2001 년 한국정보처리학회 추계 학술발표논문집 제 8 권 제 2 호

네트워크 토폴로지 기반의 동적 폴링 기법에 관한 연구

안병오*, 조강홍*, 강현중**, 안성진***, 정진욱*
*성균관대학교 전기전자 및 컴퓨터공학부
**서일대학 전자계산과
***성균관대학교 컴퓨터교육과
e-mail : boahn@songgang.skku.ac.kr

A Study on the Scheme for Dynamic Polling Based on Network Topology

Byung Oh Ahn*, Kang Hong Cho*, Hyun Joong Kang**, Seong jin Ahn ***, Jin Wook Chung*
*Department of Electrical and Computer Engineering
Sungkyunkwan University
**Department of Computer Science
Seoil College
***Department of Computer Education
Sungkyunkwan University

요 약

이 논문에서는 네트워크 모니터링을 위해 기존의 기법들과는 달리 네트워크 토폴로지를 기반으로 한 접근 방법을 통해 보다 효율적으로 폴링 트래픽에 대한 오버헤드를 제어하는 동적 폴링 기법을 제시하였다. 제안된 기법에서는 폴링 트래픽에 대한 오버헤드를 제어하기 위해 관리 도메인의 네트워크 토폴로지 정보를 이용한다. 네트워크 토폴로지 정보를 이용하여 폴링을 위한 노드 리스트에서 각 노드들 간의 연결 관계를 표현한다면 토폴로지 순서에 의해 빠르고 정확하게 실시간 네트워크 정보를 파악할 수 있다. 따라서 이러한 방법을 통해 기존의 제안된 기법들 보다 폴링 처리율을 높임으로써 보다 효율적인 동적 폴링 기법을 제안하였으며, 다른 기법들과의 비교를 통해 제안된 기법의 효율성을 증명하였다.

1. 서론

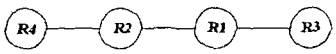
네트워크 관리 도구에서 네트워크 모니터링은 실시간의 네트워크 상태를 파악하기 위한 가장 중요한 기능들 중 하나이다. SNMP(Simple Network Management Protocol)[1]와 같은 네트워크 관리 프로토콜에서 폴링은 이러한 목적에 사용된다. 폴링은 보통 관리 도메인 내의 노드의 UP/DOWN 정보를 수집한다. 이 경우에는 ping 프로그램만으로 충분하나, 네트워크 규모가 커지고 관리 도구가 진화함에 따라 네트워크의 상태를 파악하기 위한 폴링 기법이 필요하게 되었다. 연속적인 두 개의 폴링 요청에 대해 폴링 간격이 너무 짧을 경우 폴링 메시지 트래픽이 증가하여 네트워크에 과중한 부하를 줄 것이다. 이와는 반대로 폴링 간격이 너무 길 경우에는 네트워크의 상태를 실시간으로 파악할 수 없으며, 폴링 처리율(throughput)도 또한 매우 낮아질 것이다.

이 논문에서는 기존의 기법들과는 다른 새로운 접근 방법을 통해 보다 효율적으로 폴링 트래픽에 대한 오버헤드를 제어하는 동적 폴링 기법을 제안하고자 한다. 다음 장에서, 우리는 기존의 폴링 기법에 대해 간단히 살펴볼 것이며, 그 기법들의 단점을 지적할 것이다. 그리고 3 장에서는 제안된 기법을 소개할 것이며 나머지 장에서는 제안된 기법과 관련

된 세부 사항에 대해 기술할 것이다.

