달시간은 다른 edge에서의 정보전달 여부, 전달순서 등에 관계없이 일정하다.

이와 같은 전달방식을 local broadcasting이라 하며 대부분의 message switching network에서 볼 수 있다.[3] 또한 전화를 이용해서 어떤 정보를 전달할 때도 이런 방식을 관찰할 수 있다. 이와 관련된 여러 문제 중에서 본 연구에서는 모든 vertex들이 알아야 할 정보가 어떤 vertex 에서 발생했을 때, 이를 어떻게 전달할 것인가 하는 문제를 다룬다.

네트워크의 구조가 나무형인 경우에 대해서는 많은 연구가 이루어져 있다. 즉 각 edge에서 전달시간이 1인 경우에 Proskurowski[8]와 Slater[9] 등이 간단한 해법을 제시하였고, 이에 대한 확장으로서 각 edge에서의 전달시간이 각기 다르고 정보의 출발점이 주어진 경우에 대해서도 해법이 개발되어 있다.[5] 그리고 후자의 경우 전체 정보전달시간을 최소로 하는 정보의 출발점을 결정하는 해법도 제시되었다.[6] 그러나 네트워크의 구조가 일반형인 경우에 대해서는 연구가 거의 이루어져 있지 않다.

본 연구에서는 나무형 구조를 하는 네트워크의 경우 최적기법에 대해 고찰하고 이 기법의 실용화를 위해 퍼스널 컴퓨터용 전산패키지를 개발하고자 한다. 그리고 일반형 구조에 대해서는 하나의 발견적 해법을 제시하고 그 타당성 검토를 위해 전산 모의실험을 하고자 한다.

II. 나무형 구조에서의 정보 전달

T=(V, E)를 vertex들의 집합 V와 edge들의 집합 E로 이루어진 tree-type network라 하고, T의 각 edge에서의 정보전달시간이 주어졌다고 하자. 또한 어떤 정보가 있어서 앞에서 밝힌 local broadcasting 방식에 따라 가능한 한 빠른 시간 내에 T의 모든 vertex들에 전달되어야 한다고 하자. 이때 전체전달시간에 영향을 주는 것은 각 vertex에서의 전달순서가 되며 그에 대한 최적해는 각 subtree 안에서의 전달시간이 큰 순서대로 정보를 보내는 것이다.[5]

편의상 T_v 를 T에서 출발점(root)이 주어졌을 때 vertex v와그 descendant들로서 이루어진 subtree들에게 전달할 때까지의 최소시간이라 하자. 예에서 전체 정보전달시간을 최소로 하기 위해서는 —t(6)를 구하기 위해서는— 이웃하는 vertex들(3, 7, 11) 중 각 subtree 안에서의 최소 정보전달시간이 가장 큰 곳으로 먼저 정보를 보내야 한다. 그러기 위해서는 t(3), t(7), t(11)의 값을 알아야 하는데 이것은 규모만 작아질 뿐 원 문제와 같은 형태가 된다. 즉 순환형 문제가 되는 것이다. 따라서 크기가 가장 작은 subtree—end vertex 만으로 이루어진 subtree—로부터 역으로 root까지 정보전달시간을 구해 나가면 된다.

하나의 end vertex로만 이루어진 subtree 안에서의 정보 전달 시간은 명백히 0이므로, 그림 1에서 t(1), t(4), t(5), t(9), t(10), t(13), t(14)의 값은 0이 된다. 또한 t(2)=4+2=6, t(8)=2+1=3, t(11)=5, t(12)=2 가 되는 것도 쉽게 알 수 있다. 그런데 t(3)를 구하기 위해서는 t(2), t(4) 중 큰 값을 갖는 subtree , 즉 vertex 2에 먼저 정보를 보내고 그 다음에 vertex 4에 보낸다. 그러면,

t(3)=max{1+t(2), 1+3+t(4)}=max{7, 4}=7

이 되며 마찬가지로

t(7)=max{2+t(8), 2+5+t(2)}=max{5, 9}=9

가 된다. 따라서 vertex 6에서는 vertex 7에 먼저 정보를 보내고 다음에 vertex 3, 그리고 vertex 11에 차례로 보내는 것이 최적이며 이 때 전체의 정보전달 시간은

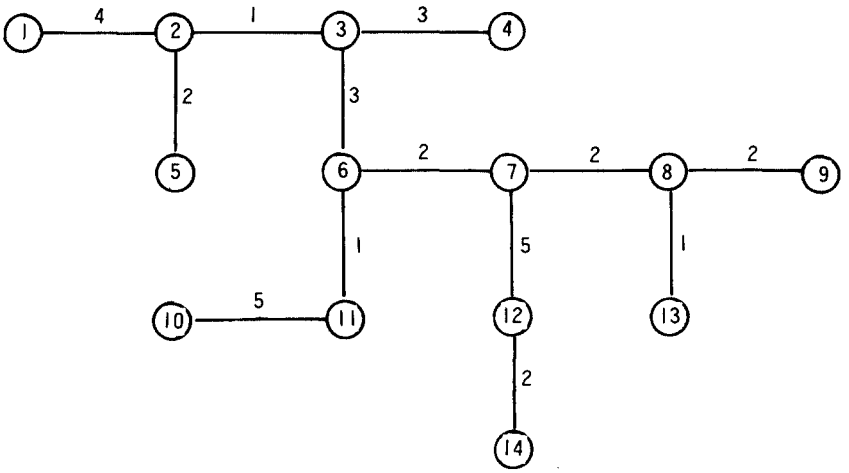


그림 1. tree-type network의 예
(각 edge옆에 있는 숫자는 그 edge에서의 정보 전달 시간)

$t(6)=\max\{2+t(7),\ 2+3+t(3),\ 2+3+1+t(11)\}$
$=\max\{11,\ 12,\ 11\}=12$

가 된다.

