P2P 환경에서 Flooding 과 Random walk 의 다이나믹 알고리즘

정상진, 김재훈
아주대학교

jinpang97@ajou.ac.kr, jaikim@ajou.ac.kr

A Study on the Dynamic P2P Flooding and Random Walk Algorithms

Sangjin Jeong, Jai-Hoon Kim
Ajou University

요 약

본 논문은 플러딩(Flooding)과 랜덤워크(random walk)를 가변적으로 사용하여 두 방법의 장점을 살린 방법을 연구하였다. 초반엔 플러딩을 사용하여 탐색 후 다중 랜덤워크를 사용하였다. 결과적으로 랜덤워크만 사용할 때보다 빠르게 검색이 가능하고 플러딩보다 비용을 적게 하여 더 효율적인 노드 검색을 가능하게 할 수 있음을 분석적으로 확인하였다.

Ⅰ. 서 론

본 논문에서는 노드를 찾거나 메시지를 전달할 때 사용하게 되는 방법인 플러딩(Flooding)과 랜덤 워크(Random walk)의 방법을 다이나믹하게 사용하여 가장 높은 효율의 방법을 찾고자 한다. 우선 플러딩은 인접한 노드 모두에게 메시지를 보내기 때문에 엣지(edges)가 많아질수록 비용이 매우 비싸 진다는 단점을 가지고 있다. 랜덤 워크의 경우 인접한 노드들 중 한 노드에게만 무작위 하게 메시지를 보내기 때문에 목적하는 노드에게 메시지를 보내기 위해서 많은 노드들을 찾아봐야 한다는 단점을 가지고 있다. 하지만 플러딩의 경우 한 번에 많은 노드들에게 메시지를 보낼 수 있기 때문에 적은 횟수 내에 일을 처리할 수 장점이 있으며, 랜덤 워크의 경우엔 적은 비용으로 일을 처리할 수 있다는 장점을 가지고 있다. 따라서 두 장점을 살린 방법을 찾는 것이 이 논문의 목적이다.

Ⅱ. 랜덤 워크와 플러딩 알고리즘 비교

본 논문에서는 단계를 구분하여 첫 단계에선 플러딩 방식을 이용하고 그 다음 단계에선 랜덤 워크방식을 이용하고자 한다. 우선 랜덤 워크와 플러딩에서 데이터를 찾을 확률을 그래프를 통해서 알아보자. 이 때 전체 노드의 수(N)는 10000 개로 가정하였고, 정보가 존재하는 노드의 수(r)는 10 개로 가정하였다. 플러딩에서 필요한 한 노드에 연결된 엣지의 수는 4 (d) 로 가정하였다. 마지막으로 k 는 실행 횟수이며, 원하는 노드를 찾을 때까지 진행되는 횟수는 D 이다. 이렇게 설정했을 때, 랜덤 워크는 100 번 실행하는 성능을 그래프로 표시하였고, 플러딩의 경우엔 7 번 진행된 경우를 그래프로 표시하였다. 이 외에도 랜덤 워크를 진행할 때 한 번에 하나의 노드에서만 시작하는 것이 아니라 n 개의 노드에서 시작하여 원하는 노드를 찾는 방법인 $n-$ 랜덤 워크도 $n=2, n=4$ 인 경우도 같이 찾아보았다.

이에 따라 랜덤 워크를 통해 원하는 노드를 찾을 때의 확률 ($P(k)$) 을 구하는 수식과, $n-$ 랜덤 워크를 통해 원하는 노드를 찾을 때의 확률 ($P_n(k)$) 그리고 플러딩을 통해 원하는 노드를 찾을 확률 ($R(k)$) 구했다. 플러딩과 랜덤 워크에 대한 성능분석은 [1,3]에 상세히 설명되었다. 본 논문에서는 이를 바탕으로 n-랜덤워크의 성능을 분석하였고 이를 플러딩과 비교하고, 다이나믹 알고리즘을 분석하였다.

우선 이를 구하는 과정부터 설명하도록 하겠다. $P(k)$의 경우, 한 번에 찾을 확률을 생각해 보자면 전체 노드에서 원하는 노드를 찾는 것이므로 원하는 노드의 수를 전체 노드의 수로 나누면 된다. 이는 첫 번째에 바로 찾을 확률이고 이를 한 번에 찾지 못한다면, 전체 확률인 1 에서 이 값을 뺀 값을 계속해서 곱하는 것이기 때문에 k 번 만에 원하는 노드를 찾을 확률은 $\frac{r}{N}\times\left(1-\frac{r}{N}\right)^{k-1}$ 이 된다. 하지만, 우리가 원하는 것은 랜덤 워크를 진행하여 D 번 만에 찾을 확률의 누적 값을 구하는 것이기 때문에 각 확률을 더한다 [3].

$$P(D)=\sum_{k=1}^{D}\left\{\frac{r}{N}\times\left(1-\frac{r}{N}\right)^{k-1}\right\}$$

다음은 $n-$랜덤 워크를 진행할 때 한 개의 노드에서 탐색을 시작하는 것이 아니라 n 개의 노드에서 탐색을 시작하기 때문에 구할 확률에 n을 곱해줘야 하고, 반대로 그 때 찾지 못한다면 찾지 못할 확률을 그 수만큼 곱해줘야 한다. 따라서 이를 식으로 표현하면 다음과 같다.

$$P_n(D)=\sum_{k=1}^{D}\left\{\frac{nr}{N}\times\left(1-\frac{nr}{N}\right)^{n(k-1)}\right\}$$