2. 기존의 폴링 기법들과 그들의 단점

LAN 과 같은 네트워크에서 대역폭은 데이터 트래픽과 네트워크 관리를 위한 폴링 메시지 트래픽이 같이 공유된다. 일반적으로 폴링 메시지 트래픽은 네트워크의 최소 대역폭의 5%로 제한되는 것이 바람직하다[1]. 폴링 메시지 트래픽을 제어하기 위해 몇 가지 동적 폴링 기법들이 제시되었다. 이러한 기법들은 네트워크 혼잡과 관련있는 RTT 를 기반으로 다음 번의 폴링 간격을 결정하는 방법[2,3,4]과 CPU 나 메모리 이용률과 같은 에이전트의 프로세스 부하를 나타내는 관리 정보를 이용하여 에이전트의 상태를 몇 가지 레벨로 분류하여 레벨에 따라 폴링 간격을 달리 적용하는 방법[5,6] 등으로 구분할 수 있다. 네트워크 관리 도구의 관점으로 볼 때 폴링을 하기위한 노드들은 단순한 하나의 리스트로 유지된다. 이러한 폴링 리스트는 네트워크 토폴로지와는 관계가 없기 때문에 폴링 절차는 단순히 리스트의 앞에서부터 순차적으로 일어날 것이다. 따라서 앞에서 기술한 기법들을 여기서부터는 임의 순서 폴링이라고 하겠다. 그리고 임의 순서 폴링에서는 폴링 간격이 관리 도구에서 명시적으로 지정된다고 가정하자. R1, R2, R3, R4 의 4 개의 노드에 대해 폴링을 수행한다고 하자. 그리고 각 노드에 대해 MIB(Management Information Base) 변수의 값을 읽어오는데 걸리는 시간이 3, 2, 4, 1 초(sec)라고 하자. 그러면 한 라운드($Round_j$)에는 10 초가 걸린다. 만약 폴링 리스트에서 각 노드간 연결 정보를 알고 있다고 한다면, [그림 1]에서 올바른 폴링 순서는 R4, R2, R1, R3 이 되어야 할 것이다.



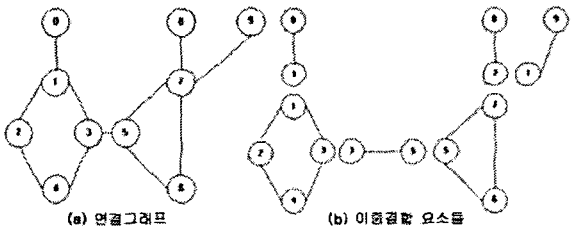
[그림 1] 폴링 리스트에서의 각 노드간 연결 정보

여기서 R2 가 DOWN 되거나 혼잡 상태일 경우에는 폴링 요청에 대한 Timeout 으로 인한 재요청이 발생한다. 이때 걸리는 시간이 back-off 알고리즘에 의해 2^k(k : 재전송 횟수)이므로 R2 에 대한 폴링 응답 시간은 7 초가 더해진 9 초가 되며 이는 R2 에 연결된 R1 은 10 초, R3 는 11 초가 된다. 이 경우 임의 순서 폴링에서는 한 라운드 당 31 초가 소요된다. 그러나 만약 각 노드간 연결 정보, 즉 네트워크 토폴로지를 이용한다면 실질적으로는 R4, R2 의 두 노드만 폴링하면 되므로 한 라운드 당 10 초가 걸릴 것이다.

