제36회 한국정보처리학회 추계학술발표대회 논문집 제18권 제2호 (2011. 11)

자원 분할수용을 통한 가상네트워크 임베딩 성능 향상

하지훈, 박용태, 김효곤*, 김은아, 양선희†
*고려대학교 정보통신대학 컴퓨터·전파통신공학과
† 한국전자통신연구원
e-mail:{jhha85, ytpark, hyogon}@korea.ac.kr*
{eakim, shyang}@etri.re.kr†

Improving Virtual Network Embedding Performance through Resource Splitting

Jihun Ha, Yongtae Park, Hyogon Kim*
Eunah Kim, Sunhee Yang†
*College of Information & Communication, Korea University
† Electronics and Telecommunications Research Institute

요 약

기반 네트워크 (substrate network)의 자원 여분이 새로 삽입(embed)하고자 하는 가상 네트워크의 자원 요구량을 수용할 수 없을 때, 삽입하고자 하는 가상 네트워크의 요구 자원량을 분할하여 분산 수용함으로써 삽입을 가능케 할 수 있다. 그러나 이러한 작업을 위해서는 각 가상 네트워크의 자원간 상관관계를 꼭 알아야 한다. 이 논문에서 각 가상 네트워크의 명세에 자원 사용 패턴에 있어서의 상관관계를 입력 받음으로써 기반 네트워크의 사용률(utilization)과 가상 네트워크 수용률(acceptance ratio)을 높일 수 있음을 보인다.

1. 서론

가상화된 네트워크 (virtualized network)에서는 기반 네트워크 (substrate)에 독립적인 다수의 가상네트워크 (virtual network)들을 삽입(embedding)할 수 있다. 그러나 가상 네트워크 삽입은 NP-hard 문제라는 것이 널리 알려져 있다.[1] 따라서 최적 알고리듬 대신 사용할 수 있는 다양한 알고리듬이 제안되고 있다.[1][2][4][5][7][10] 가상 네트워크 삽입은 노드 삽입과 링크 삽입 부분으로 구분된다. 노드 삽입에서는 CPU나 디스크공간과 같은 노드 자원을, 링크삽입에서는 대역폭과 같은 링크 자원을 기반 네크워크로부터 할당해야 한다. 각각의 할당 방법에 따라 삽입할 수 있는 가상 네트워크의 수와 같은 효율이 달라질 수 있다.

이 논문에서는 새로 삽입하고자 하는 가상 네트워크가 요구된 만큼의 노드 자원 혹은 링크 자원의 할당이 요구된 위치에서는 불가능한 상황에서, 이 가상 네트워크의 요구 자원량을 분할(split)하여 사용자가 원래 요구하지 않았던 여러 위치에 분산 수용하는데 있어서의 문제를 다룬다. 물론 이러한 자원 위치의 변화는 사용자에게 투명하게 이루어져야 하며, 사용자가 요구한 가상 네트워크 토폴로지를 위배해서는 안되고, 또한 기반 네트워크의 오버헤드 발생을 감수해야 한다.

* 본 연구는 방송통신위원회의 차세대통신네트워크 원천기술개발사업의 연구결과로 수행된 결과임. (KCA-2011-10913-05003)

지금까지의 가상 네트워크 연구에서는 오직 링크 자원인 대역폭의 분리 수용이 고려되어 왔다. 대역폭 요구량이 너무 커서 지정된 위치에 수용할 수 없는 경우, 이를 쪼개어 여러 링크로 분산하되 출발과 도착 노드는 같게 하는 것인데, 노드에서의 라우팅 메커니즘을 약간 수정함으로써 가능하다. 그러나 노드 자원인 CPU 처리용량이나 디스크 공간의 경우는 분할에 따른 스케쥴링 문제가 발생할 수도 있기 때문에 항상 가능하다고 보장할 수 없다. 이런 이유로 노드 자원 분할을 통한 가상 네트워크 임베딩 성능 향상에 대한 연구는 거의 전례가 없다. 이 논문에서 일단 노드 자원 분할에 의한 스케쥴링 문제가 없는 최선의 경우에 과연 노드 및 링크 자원의 분할과 재배치가 가상 네트워크 임베딩 성능에 어떤 긍정적인 효과를 가져올 수 있는지 알아보기로 한다.