이상의 과정을 퍼스널 컴퓨터에서 실용화할 수 있도록 전산 패키지를 개발하였다. 패키지 개발은 다음과 같은 의미에서 필요하다 하겠다.

a. network connectivity 가 시간에 따라 고정되어 있지 않은 경우에 유효하게 쓰일 수 있다. 즉 vertex 간의 연결이 시간에 따라 또는 필요에 따라 바뀌는 경우에는 각 vertex에서는 빠른 시간 내에 정보전달순서를 찾아야 한다.

b. 통신네트워크가 나무형이 아닌 일반적 구조일 때, 그 네트워크 안에 하나의 나무형 구조(예를 들어 a minimum spanning tree)를 택하여 정보전달을 하는 경우가 있다. 이때는 택하는 나무 구조에 따라 이웃하는 vertex가 달라질 수 있으며 각각의 경우에 전달순서를 구하여 비교할 수 있다.

c. 일반적 구조의 네트워크에서의 정보전달순서를 결정하는데 보조적인 subroutine으로 활용될 수 있다.

이상의 의미에서와 같이 여기서 개발된 패키지는 하나의 subroutine 으로 다른 main routine의 한 부분으로서, 또는 자료처리 능력이 있는 vertex에서 필요할 때 처리과정의 일부분으로서 이용될 수 있다.

도구로서는 퍼스컴에서 자료 처리 능력이 우수하다고 알려진 d—BASE III[3]를 이용하였다. 구체적인 flow chart와 program list는 부록 1, 2와 같다.

III. 일반형 구조에서의 정보 전달

1. 개 요

통신네트워크의 구조는 local network를 제외하고는 나무형 구조가 아닌 경우가 대부분이다. 따라서 일반형 구조에 대한 연구가 필수적이라 하겠다. 그러나 network이론에서 일반형 구조에 관한 문제는 대부분이 NP-complete이며[4], 본 연구의 대상도 예외는 아니다. 즉 전체의 정보전달시간을 최소로 하는 전달순서를 결정하는 문제는 각 edge에서의 전달시간이 1인 경우라도 NP-complete임이 알려져 있다.[9]

따라서 이 경우에 대한 최적기법을 찾기가 어려우며 또 찾는다 하더라도 그 계산시간이 비현실적으로 매우 크리라고 짐작된다. 이때에는 차선책으로서 최

적기법은 아니지만 빠른 시간 내에 최적해에 가까운 해를 구할 수 있는 발견적 기법(heuristic technique)을 이용하는 것이 실용적이라 하겠다.

본 절에서는 일반적인 구조에서 각 edge에서의 정보전달시간이 1인 경우에 대해서 몇개의 발견적 해법을 제시하고, 이의 실용성을 검증하기 위해 전산 모의실험을 한다.

2. 발견적 해법

N=(V, E)를 vertex의 집합 V와 edge들의 집합 E로 이루어진 연결 네트워크(connected network)라고 하자. 각 edge에서의 전달시간은 1로 한다. V의 한 원소 v_0에 어떤 정보가 발생하여 이를 다른 모든 vertex에 가장 빠른 시간내에 전달하는 문제를 생각하자.

이 때 정보의 전달과정 중 중간단계를 살펴 보면 V를 크게 두가지 부류로 나눌 수 있다. 즉 이미 정보를 전달받은 vertex들(I)과 아직 전달받지 못한 vertex들(I')로 구분할 수 있다. 여기서 I에 있는 vertex들 중 I'의 vertex들과 직접 연결된(이웃하는) vertex들의 집합을 S, I'에 있는 vertex들 중 I의 vertex들과 직접 연결된 vertex들의 집합을 S'라 하자.(그림 2) 그러면 이 단계에서는 S에 있는 vertex에서 S'에 있는 vertex로 정보전달을 하게 된다. 이는 S와 S'를 연결하는 edge들을 통해 이루어진다.

여기서 S, S', 그리고 S와 S'를 엮결하는 edge들로 구성된 bipartite graph를 생각할 수 있으며, S와

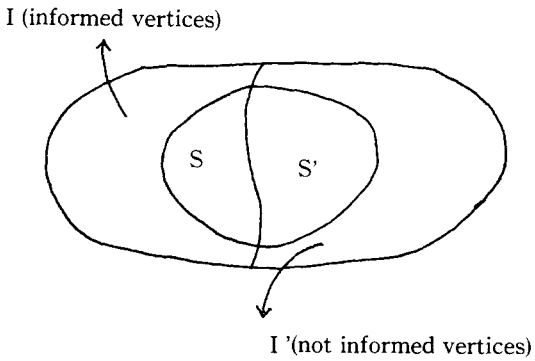


그림2. 정보전달의 중간 단계

S'의 vertex들을 어떻게 대응시킬 것인가 하는 문제가 발생한다. 왜냐하면 이 대응관계가 바로 정보전달의 송신자—수신자 관계가 되며, 이 대응관계에 따라서 전체의 전달시간이 달라지기 때문이다. 따라서 원 문제의 핵심은 bipartite cardinality matching problem(BCMP) — S의 vertex들과 S'의 vertex들 간에 edge를 통한 1 : 1 대응관계의 수를 최대로 하는 문제 — 로 귀결됨을 알 수 있다.