3. 새로운 동적 폴링 기법 제안

앞의 내용에서 볼 수 있듯이 폴링 리스트에서 각 노드간 연결 정보, 즉 네트워크 토폴로지 정보를 통해 각 라운드당 폴링에 소요되는 시간을 줄일 수 있었다. 또한, 이렇게 각 라운드 당 소요 시간을 줄임으로써 폴링 처리율을 높일 수 있었다. 여기서 제안하려는 동적 폴링 기법은 바로 이 점을 이용한 것이다. 네트워크 토폴로지를 이론적으로 표현하는 데는 그래프를 이용할 수 있다. 그래프는 정점과 간선들의 집합으로, **그래프 G** 는 공집합이 아닌 정점들의 유한집합과 공집합도 허용하는 간선들의 유한집합으로 정의할 수 있다. V(G)와 E(G)는 그래프 G 의 정점들의 집합과 간선들의 집합을 나타내며, 임의의 그래프를 G = (V, E)로 표기할 수도 있다. 따라서 노드는 정점으로 두 노드를 연결하는 링크는 간선으로 볼 수 있다. 이 논문에서는 간선을 나타내는 정점의 쌍에 순서가 없는 무방향 그래프(undirected graph)를 이용하여 네트워크 토폴로지를 표현하며, 각 노드에 대한 연산을 위해 깊이 우선 탐색(Depth First Search : DFS)과 너비 우선 탐색(Breadth First Search : BFS) 연산을 이용한다. 또 다른 중요한 개념으로 신장트리와 이중결합 요소, 그리고 단절점이 있다. **신장트리(spanning tree)**는 G 의 간선들로만 구성되고 G 의 모든 정점들을 포함하는 트리를 말한다. 먼저 그래프 G 가 무방향 연결 그래프라고 가정하자. **단절점(articulation point)**이란 그래프 G 의 정점들 중에서 그 정점을 그 정점에 부속한 모든 간선들과 같이 삭제하면 최소한 두 개의 연결 요소를 갖는 그래프 G'를 생성하는 정점 v 를 말한다. 그리고 **이중결합 그래프(biconnected graph)**란 단절점을 갖지 않는 연결 그래프를 말한다. 연결 무방향 그래프 G 에서 **이중결합 요소(biconnected component)**란 최대 이중결합 부분그래프(maximal biconnected subgraph) H 를 말한다. 여기서 '최대'란 그래프 G 에는 이중결합이면서 H 를 완전히 포함하는 부분 그래프가 존재하지 않는다는 것을 의미한다. 다수의 응용에서 단절점의 존재는 바람직하지 못하다. [그림 2]이 관리되는 네트워크 도메인이라고 가정하면, 정점은 라우터나 스위치 등의 네트워크 장비들이 되고 간선은 링크가 될 것이다. 이 때 단절점에 해당하는 장비에 이상이 생겼다고 하면, 그 결과는 하나의 장비만이 아닌 여러 장비들간의 통신의 단절을 초래하여 폴링을 수행할 수 없게 된다. 따라서 제안하고자 하는 기법은 먼저 폴링 노드 리스트와 네트워크 토폴로지 정보를 이용하여 단절점의 존재유무를 통해 그래프를 이중결합 요소로 분리하는 과정을 포함하여야 한다.

연결 무방향 그래프 G 의 이중결합 요소들은 G 의 깊이 우선 신장트리를 이용하여 구할 수 있다.



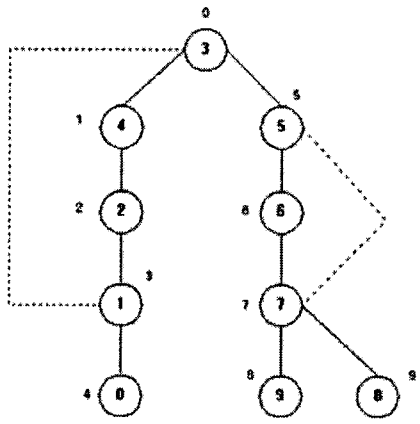
[그림 2] 연결 그래프와 이중결합 요소들

예를 들어, [그림 2(a)]의 그래프에서 폴링 서버가 정점 3 에 있다고 가정하면 dfs(3)을 호출하여 [그림 3]의 신장 트리가 만들어 진다. 그림에서 정점 바깥의 번호는 깊이 우선 탐색의 순서를 나타내며, 여기서는 깊이 우선 번호 dfn 이라고 하자. 일반적으로 두 개의 정점 u, v 에 대해 깊이 우선 신장 트리상에서 u 가 v 의 조상이면 dfn(u) < dfn(v)이다. [그림 3]에서 점선은 비트리 간선을 나타낸다. u 가 v 의 조상이거나 v 가 u 의 조상인 경우 비트리 간선 (u, v)는 백 간선(back edge)이 되는데, 깊이 우선 탐색의 정의에 따라 모든 비트리 간선은 백 간선이 된다. 이는 최소한 두 개의 자식을 갖는 깊이 우선 신장 트리의 루트는 단절점이 됨을 의미한다. 또한, 정점 w 와 w 의 후손들과 단 하나의 백 산선으로만 구성된 경로를 이용해서는 u 의 조상에 도달할 수 없는 그런 정점 w 를 적어도 하나의 자식 정점으로 갖는 정점 u 도 단절점이 된다. 이러한 사실을 통해 G 의 각 정점에 대해 low 의 값을 다음과 같이 정의할 수 있다.