2. 자원의 분할 및 분산수용 문제

2.1 자원 간 상관계수

사용자가 가상 네트워크 V_1상의 한 자원 R_1의 위치를 L_1으로 지정하여 $|R_1(L_1)|$ 만큼 요구하였다고 하자. 그리고 L_1에서의 R_1의 가용 용량이 $C(L_1,R_1)$이라고 하자. 만일 $C(L_1,R_1)>|R_1(L_1)|$ 이라고 하면 문제없이 수용할 수 있지만 아니라고 하면 기반 네트워크 운용자는 사용자의 네트워크 V_1의 수용을 아예 거절하거나, 이미 삽입된 다른 가상

네트워크들 {V_n}의 재배치를 통해 L_1에서의 자원의 가용량을 증가시키거나, L_1 대신 다른 위치들 {L_k}에 대신 수용하는 식으로 대응할 수 있다. 이 논문에서는 세 번 째 가능성을 탐색한다. 이 분할 및 분산 수용이 가능한지를 판단하기 위해서 필수적인 것은 다른 자원 R_i의 요구량이 추가로 발생하지 않을지, 발생한다면 어느 정도인지를 판단하는 것이다. 만일 그러한 고려 없이 R_1의 분할 및 분산 수용만을 시도한다면 결과적으로 V_1은 물론이려니와, 이미 삽입된 다른 가상 네트워크들이 추가로 발생한 자원 요구량에 의해서 애초에 요구한 조건을 위배하게 될 수 있다.

이러한 문제를 피하려면 가상 네트워크의 자원 간 상관관계를 꼭 알아야 한다. 즉, 위의 예에서 R_1이 CPU 계산 용량이라 하고 R_2는 대역폭이라 하자. R_1이 {$R[L_1]$, $R[L_2]$, ... , $R[L_k]$}으로 분할된 후 각각 {L_1, L_2, ... , L_k}에 분산 수용되었다면 L_2에 수용된 $R[L_1]$을 사용하기 위해서 사용자 가상 네트워크는 $L_1 \rightarrow L_2$ 경로의 길이에 비례하는 비용을 지불해야 한다. 예를 들어 이 비용은 계산 입력과 출력을 이동시키는데 필요한 대역폭일 수 있다. $L_1 \rightarrow L_2$ 경로상의 각 링크에서 추가로 소요되는 대역폭은 자원 간 상관관계 $COR(R_1 \rightarrow R_2)$와 분산 수용된 자원량 $R[L_2]$의 곱으로 표현될 수 있다. 즉 $L_1 \rightarrow L_2$ 경로상의 각 링크에서 $COR(R_1 \rightarrow R_2) \bullet R[L_2]$의 자원량이 추가로 필요하다.

여기서 문제는, 자원 간 상관관계를 기반 네트워크 제공자가 미리 알기 힘들다는 것이다. 예를 들어 위의 CPU 처리 용량 분산 수용의 예에서 원격으로 계산을 수행하는데 필요한 대역폭은 사용자의 사용 패턴에 의해 결정되는 것으로서 가상 네트워크의 할당 시에는 미리 알기 어렵다. 따라서 이 자원간의 상관관계는 가상 네트워크의 용도에 대한 더 많은 정보를 가진 사용자가 명세하는 것이 바람직하다. 따라서 이 논문에서 사용자가 분할 수용의 경우 사용해야 할 자원 간 상관계수, 즉 위의 예에서는 CPU→대역폭 상관 계수를 명세를 가상 네트워크 삽입 이전에 기반 네트워크 사업자에게 제출하는 것을 가정한다. 아래에서 이 상관 계수들은 행렬 D로 주어진다고 가정하며, $D_{ij} = COR(R_i \rightarrow R_j)$라고 하자.

2.2 가상 네트워크 임베딩

가상 네트워크의 노드 {V_1, V_2, ... , Vn}과 이를 연결하는 링크를 각각 기반 네트워크의 노드 {S_1, S_2, ... , Sn}과 링크에 삽입하는 알고리즘은 다음과 같다.

먼저 가상 노드 V_1이 할당될 기반 노드는 사용자에 의해 요청된 위치 L_1에 존재하는 기반 노드 S_1과 S_1에 직접 연결된 노드 $\{S_c\} = \{S_1, S_2, \ldots, S_m\}$을 후보군으로 한다(줄5). 이때 후보군 중 CPU 자원이 가장 많은 노드부터 내림차순으로 정렬한 후 순서대로 노드 할당을 시도한다(줄6~15). 만약 후보군에 V_1의 노드 자원을 수용할 수 있는 기반 노드를 찾으면 선택하지만(줄7~11) 모든 후보 노드가 자원이 모자라 V_1의 요청을 받아들일 수 없는 경우 노드 자원 분할을 시도한다(줄17~31). 우선 정렬된 후보 기반 노드들 중 남는 노드 자원이 있는 기반 노드(S_i)를 선택한다. S_i로부터 가장 가까우면서 자원이 많이 남은 기반 노드부터 순차적으로 검색하여 분할된 노드 자원의 수용 여부를 판단한다(줄19~30). 이 때 선택된 노드(S_j)와 S_i의 자원 간 상관관계에 의해 두 노드를 연결 하는데 $D_{ij} \bullet R[L_j]$ 만큼의 대역폭이 추가로 사용된다.