그런데 BCMP는 최적해가 여러 개인 경우가 대부분이다. 예를 들어 그림 3에서 최적해는 다음과 같이 여러개임을 쉽게 알 수 있다.

a. 1—4, 2—6, 3—7
b. 1—4, 2—6, 3—5
c. 1—5, 2—4, 3—6
d. 1—5, 2—4, 3—7
e. 1—5, 2—6, 3—7

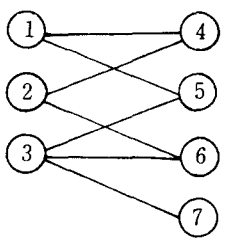


그림3. Bipartite Graph의 예

그러므로 단순히 maximum cardinality matching을 찾는 것 외에 어떤 vertex들을 먼저 대응시킬 것인가 — 어떤 vertex에 우선순위를 줄 것인가 — 하는 문제도 아울러 생각해야 한다. BCMP를 푸는 해법을 살펴 보면, 각 단계에서 하나의 vertex를 선택하여 상대방 vertex들 중의 하나와 대응을 시키는데, 어떤 vertex를 선택하더라도 일단 선택된 vertex는 최종단계의 optimal matching에 여전히 남게 된다. 이 maximum cardinality matching에 포함되지 않은 vertex들은 그 다음 iteration 이후에 대응시키게 되며 그러한 vertex들에 대한 정보전달은 그만큼 늦어지게 된다. 따라서 vertex들의 대응

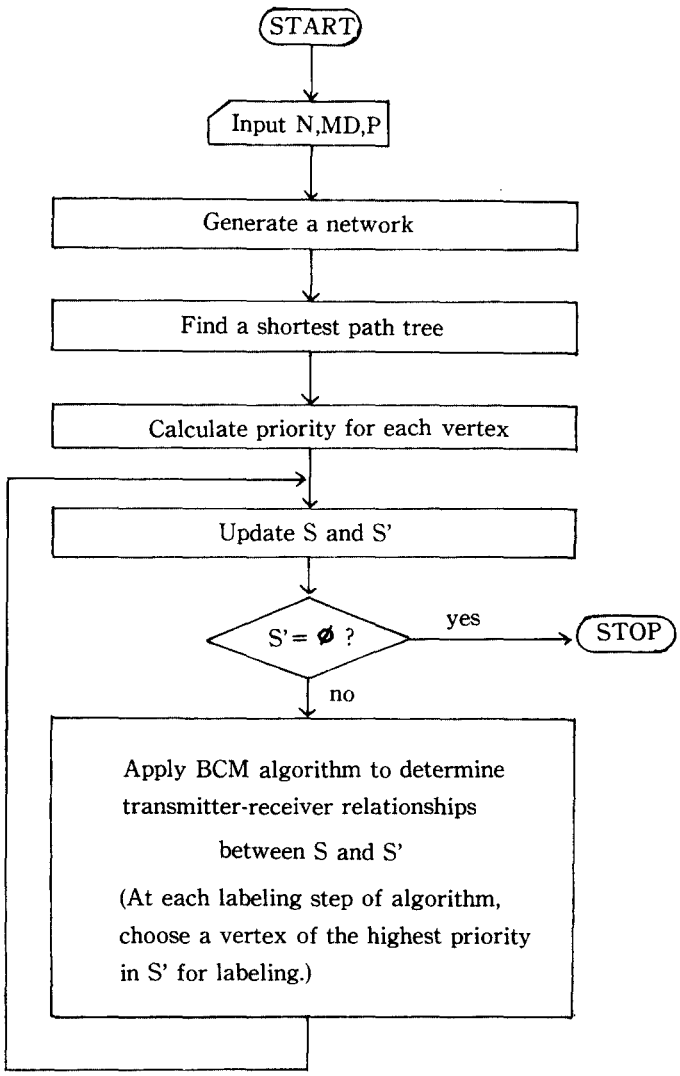


그림4. 일반형 구조에 대한 발견적 해법

순서가 중요한 점이 된다.

본 연구에서는 S'의 vertex들에 우선순위를 주는 방법을 다음과 같이 몇가지 제시한다. 여기서 제시된 우선순위를 참조하여 bipartite cardinality matching algorithm[7] 을 적용시켜도 역시 maximum cardinality matching을 구할 수 있다.

(1) 우선순위를 고려하지 않는 방법

단순히 BCM algorithm 을 적용하여 maximum cardinality matching 을 찾고 이에 따른 송신자—수신자 관계로부터 정보를 전달하는 방법이다. 이는 가장 단순한 방법으로 전체 전달시간의 최적해와 다소 거리가 생길 수 있다. 그럼에도 불구하고 이 방법은 다른 방법을 찾기 위한 시작점이라고 볼 수 있으며, 여기서 구해진 해는 다른 방법의 평가기준이 될 수 있다.

(2) vertex의 degree에 따라 우선순위를 주는 방법

여기서 어떤 vertex의 degree라 함은 그 vertex에 연결된 edge들의 수를 말한다. 이 방법은 이웃하는 vertex가 많은 vertex에 먼저 정보전달을 함으로써 전체 정보전달시간을 줄일 수 있다는 기대감에서 직관적으로 생각해 볼 수 있다.

(3) vertex의 outward degree에 따라 우선순위를 주는 방법

여기서 어떤 vertex의 outward degree라 함은 그 vertex에 이웃하는 vertex들 중에서 아직 정보를 받지 않은 vertex들의 수를 말한다. 이 방법은 앞의 (2)의 방법을 개선시킨 것이라 할 수 있다. 이미 정보를 전달받은 vertex와 연결되는 edge는 더 이상 고려할 필요가 없기 때문이다.