$$low(u) = min\ \{\ dfn(u),\ min\ \{low\ (w) | w \text{ 는 } u \text{ 의 자식}\},\ min\ \{\ dfn(v) | (u,v) \text{는 백 간선}\}\}\quad (1)$$

즉, low(u)는 u 의 후손들과 많아야 하나의 백 간선으로 된 경로를 이용해 u 로부터 도달할 수 있는 가장 적은 깊이 우

선 번호이다. 그러므로, 정점 u 가 두 개 이상의 자식을 갖는 신장 트리의 루트이거나 , 루트가 아니면서 low(w)≥ dfn(u) 를 만족하는 자식 w 를 갖게 되면 단절점이 된다. 따라서 [그림 2]의 (b)에서는 3, 1, 7 의 단절점을 찾을 수 있으며 이를 기준으로 그래프를 이중결합 요소들로 분리하였다.



[그림 3] 깊이 우선 신장 트리

지금까지는 관리 도메인의 각 노드들을 이중결합 요소로 분리하는 과정에 대해 기술하였다. 이러한 과정을 거쳐 분리된 이중결합 요소들은 초기에 방문할 노드에 인접한 순서에 의해 정렬이 되고, 각 이중결합 요소 내에서는 너비 우선 탐색 순서로 정렬을 하면 최종적으로 수행해야 할 폴링 리스트가 얻어진다.

다음으로 얻어진 폴링 리스트로부터 실제 연속적인 폴링을 수행하는 방법에 대해 알아보겠다. 앞의 과정은 실질적으로 폴링을 수행하기 전 단계인 초기화 단계에 해당된다. 다음 단계는 초기화 단계의 결과로써 얻어진 리스트의 처음 노드부터 폴링을 수행하는 단계이다.

```
    Rj : j 번째 라운드
    Pj : j 번째 폴링 리스트
    isArticulationPoint : TRUE 혹은 FALSE 값을 가지는 Boolean
    타입의 변수

1   BEGIN
2   N : 이중결합 요소들의 수
3   각 이중결합 요소 Bi에 대해 i=0...N-1
4   B0 : root 노드를 포함하는 이중결합 요소
5   vi : Bj 에서의 root 노드. 즉, 단절점
6
7     Bi로부터 단절점인 노드(root 노드) vi를 선택
8     BFS_POLLING(vi) ; BFS 순으로 폴링을 수행
9   END
                        (a)

    In BFS_POLLING(vi),
1   BEGIN
2     POLLING(vnext)
3     if vnext 가 폴링이 되지 않을 경우
4     then
5       isArticulationPoint <- whether vi == vnext
6     end
7     if isArticulationPoint == TRUE
8     then
9       if vi ∈ Bprevious 인 B 가 존재
10      then
11        Pj+1 에서 Vi 를 제거
12      end
13      vi 를 제외한 현재의 Bi 의 모든 원소와 현재의 Bcurrent 의
14      원소가 i > current 를 만족하는 Bi 의 단절점이 되는 모
15      든 Bi 를 Pj+1 에서 제거
16    else
17      vnext 와 vnext 가 i > current 를 만족하는 Bi 의 단절점이
18      되는 모든 Bi 를 Pj+1 에서 제거
19    end
20  END
                        (b)
```