<알고리즘 1> 가상 네트워크 임베딩 알고리즘

```
1:  function embed_network(Vk)
2:      if (k > n) then
3:          return SUCCESS; // Embedding finish.
4:      // sort by residual capacity of substrate node
5:      {Sc} = sort(getCandNodes(Vk))
6:      for i=1 to m // # of elements in {Sc}
7:          if (embed_virtual_node(Vk, Sc[i]) ==
8:  SUCCESS && embed_virtual_link(Vk) ==
9:  SUCCESS && embed_virtual_network(Vk+1) ==
10: SUCCESS) then
11:             return SUCCESS;
12:         else
13:             disembedding();
14:         end if
15:     end for
16:     // need to split the virtual node
17:     for i=1 to m
18:         // the set of candidate nodes for splitting
19:         {Scs} = sort(getCandNodes(Sc[i]) - {Sc[i]})
20:         for j=1 to l // # of elements in {Scs}
21:             Sj = Scs[j];
22:             if (embed_split_node(Si, Sj) ==
23: SUCCESS && embed_split_node_link(Si, Sj) ==
24: SUCCESS && embed_virtual_network(Vk+1) ==
25: SUCCESS) then
26:                 return SUCCESS;
27:             else
28:                 disembedding()
29:             end if
30:         end for
31:     end for
32:     return FAIL
33: end function
```

이어서 {V_2, ... , Vn}을 위와 같은 방법으로 할당하되 각 노드(V_k) 를 할당한 후에 V_k와 {V_1, ... ,V_{k-1}}를 연결하는 가상 링크가 존재하는지 검사한다. 가상 링크를 할당할 때에는 링크에 연결된 두 가상 노드가 삽입된 기반 노드 S_k, S_l을 연결하는 기반 링크를 찾아 가상 링크가 필요로 하는 대역폭을 할당 한다. 이 때 S_i와 S_j 사이에 요청된 대

역폭을 만족하는 경로는 최대 유량 문제를 구하는 Edmonds-Karp 알고리듬[7]을 이용하여 두 노드(S_k, S_l) 사이의 가능한 모든 경로 중 주어진 대역폭을 만족하는 경로의 집합을 구하여 가장 적게 대역폭이 소모되는 경로로 한다(줄8, embed_virtual_link). 위와 같은 과정을 통하여 V_k와 {V_1, ... ,V_{k-1}} 사이에 존재하는 가상 링크 할당이 모두 완료되면 가상 노드 V_{k+1}의 할당을 시도한다(줄9 또는 24, embed_virtual_network).

3. 성능 평가

3.1 실험환경

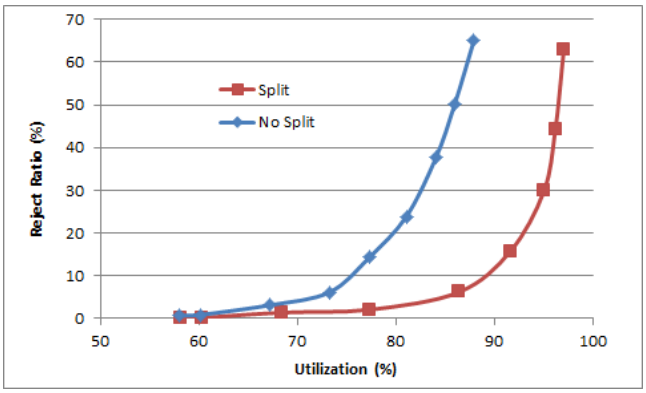
실험에 사용한 파라미터 값은 기존 연구에서 사용한 것과 유사하며, <표 1>에 나타내었다. 먼저 기반 네트워크(SN)의 크기는 노드가 50개이며 연결성은 heavy-tailed 특성을 지니도록 Power-law 분포(exponent ϒ=3)를 사용하였다.[11][12] 각 노드에서 최대 가용 CPU 용량은 uniform 분포로 50 ~ 100단위의 값을 가지며, 링크에서도 최대 가용 대역폭은 50 ~ 100 단위라고 가정하였다.[4] 사용자로부터 주어지는 가상 네트워크(VN)는 Waxman 토폴로지 생성 알고리즘을 사용해 랜덤 토폴로지를 구성하였다.[13] 가상 네트워크 크기는 비교적 작아 2 ~ 5개의 노드를 가지며, 각 노드는 10 ~ 20 단위의 CPU 용량과 역시 10 ~ 20 단위의 대역폭을 요구한다고 가정하였다. 기반 네트워크에 들어오는 가상 네트워크 임베딩 요청간의 시간은 exponential하게 5 ~ 25 시간단위 범위에 펼쳐져 있으며, 기반 네트워크에 설치되어 사용되는 시간 단위는 역시 exponential 분포를 가지며 평균 1000 시간단위이다.[4] Little의 법칙에 의하면 steady state에서 기반 네트워크 상에 평균적으로 존재하는 가상 네트워크의 숫자는 $N=\lambda T$ 이므로 λ 값에 따라 40 ~ 200 개 범위라고 할 수 있다. CPU자원과 대역폭자원 간의 상관관계는 0.2라고 가정하였다. 이 상관관계 값이 크면 클수록 분산 수용의 오버헤드가 커져 분산 수용이 어려워지는 특성이 있다.