(4) 가장 깊은 vertex로 가는 길목에 있는 vertex에 최우선순위를 주는 방법

여기서 어떤 vertex의 깊이라 함은 정보의 발생지 v_0로부터 그 vertex까지의 최단경로(shortest path)의 길이를 말한다. 단, 각 edge의 길이는 정보전달시간인 1로 한다. 이 방법은 v_0로부터 멀리 떨어져 있는 vertex가 정보를 받는 시간이 전체의 정보전달시간에 직접적으로 영향을 준다는 데 근거를 둔 것이다. 이 방법을 적용키 위해서는 먼저 v_0로부터 다른 모든 vertex까지의 최단 경로들을 구해야 한다. 이 때 이 최단 경로들은 하나의 tree—shortest path tree— 를 형성한다.[7]

(5) shortest path tree에서의 최적 전달순서에 따라 우선순위를 주는 방법

이 방법은 (4)의 방법과 연관하여 나무형 구조에서의 연구결과를 직접 이용한 것이다. 주어진 네트워크가 나무형이면 이 방법 자체가 최적기법이 되고 나무형이 아닐 때는 각 vertex까지의 최단경로를 따라 정보전달을 하는 것이 좋다고 보는 것이다. 이 때 shortest path tree 에 포함되지 않은 edge도 필요할 때 보조적으로 이용할 수 있으므로 전체 전달시간은 shortest path tree에서보다 더 단축시킬 수 있다. 따라서 이 방법과 (4)의 방법이 최적해에 가장 가까운 해를 찾을 수 있는 것이라고 기대된다.

이상의 방법에 따라 우선순위를 정하여 정보전달을 하는 과정에 대한 전체적인 윤곽을 정리하면 그림 4와 같다. 단, (3)의 방법에서는 outward degree를 구하는 subroutine이, (4)와 (5)의 방법에서는 초기에 shortest path tree를 구하고 각 vertex의 우선순위를 결정하는 routine이 따로 필요하다. 전체 과정을 세부적으로 살펴보면 다음과 같이 크게 4부분으로 나눌 수 있다.

가.네트워크를 발생시키는 과정

네트워크를 확률적으로 만들어 내는 과정이다. 이는 vertex의 총 수 N을 입력으로 하여 일반형 연결 네트워크(general-type connected network)를 발생시키는데, 각 vertex의 degree는 시행회수가 MD (maximum degree)이고 1회 시행시 성공확률이 P인 이항분포, b(MD, P), 를 한다고 가정한다. 단, degree가 0인 경우는 없으므로 이 분포에서 0의 값을 배제한 절단형분포를 사용한다. 그리고 MD와 P의 값에 따라서 연결되지 않은 네트워크가 생성될 수 있는데, 이 경우에는 edge를 추가하여 전체네트워크가 연결되도록 한다. 따라서 각 vertex의 degree의 확률분포는 이항분포에서 약간 오른쪽으로 치우친 형태가 될 수 있다.

나.shortest path tree 를 구하는 과정

앞에서 발생시킨 네크워크에서 임의로 정보의 출발점을 선택하여, 이로부터 다른 모든 vertex까지의 최단 경로와 이런 경로들로 구성된 shortest path tree를 구하는 과정이다. 여기서는 Dijkstra의 방법을 이용한다.[7]

다.각 vertex의 가중치를 구하는 과정

shortest path tree 를 토대로 각 vertex의 우선순위 결정에 이용될 가중치를 구하는 과정이다. 방법

(4)에서는 각 vertex의 깊이에 근거한 가중치를, 방법 (5)에서는 각 subtree 에서의 최소 정보전달시간을 구한다. 여기서 구한 가중치에 따라 큰 순서대로 각 vertex에 우선순위를 부여한다.

라. 정보전달순서를 구하는 과정

정보의 출발점으로부터 전달순서를 차례로 구해 나가는 과정이다. 이 과정은 다시 두 부분으로 나눌 수 있다.

먼저 S와 S'를 결정하는 부분이다. 이미 정보를 전달받은 vertex들에 이웃하는 vertex들 중 아직 정보를 전달받지 못한 것들은 S'라 두고, S'에 이웃하는 vertex들 중 이미 정보를 전달받은 것들을 S라 두면 된다.

다음으로 S와 S'에 대하여 BCM algorithm을 적용하여 송신자-수신자 관계의 수를 최대로 하는 부분이다. 이 때 S'의 vertex들을 앞에서 구한 우선순위에 따라 차례로 대응 시켜나간다. 즉 BCM algorithm에서는 각 단계에서 하나의 vertex를 골라 labeling을 한 다음 그 vertex의 대응관계를 결정하는데, 이 때 앞에서 구한 우선순위에 따라 차례로 vertex를 선택하여 대응관계를 정한다. 이 대응관계에 따라 정보전달을 한다고 생각한다. 따라서 우선순위가 낮은 vertex들은 이 단계에서 정보를 받지 못하고 다음 단계 이후에 받게 된다. 이 단계가 끝나면 다시 S와 S'를 결정하는 부분으로 간다.

3. 결과 분석

앞에서 제시한 5가지의 우선순위 결정방법의 성능을 비교하기 위해 N, MD, P의 여러 값에 대하여 전산모의실험을 하였다. 전체 전달시간(정보의 출발점으로부터 다른 모든 vertex까지 정보를 전달하는 시간)을 평가기준으로 한 결과를 정리하면 표 1로부터