[그림 4] 동적 폴링 기법

제안된 동적 폴링 기법에서는 노드 리스트 P_i 로부터 root 노드가 포함된 B_0 부터 BFS 순서로 폴링을 수행한다([그림 4] 참고). 만약 폴링을 수행하는 도중, 네트워크 혼잡 혹은 장비의 DOWN 등의 이유로 timeout 이 발생할 경우에는 현재 폴링되는 노드(v_{next})가 $B_{current}$ 의 root 노드(즉, 단절점)가 되는지 검사한다. 만약 단절점일 경우에는 v_i 를 포함하는 $B_{previous}$ 가 존재하는지 검사한 후 존재한다면 해당 v_i 를 제거하고 현재의 $B_{current}$ 의 원소가 i > current 를 만족하는 B_i 의 단절점이 되는 모든 B_i 를 다음 번 라운드의 리스트(P_{j+1})에서 제거한다. 그렇지 않은 경우 v_{next} 가 다른 이중결합 요소의 단절점이 되는지 검사한 다음, 단절점인 경우에는 해당 B_i 를 다음 번 라운드(R_{j+1})의 폴링 리스트(P_{j+1})에서 제거한다. 제거된 원소들은 그 다음 라운드(R_{j+2})의 폴링 리스트에 추가되어 해당 노드에 대해 다시 폴링을 수행한다.

4. 혼잡 매트릭(Congestion Metric)

앞에서 우리는 폴링 간격이 관리 도구에서 명시적으로 지정된다고 가정하였다. 그러나 연속해서 폴링을 수행할 경우에는 네트워크의 상황에 따라 폴링 간격을 적절하게 변화시키는 것이 바람직하다. 최대 폴링 간격(r_{max})은 제안된 기법에서 두 개의 연속적인 폴링을 수행하기 위한 최소 시간 간격으로 정의할 수 있으며, 이 값은 폴링을 위한 서브넷 대역폭(일반적으로 링크 용량의 5%[1]), 관리 정보의 시간에 따른 변화율[7], 현재 동작중인 최대 노드들의 수 등에 영향을 받는다. r_{max} 를 이용하여 한 라운드 동안의 시간을 계산할 수 있으며, 이는 네트워크가 혼잡 상태일 경우, 폴링 트래픽이 네트워크에 과중한 부하를 주지 않도록 하기 위한 역할을 한다.

D_j의 계산

D_j 는 R_j 에서의 폴링 수행 시간을 의미하며 다음과 같이 계산된다.

$$T_j = r_{max}^{j-1} \times n_{active}^{j-1} \quad (2)$$

$$D_j = T_j + E_j \quad (3)$$

D_j 는 라운드(j-1)에서의 UP 상태인 노드들의 수에 r_{max} 를 곱한 값이 되며, 라운드(j+1)은 timeout 이 발생한 노드의 존재 유무에 따라 폴링 간격을 확장하거나 축소시킨다. 이것과 관련된 파라미터 E_j 는 식(4)와 같이 계산한다. 식(4)에서 $n_{timeout}$ 은 timeout 이 발생하여 폴링 간격에 영향을 주는 노드 수를, k 는 timeout 횟수를 의미한다. 예를 들어 [그림 2]에서 정점 3 이 폴링의 시작점이라고 한다면, 정점 1 과 7 에서 timeout 이 발생한다면 이와 연결된 정점 0, 8, 9 에 대해서도 timeout 이 발생하게 되지만 이 노드들에 대해서는 폴링하지 않아도 되므로 $n_{timeout}$=2 가 된다. 다른 예로 정점 2,4 에서 timeout 이 발생한다면 이 경우 같은 $B_{current}$ 에 속하므로

$n_{timeout}$=2 가 된다.

E_j의 계산

$$E_j = 2^k \times n_{timeout}^{j-1} \quad (4)$$

식(2)와 (4)에 의해 식(3)은 다음과 같이 정리될 수 있다. 식(5)는 R_j 의 수행 시간을 계산한 것이다. 이 값이 실제 처리 시간($D_{current}$)보다 작으면 네트워크가 혼잡상태라고 판단하고 혼잡을 피하기 위해 R_{j+1}의 수행시간을 확장한다.