<표 1> 실험 파라미터

파라미터	값
SN 노드 수	50
SN 연결성 파라미터	Power-law distribution with exponent ϒ=3
SN 노드 최대 CPU 가용자원	U[50,100]
SN 링크 최대 대역폭 가용자원	U[50,100]
VN 노드 수	U[2,5]
VN 토폴로지 생성 파라미터	Waxman random topology generator (α = 0.15, β = 0.2)
VN CPU 요구자원	U[10,20]
VN 대역폭 요구자원	U[10,20]
VN 임베딩 요청간 시간 (1/λ)	[5,25] increment of 2.5, exponential
VN 지속 시간 (1/μ)	1000 exponential
CPU-대역폭 상관관계	$D=\begin{bmatrix}0 & 0.2\\ 0.2 & 0\end{bmatrix}$

3.2 실험결과

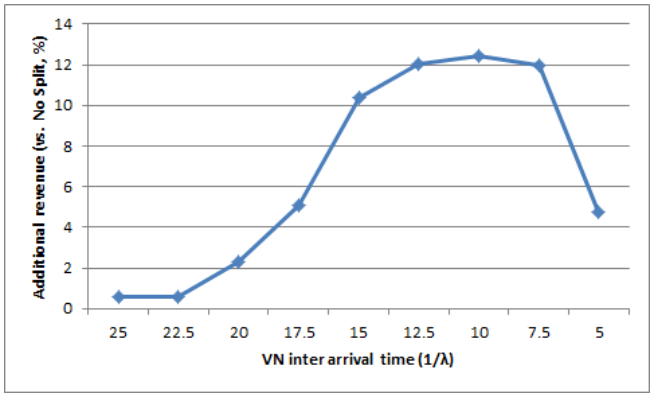
위와 같은 파라미터 값으로 시스템을 steady state에서 운용하였을 때 (그림 1), (그림 2)의 결과를 얻었다. (그림 1)은 기반 네트워크 활용률(utilization) 대비 임베딩 실패(rejection) 비율이다. 이 실패비율은 어떤 사용자가 VN 임베딩을 요구하였을 때 기반 네트워크 공급자가 임베딩에 실패하여 거절하게 되는 비율이다. 그림에서 x축이 네트워크 활용률로서 전체 기반 네트워크 자원 중 어떤 가상 네트워크에 할당되어 있는 자원의 비율이다. 이 논문에서 제안하는 분할 수용 알고리듬을 사용하기 전에는 네트워크 활용률이 60%를 넘어가기 시작하면 거절이 발생하기 시작하여 90% 근처에 가게 되면 대부분의 가상 네트워크 임베딩 요청을 거절하게 된다. 그러나 분할 수용 알고리듬을 가동하게 되면 80% 이상의 네트워크 사용률 이후에 거절이 발생하기 시작하고 90% 중반 이상의 네트워크 사용률 하에서만 거절이 심각하게 발생하는 것을 알 수 있다. 이와 같이 분할 수용 알고리듬을 사용하면 기반 네트워크 제공자의 입장에서는 기반 네트워크 자원들을 거의 낭비 없이 운용하면서도 사용자의 요구를 거의 항상 수용할 수 있게 됨으로써 더 큰 수익을 기대할 수 있으며, 반대로 사용자의 입장에서는 거절율이 낮아져 만족도가 높아질 수 있다는 장점이 있다.