표 1. N=20인 경우

MD	P	Dens	H=1	H=2	H=3	H=4	H=5	LB
3	0.3	0.100	9.2	8.4	8.4	8.4	8.2	7.8
3	0.4	0.100	9.8	8.2	8.2	7.4	7.4	6.4
3	0.5	0.103	9.2	8.4	8.2	7.6	7.8	7.4
3	0.6	0.111	0.2	9.8	9.6	9.0	8.8	8.0
3	0.7	0.122	8.4	7.8	7.6	7.0	7.0	6.8
5	0.3	0.111	8.8	7.4	7.4	7.2	7.0	6.2
5	0.4	0.118	9.0	7.4	7.2	7.2	6.6	5.6
5	0.5	0.147	8.0	7.0	7.0	6.8	6.8	6.0
5	0.6	0.166	6.6	6.2	5.8	5.8	5.6	5.0
5	0.7	0.182	6.4	6.2	5.6	5.8	5.6	5.0
7	0.3	0.132	7.0	7.0	7.0	6.6	6.6	5.6
7	0.4	0.172	6.8	6.6	6.0	6.4	6.4	5.0
7	0.5	0.179	6.6	5.8	5.6	6.0	5.6	5.0
7	0.6	0.241	5.8	5.2	5.0	5.2	5.0	5.0
7	0.7	0.272	5.0	5.0	5.0	5.0	5.0	5.0
9	0.3	0.154	7.6	7.4	7.0	6.8	6.8	6.0
9	0.4	0.195	6.2	5.6	5.2	5.6	5.4	5.0
9	0.5	0.258	5.2	5.2	5.2	5.2	5.2	5.0
9	0.6	0.302	5.0	5.0	5.0	5.0	5.0	5.0
9	0.7	0.355	5.0	5.0	5.0	5.0	5.0	5.0

표 2. N=50인 경우

MD	P	Dens	H=1	H=2	H=3	H=4	H=5	LB
3	0.3	0.040	13.6	13.2	13.2	12.6	12.4	11.4
3	0.4	0.040	13.2	13.4	13.4	12.0	11.8	11.0
3	0.5	0.040	16.6	14.4	14.4	13.2	13.2	12.6
3	0.6	0.042	18.6	17.6	17.4	16.0	16.2	15.8
3	0.7	0.045	16.8	16.2	16.2	14.6	14.6	13.8
5	0.3	0.042	18.6	16.2	16.0	15.2	15.0	14.4
5	0.4	0.045	16.4	15.6	15.2	13.4	13.4	13.2
5	0.5	0.053	13.6	12.6	12.0	11.2	11.0	9.4
5	0.6	0.064	9.4	9.6	9.2	8.6	8.2	7.0
5	0.7	0.073	8.8	7.2	7.8	8.0	7.8	6.6
7	0.3	0.045	18.8	16.6	16.4	15.2	15.2	14.8
7	0.4	0.059	10.8	9.8	9.6	8.8	8.8	7.4
7	0.5	0.072	9.2	8.6	8.0	8.0	7.8	6.4
7	0.6	0.088	7.6	6.0	6.6	7.0	7.0	6.0
7	0.7	0.103	7.4	6.2	6.4	6.8	6.4	6.0
9	0.3	0.057	11.2	10.4	10.2	9.6	9.6	8.0
9	0.4	0.073	8.4	7.0	7.8	7.8	7.2	6.0
9	0.5	0.094	7.8	7.2	7.0	7.0	7.0	6.0
9	0.6	0.113	6.8	6.8	6.2	6.2	6.6	6.0
9	0.7	0.129	6.6	6.2	6.0	6.0	6.0	6.0

표 3. N=80인 경우

MD	P	Dens	H=1	H=2	H=3	H=4	H=5	LB
3	0.3	0.025	17.0	16.2	16.2	15.2	15.0	14.0
3	0.4	0.025	18.2	18.4	18.4	16.0	16.2	15.6
3	0.5	0.025	21.2	20.0	20.0	17.6	17.6	17.0
3	0.6	0.026	27.0	25.4	25.2	23.2	23.0	22.4
3	0.7	0.027	21.8	20.8	20.4	18.0	18.2	17.6
5	0.3	0.025	21.6	20.2	20.2	17.8	17.4	16.8
5	0.4	0.028	19.8	18.6	17.8	15.8	16.0	15.0
5	0.5	0.035	13.0	12.8	12.4	11.4	11.0	9.6
5	0.6	0.039	12.0	11.4	10.6	9.8	9.4	8.0
5	0.7	0.044	9.2	8.8	8.6	8.6	8.4	7.0
7	0.3	0.029	17.6	16.8	16.0	14.4	14.4	12.8
7	0.4	0.037	15.8	13.8	13.2	11.6	11.2	10.2
7	0.5	0.047	9.6	8.8	8.2	8.4	8.0	7.0
7	0.6	0.054	9.0	8.4	8.2	8.4	7.8	7.0
7	0.7	0.063	8.0	7.6	7.2	7.2	7.8	7.0
9	0.3	0.036	13.6	12.8	12.2	11.0	10.8	9.6
9	0.4	0.047	9.8	9.0	8.8	8.4	8.6	7.0
9	0.5	0.056	8.6	8.2	7.8	7.8	8.0	7.0
9	0.6	0.069	7.8	7.2	7.0	7.2	7.0	7.0
9	0.7	0.083	7.4	7.0	7.0	7.0	7.0	7.0

표 4. N=150인 경우

MD	P	Dens	H=1	H=2	H=3	H=4	H=5	LB
3	0.3	0.013	21.0	20.2	20.2	17.4	17.2	16.4
3	0.4	0.013	21.8	20.0	20.0	17.8	17.6	16.2
3	0.5	0.013	27.8	25.6	25.6	22.0	22.2	21.4
3	0.6	0.014	28.4	26.0	26.2	23.4	23.6	22.6
3	0.7	0.015	29.2	28.2	28.4	24.4	24.4	24.2
5	0.3	0.014	26.6	25.8	25.6	22.2	22.8	21.2
5	0.4	0.015	25.4	23.8	23.4	20.4	21.0	19.6
5	0.5	0.018	17.4	17.0	16.6	14.6	14.0	13.0
5	0.6	0.020	13.0	12.2	12.0	11.0	10.8	9.2
5	0.7	0.024	12.0	11.6	11.4	9.8	10.2	8.4
7	0.3	0.016	27.0	25.8	25.4	22.2	22.4	21.0
7	0.4	0.019	17.4	16.4	15.6	14.0	14.2	12.6
7	0.5	0.023	12.2	12.4	11.6	10.8	10.6	8.4
7	0.6	0.029	9.8	9.8	9.2	9.4	9.0	8.0
7	0.7	0.033	9.0	9.0	8.2	8.8	8.8	8.0
9	0.3	0.019	14.8	13.4	12.8	11.6	11.4	9.4
9	0.4	0.024	12.4	11.6	11.2	10.0	9.8	8.4
9	0.5	0.031	9.2	9.0	8.6	8.8	8.8	8.0
9	0.6	0.032	9.0	8.6	8.0	8.4	8.2	8.0
9	0.7	0.046	8.4	8.0	8.0	8.0	8.0	8.0