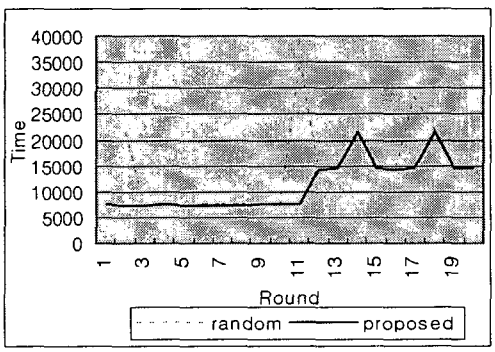
$$D_j = r_{max}^{j-1} \times n_{active}^{j-1} + 2^k \times n_{timeout}^{j-1} \quad (5)$$

5. 성능 평가

3 장에서 제안된 기법의 이론적 수행 시간은 식(6)과 같으며, 최악의 경우 식(7)과 같은 수행 시간을 갖는다. N_B는 B_i의 원소들의 수이며, N_{bi} 는 이중결합 요소들의 수, 그리고 τ_p는 R_j에서의 폴링 수행 시간을 나타낸다.

$$O(\sum_{j=1}^{N_B} N_B^j(\tau_p + \log_2 N_B^j)) \le O(\frac{n^2}{N_{bi}}(\tau_p + \log_2 n)) \quad (6)$$

[그림 5]에서는 이러한 수행 시간을 기반으로 임의 순서 폴링 기법과의 성능을 비교하였다. 여기서 성능 파라미터는 라운드 당 수행 시간이다. 그림을 보면 보통의 경우 두 기법이 비슷한 성능을 나타내나 timeout 이 발생할 경우, 즉 네트워크가 혼잡한 상태일 경우에 제안된 기법이 더욱 효율적이라는 것을 알 수 있다. 20 라운드를 기준으로 수행시간이 34%감소하였으며, 총 폴링 메시지는 임의 폴링 기법이 282 회임에 비해 제안된 기법은 252 회로 상대적으로 적었다.



[그림 5] 임의 폴링 기법과의 성능 비교

6. 결론

이 논문에서는 네트워크 모니터링을 위해 기존의 기법들과는 다른 네트워크 토폴로지를 기반으로 한 새로운 접근 방법을 통해 보다 효율적으로 폴링 트래픽에 대한 오버헤드를 제어하는 동적 폴링 기법을 제시하였다. 제안된 기법에서는 폴링 트래픽에 대한 오버헤드를 제어하기 위해 관리 도메인의 네트워크 토폴로지 정보를 이용하였다. 폴링을 위한 노드 리스트에서 각 노드들 간의 연결 관계를 표현하기 위한 방법을 제시하였으며, 이를 통해 토폴로지 순서에 의해 빠르고 정확하게 실시간 네트워크 정보를 파악할 수 있다. 제시된 기법은 폴링 처리율을 높임으로써 기존의 제안된 기법들보다 효율적으로 동작하며, 이것을 다른 기법들과의 비교를 통해 증명하였다. 실제 네트워크 관리 도구에 이를 적용시키는 것은 차후 연구과제로 남겨둔다.

참고문헌

[1] J. Case, M. Fedor, M. Schoffstall, J. Davin, "A Simple Network Management Protocol(SNMP)", RFC 1157, Network Working Group, IETF, 1990.

[2] Moghe, P.; Evengelista, M.H., "RAP-rate adaptive polling for network management applications", In Proc. Of IEEE NOMS 98, pp.395-399, 1998

[3] K. Ohta, N. Sun, G. Mansfield, and Y. Nemoto, "Effective polling control for network management using SNMP", IEICE Technical Reports on Information Netowrk, IN94-135:91-96, Nov. 1994

[4] B. Bondi, "A nonblocking mechanism for regulating the transmission of network management polls", In Proc. Of IFIP/IEEE IM'97, pp.565-580, 1997

[5] P. Dini, G. Bochmann, T. Koch, and B. Kranmer, "Agent based management of distributed systems with variable polling frequency policies", In Proc. Of IFIP/IEEE IM'97, pp.553-564, 1997

[6] P. Dini and R. Boutaba, "Deriving variable polling frequency policies for pro-active management in networks and distributed systems", In Proc. Of IFIP/IEEE IM'97, pp.541-552, 1997

[7] Yoshihara, K.; Sugiyama, K.; Horiuchi, H.; Obana, S., "Dynamic polling scheme based on time variation of network management information values", Integrated Network Management, 1999. Distributed Management for the Networked Millennium. Proceedings of the Sixth IFIP/IEEE International Symposium on, 1999, pp.141-154