(그림 1) 분할 수용에 의한 임베딩 실패율 비교

(그림 2)는 분할 수용 알고리듬을 사용하지 않았을 때와 비교하였을 때 분할 수용 알고리듬이 추가적으로 발생시키는 revenue를 보이고 있다. 여기서 revenue는 가상 네트워크에 할당되어 사용되는 자원량을 말한다. (그림 1)에서 이미 시사된 것처럼, 분할 수용 알고리듬은 더 많은 자

원이 가상 네트워크들을 위해 사용되도록 만들어준다. 그림에서 x축은 VN 임베딩 요청 간 시간 (1/λ)으로서 오른쪽으로 갈수록 더 많은 요청이 들어오는 상황을 뜻한다. 결과를 보면 요청이 들어오는 시간 간격이 좁아짐에 따라서 처음에는 추가적인 revenue가 점차로 증가하다가 1/λ =10에 이르게 되면 약 12%로 정점에 다다르게 된다. 그러다가 더 높은 속도로 요청이 쇄도하게 되면 분할 수용을 통해서도 받아들일 수 있는 양이 한계에 이르면서 상대적인 차이가 줄어드는 모습을 보인다.



(그림 2) 분할 수용에 의한 추가적인 revenue

4. 결론

이 논문에서 가상 네트워크의 임베딩이 요구된 위치의 자원의 부족으로 불가능할 때, 내부적으로 노드 및 링크 자원을 분할하여 수용함으로써 얻어질 수 있는 이득에 대해 보였다. 이를 통해 기반 네트워크 제공자는 이미 투자된 네트워크 기반시설의 활용도를 높일 수 있으며, 가상 네트워크 사용자에게는 훨씬 낮은 거절율을 제공할 수 있다. 그러나 이러한 분할 수용이 가능하려면 가상 네트워크 사용자로부터 그 네트워크 상의 자원 사이에 어떤 상관관계가 있는지 입력을 받는 것이 중요하다. 앞으로 이 상관관계를 보다 정확히 명세할 수 있도록 하는 방법에 대해 연구할 계획이다. 또한, 이 논문에서 관찰되는 성능 개선폭은 입력되는 가상 네트워크의 자원 요구량의 균일도에도 영향을 받는다. 따라서 이 주제에 대한 연구도 수행할 계획이다.

참고문헌

[1] N.M.M.K. Chowdhury, M.R. Rahman, and R. Boutaba, "Virtual Network Embedding with Coordinated Node and Link Mapping", in Proc. INFOCOM, 2009.

[2] N.F. Butt, N.M.M.K. Chowdhury, and R. Boutaba, "Topology-Awareness and Reoptimization Mechanism for Virtual Network Embedding", in Proc. NetWORKing, 2010.

[3] I. Houidi, W. Louati, D. Zeghlache, P. Papadimitriou, L. Mathy, "Adaptive Virtual Network Provisioning", in Proc. the 2nd ACM SIGCOMM workshop on VISA, 2010.

[4] I. Fajjari, N. Aitsaadi, G. Pujolle, H. Zimmermann, "VNE-AC: Virtual Network Embedding Algorithm Based on Ant Colony Metaheuristic", in Proc. IEEE ICC, 2011.

[5] J. Lischka, H. Karl. "A virtual network mapping algorithm based on subgraph isomorphism detection", in Proc. the 1st ACM workshop on VISA, 2009.

[6] I. Houidi, W. Louati, D. Zeghlache, "A Distributed Virtual Network Mapping Algorithm", in Proc. IEEE ICC, 2008.

[7] C. Edmonds, R.M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems", Journal of the Association for Computing Machinery, Vol. 19, No. 2, p. 248-64. 1972.

[8] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration", in Proc. the ACM SIGCOMM, 2008.

[9] D. Perino, M. Varvello, "A reality check for content centric networking", in Proc. the ACM SIGCOMM workshop on ICN, 2011.

[10] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, J. Wang, "Virtual Network Embedding Through Topology-Aware Node Ranking", in the ACM SIGCOMM Computer Communication Review archive Volume 41 Issue 2, 2011.

[11] A. Medina, A. Lakhina, I. Matta, J. Byers, "BRITE: An Approach to Universal Topology Generation", in Proc. the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems- MASCOTS'01, 2001.

[12] A.L. Barabasi, R. Albert, "Emergence of Scaling in Random Networks", Journal of the American Association for the Advancement of Science pp. 509 - 512, October 1999.

[13] B. Waxman, "Routing of Multipoint Connections", IEEE Journal on In Selected Areas in Communications, Vol. 6, No. 9, pp. 1617-1622, August 2002.