표 5. N=180인 경우

MD	P	Dens	H=1	H=2	H=3	H=4	H=5	LB
3	0.3	0.011	18.6	19.0	19.0	16.6	16.6	14.8
3	0.4	0.011	22.8	23.4	23.4	20.0	19.8	18.4
3	0.5	0.011	31.6	30.4	30.2	27.4	27.4	26.6
3	0.6	0.011	38.4	32.6	36.2	32.6	32.8	32.2
3	0.7	0.012	33.8	36.2	32.0	28.8	28.8	28.4
5	0.3	0.011	30.0	26.8	26.6	23.4	24.0	23.0
5	0.4	0.012	31.4	29.4	29.4	24.2	24.2	23.4
5	0.5	0.015	19.0	18.8	18.0	15.8	16.0	14.4
5	0.6	0.017	15.0	13.2	13.0	11.8	11.4	9.6
5	0.7	0.020	11.6	10.8	10.8	10.4	10.2	8.0
7	0.3	0.013	27.8	25.4	24.4	21.2	21.2	20.0
7	0.4	0.016	15.4	14.6	14.0	13.0	13.4	10.4
7	0.5	0.019	12.2	11.2	11.0	10.2	10.2	8.2
7	0.6	0.024	10.2	9.2	9.4	9.8	9.6	8.0
7	0.7	0.028	9.0	9.0	8.4	9.0	9.0	8.0
9	0.3	0.016	15.8	15.2	14.8	13.0	13.6	11.2
9	0.4	0.020	11.6	10.8	10.6	10.4	10.6	8.0
9	0.5	0.025	10.8	9.8	9.4	9.2	9.0	8.0
9	0.6	0.030	9.6	9.2	8.4	9.0	8.6	8.0
9	0.7	0.035	8.8	8.8	8.0	8.2	8.4	8.0

표 5까지와 같다. 여기서 Dens는 생성된 네트워크의 edge density(생성된 네트워크의 edge의 총수 ÷ 완전연결 네트워크의 edge의 총수)이고, H는 우선순위를 결정하는 방법을 나타낸다. 하나의 생성된 네트워크에 대해서 전체 전달시간의 하한값도 같이 구하였다. 이것은 정보의 출발점으로부터 각 vertex까지의 최단경로의 길이 중 가장 큰 값을 취한 것인데, 이 값이 $\lceil \log_2 N \rceil$보다 작으면 전체전달시간의 하한은 $\lceil \log_2 N \rceil$으로 된다. 왜냐하면 vertex가 N개 있을 때 이것들이 완전연결 네트워크(full connectivity network)를 형성하더라도 전체 전달시간은 $\lceil \log_2 N \rceil$보다 작아질 수 없기 때문이다.[2)]

한 조의 (N, MD, P) 값에 대하여 각각 5개의 네트워크를 생성시키고, 각 방법의 평균 전달시간을 표에 나타냈다. 그리고 각 네트워크에 대하여 전체 전달시간의 하한을 구하고 그 평균값을 LB(lower bound)로 나타내었다.

결과를 살펴보면 P가 작은 값일 때는 전반적으로 방법 (4), (5)가 양호한 것으로 나타났는데, N이 작을 때는 각 방법 간에 차이가 크게 없으나 N이 큰 값일수록 그 차이는 점점 벌어진다. 예를 들어 MD=5, P=0.4인 경우 N=20일 때 방법 (2)와 방법 (4)의 차이는 0.2였으나, N=50일 때는 2.2, N=150일 때는 3.4, N=180일 때는 5.2가 된다.

그러나 MD와 P의 값이 클 때는 각 방법 간에 차이가 거의 없음을 보여준다. 오히려 (N, MD, P)의 값이 (50, 7, 0.6), (150, 9, 0.6), (180, 9, 0.7)일 때와 같이 방법 (3)이 더 좋은 경우도 있다. 하지만 그 차이는 미미한 정도이다.

(N, MD, P)의 값이 (20, 7, 0.7), (20, 9, 0.7), (50, 9, 0.7), (80, 9, 0.7), (150, 9, 0.7)일 때 방법 (3), (4), (5)에 의해 구한 전달시간이 하한값과 일치하고 있음을 볼 수 있다. 이것은 vertex 간에 degree의 차이가 별로 없고(P=0.7) 그 평균이 약 5이상이면, 본 연구에서 제시된 해법으로써 거의 최적해를 구할 수 있음을 의미한다. 또한 N이 커지더라도 최적에 가까운 해를 얻을 수 있음을 짐작할 수 있다.

이상의 결과를 정리하면 다음과 같다.

첫째, P가 작을 때 방법 (4)와 (5)에 의해 최적에 가까운 해를 얻을 수 있다. 즉 네트워크의 형태가 나무형에 가까울 때는 정보전달순서가 전체 전달시간에 민감한 영향을 주며, shortest path tree와 연관하여 그 순서를 결정하는 것이 효율적이라 할 수 있다.

둘째, P가 클 때 즉 각 vertex의 degree가 별 차이가 없을 때 정보전달순서는 전체의 전달시간에 큰 영향을 주지 않는다.

세째, 방법 (4)와 (5)사이엔 차이가 별로 없으며, 거의 모든 경우에 이 두 방법이 다른 방법보다 나쁘지 않다. 특히 P가 작은 경우 N이 커짐에 따라 이 두 방법의 우수성은 점점 크게 나타난다. 따라서 실제 문제에 적용할 때는 이 두 방법 중에서 선택하는 것이 좋다.

네째, 각 vertex 간에 degree의 차이가 별로 없고 그 값이 약 7이상이면 거의 최적해 또는 최적에 근접한 해를 구할 수 있다. 이는 네트워크 설계시 한 vertex에 연결하는 edge의 수의 적정선을 정하는데 중요한 참고로 활용될 수 있다.

IV. 결 론

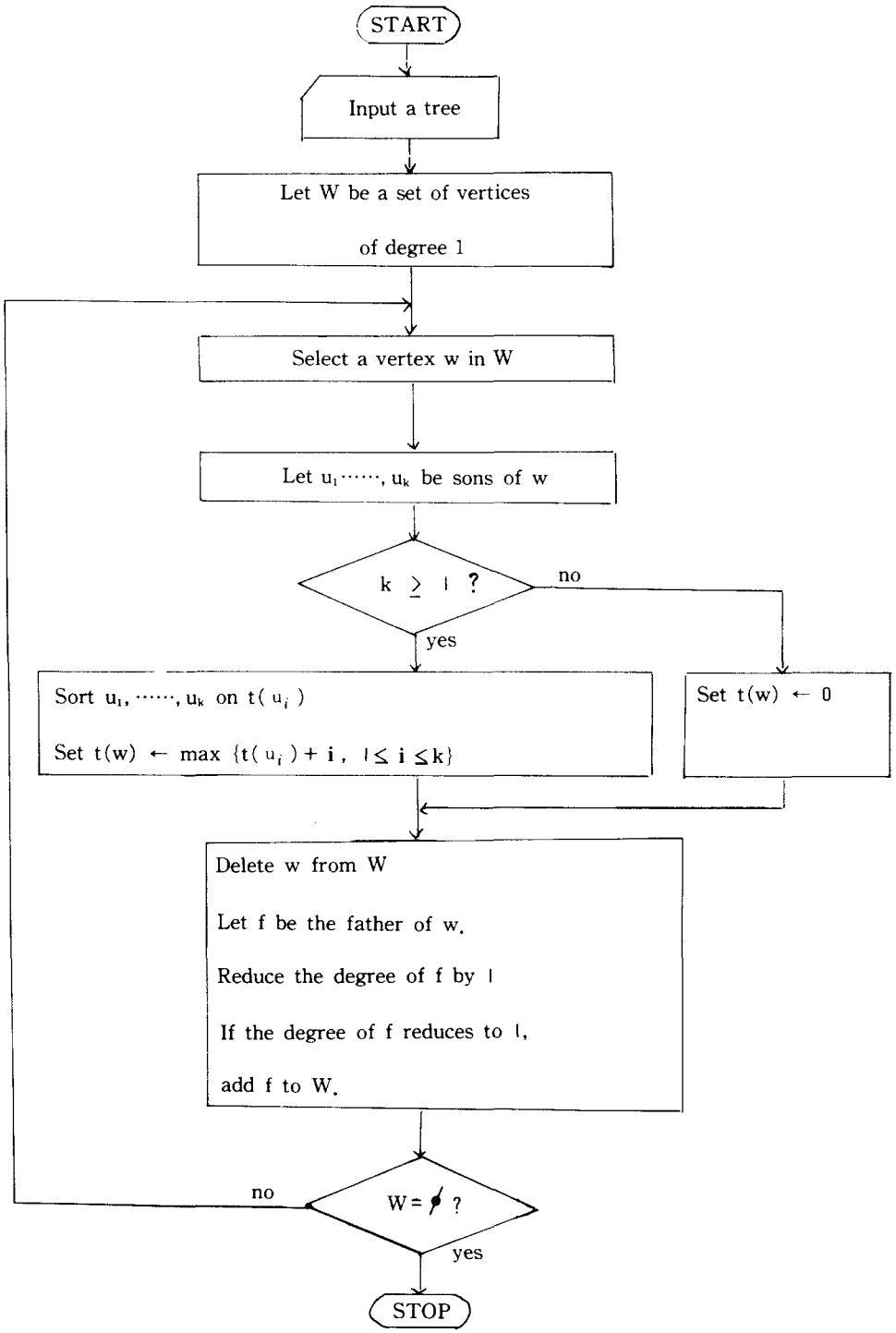
본 연구에서는 통신네트워크에서 local broadcasting 방식에 따라 정보전달을 하는 문제를 다루었다. 먼저 나무형 구조의 경우 최적 기법에 대해 고찰하고 이의 퍼스널 컴퓨터용 전산패키지를 개발하였다. 이것은 일반형 구조의 경우에서 하나의 subroutine으로 이용될 수 있다. 다음으로 일반형 구조의 경우 발견적 해법을 제시하고 이에 대한 전산 모의실험을 수행하였다 그 결과 일반형 구조의 네트워크에서 shortest path tree 를 구한 다음, 이에 근거하여 정보전달순서를 결정하는 것이 효과적임을 알 수 있었다.

본 연구에 대한 확장으로서 각 edge에서의 전달시간이 단위시간이 아닌 좀더 일반적인 경우에 대해서 연구할 필요가 있다 또 전산 모의실험을 할 때 좀더 다양한 확률분포에 대해 검증할 필요가 있겠다. 이를 위해서는 현존하는 통신네트워크에 대한 통계적 연구가 선행되어야 할 것이다.

references

1. Ashton-Tate, *Learning and Using dBASE* III *Plus*, 1985.
2. A. M. Farley, "*Minimal Broadcast Networks*,"

Networks, Vol. 9, pp. 313—332, 1976.
3. A. M. Farley, *Minimum—Time, Line Broadcast Networks* " Networks, Vol. 10, pp. 59—70, 1980.
4. M. R. Garey & D. S. Johnson, *Computers and Intractability*, W. H. Freeman and Company, 1979.
5. J. M. Koh & D. W. Tcha, "*Minimum Broadcasting* Time *in a Tree-type Network*," J. of KISS, Vol. 11, No. 4, pp. 296—301, 1984.
6. J. M. Koh, "*A Broadcasting Center in a Tree-type Network*," UIT Report, Vol. 16, No. 2, pp. 191—198, 1985.
7. E. L. Lawler, *Combinatorial Optimization : Networks and Matroids*, Holt, Rinehart and Winston, 1976.
8. A. Proskurowski, "*Minimum Broadcast Trees*," IEEE Trans. Compu., Vol. C—30, No. 5, pp. 363—366, 1981.
9. P. J. Slater, E. J. Cockayne, & S. T. Hedetniemi, "*Information Dissemination in Trees*," SIAM J. COMPUT, Vol. 10, No. 4, pp. 692—701, 1981.



부록1. 나무형 구조에서의 정보 전달 순서 결정 과정

부록 2 - 1 나무형 네트워크의 입력

```
* Input a tree  (filename = treei)
set talk off
set safety off
select 1
use tree
delete all
pack
clear
@1,3 say "Input your tree with edge transmission times."
append blank
@5,5 say "Root vertex ID : " get id picture "xxxxx"
read
cfrn=1
nn=1
do while .t.
r=id
k=1
do while .t.
clear
@10,5 say "Current father is "
@10,25 say r
@15,5 say "Input son vertex."
@17,10 say "(If no more son, then press RETURN.)"
s="     "
@20,5 say "ID : " get s picture "xxxxx"
read
  if s="     "
        exit
  endif
t=0
@22,5 say "Transmission time to the father : " get t picture "99999"
read
append blank
nn=nn+1
k=k+1
replace id with s
replace fath with cfrn
replace ett with t
csrn=recno()
 if k=2
         go cfrn
         replace son with csrn
         go csrn
    else
         skip -1
         replace broth with csrn
         skip
 endif
enddo
go cfrn
replace degr with k
cfrn=cfrn+1
if cfrn > nn
          exit
endif
go cfrn
enddo
go top
replace degr with 990
clear
@3,5 say "If you had mistakes, try again"
@5,5 say "by typing DO TREEI."
@7,5 say "Otherwise, type DO TREEC."
clear all
```

부록 2 - 2 나무형 네트워크에서의 전달순서 결정

```
* Calculation of broadcasting time, t(root)
*    (filename = treec)
set talk off
set safety off
select 1
use tree
go bottom
nn=recno()
select 2
use w
delete all
pack
append blank
nrw 1
select 3
use u
delete all
pack
append blank
nru=1
select 1
clear

go top
do while .not. eof()
replace deg with degr
if son=0
  ? "*", id
  frn fath
  crn=recno()
  go frn
  replace deg with deg-1
  go crn
  endif
skip
enddo

go top
do while .not. eof()
 if  deg=1 .and. son<>0
    crn=recno()
   select 2
    replace wrn with crn
    skip
     if eof()
            append blank
            replace wrn with 1
     endif
   select 1
 endif
    skip
enddo
```

```
select 2
go top
crn=wrn

do while .t.
  select 3
  use u
  select 1
  go crn
  srn=son
  nul=0
   do while srn<>0
     nul=nul+1
     go srn
     tu=bt
     select 3
        if nul>nru
            append blank
            nru=nru+1
        endif
        go nul
        replace tul with -tu
        replace ulrn with srn
     select 1
     srn=broth
   enddo

   select 3
        skip
        do while .not. eof()
        replace tul with 1
        skip
        enddo

      go top
      index on tul to undxl
      use u index undxl

      go top
      rn=ulrn
    select 1
        go crn
        replace fcv with rn
        go rn
        et=ett
        tv=bt+et
        i=1
    do while i<nul
        i=i+1
     select 3
        skip
        trn=ulrn
     select 1
        replace cseq with trn
        rn=trn
        go rn
        et=et+ett
        btt=bt+et
        if tv<btt
             tv=btt
        endif
    enddo
```

```
  go crn
  replace bt with tv
    go fcv
    frcv=id
    go crn
? "  *", id, bt, frcv
   if crn=1
       exit
    endif
  frn=fath
  go frn
  if deg=2
      crn=frn
  else
      replace deg with deg-1
      select 2
      skip
       if eof()
            exit
       endif
      crn=wrn
  endif
enddo
?
?
?
wait
clear
@3,5 say "Now all calculations are completed
@6,5 say "If you want to see the result,"
@8,5 say "Type DO TREER."
clear all
```

*부록 2-3. 나무형 네트워크에서의 결과 출력

```
* Result display
*   (filename = treer)
set talk off
set safety off
use tree
clear
@5,5 say "Result display."
@8,6 say "1. Monitor"
@10,6 say "2. Printer"
a=0
@12,5 say "Select the No. : " get a picture "9"
read
do case
    case a=0
        clear all
        cancel
    case a=2
        set print on
endcase

clear
? "Vertex       BT(v)       Call sequence"
?
do while .not. eof()
crn=recno()
pl=id+str(bt)+"          "
rn=fcv
if rn<>0
   go rn
   pl=pl+id
   nrn=cseq
    do while nrn<>0
      go nrn
      pl=pl+"  "+id
      nrn=cseq
    enddo
endif
? pl
go crn
skip
enddo
clear all
```