다음은 플러딩을 통해 원하는 노드를 구하는 식이다. 플러딩의 경우엔, 자신이 연결되어 있는 엣지의 수만큼 뻗어 나가서 찾을 확률이 존재하고 이 때 찾지 못하면 뻗어 나갔을 때 기준으로 자신을 제외하기 때문에 $(d-1)$ 씩 곱해줘야 한다. 따라서 이를 식으로 표현하면 다음과 같다 [1].

$$R(D)=\sum_{k=1}^{D}\left\{\frac{d(d-1)^{k-1}}{N}\right\}$$

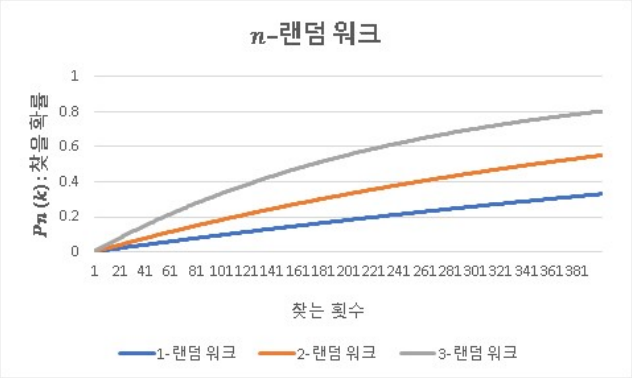

그림 1. $n-$랜덤 워크 알고리즘의 성공률

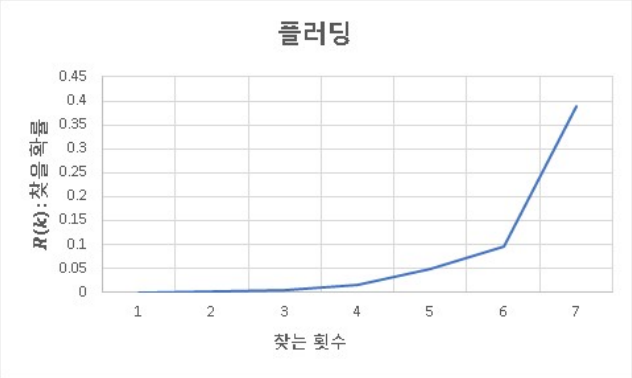

그림 2. 플러딩 알고리즘의 성공률

그림 1. $n-$랜덤워크 그래프 에서와 같이 랜덤 워크의 경우엔 여러 번 진행할수록 누적되어 찾을 확률이 1 차 방정식과 유사하게 증가했다. 2-랜덤 워크에서도 1-랜덤 워크의 약 2 배의 증가율을 보여주며 1-랜덤 워크와 유사한 증가 그래프를 나타냈다. 하지만, k 가 점점 커지게 되면, 이미 찾아봤던 노드를 다시 중복해서 찾을 확률이 증가하기 때문에 1 차 방정식에 유사하게 흘러가던 그래프는 점차 그 수치가 떨어지게 된다.

그림 2. 플러딩 그래프 에서와 같이 플러딩의 경우 찾는 횟수가 지수함수의 그래프를 그리며 급격하게 증가했다. 이 때, 지수함수이기 때문에 k 값이 5 를 초과하게 된다면 비용 및 대역폭의 사용량이 커지기 때문에 플러딩을 효율적으로 사용할 수 없게 된다.

Ⅲ. 다이나믹 알고리즘

본론을 통해 알 수 있는 내용은 랜덤 워크와 플러딩의 찾을 확률 $P(k), R(k)$ 를 구해보았다. 결과만 보았을 땐, 플러딩이 적은 횟수를 통해 찾을 확률이 많아 보이지만, 매우 많은 양의 노드들을 탐색하는 것이기 때문에 밴드위드 소비에 있어 매우 큰 비용을 발생하기 때문에 플러딩을 많이 사용하는 것은 효율적이지 않다. 따라서 이러한 근거들을 토대로 단계적으로 랜덤 워크와 플러딩을 각각 다이나믹 하게 사용한다.

첫 번째 단계에선 플러딩을 사용하고, 두 번째 단계에선 랜덤 워크를 진행하게 된다. 또한 이 때 $n-$랜덤 워크를 진행하게 된다. $n-$랜덤 워크의 경우 n 값이 증가함에 따라 거의 정수비례로 효율이 좋아지기 때문에 기대한 값을 얻을 수 있게 된다. 이를 분석하기 위하여, 반복 회수는 D로 표현하였고, 5 회 플러딩 이후에 $P(5)$ 개의 노드에 메시지가 전파되고, 각 노드가 랜덤 워크로 동작을 하므로, 이후 $P(5)$-랜덤 워크의 효과가 있다. 따라서 D 회 반복 이후에 해당 데이터를 발견할 확률$(P(D))$은 다음과 같다.

$$P(D) = \begin{cases} \sum_{k=1}^{D}\left[\frac{d\times(d-1)^{k-1}}{N}\right] & (D \le 5) \\ P(5) + \sum_{k=6}^{D}\left[\frac{rn}{N} \times \left(1-\frac{nr}{N}\right)^{n(k-6)}\right] & (D > 5) \end{cases}$$

본론에서 진행했던 조건과 같은 기준으로 진행하였다. 플러딩만 사용할 순 없지만, 이 방법만 사용했을 땐 약 9 번 진행했을 때 찾을 확률이 거의 100%가 넘으며, 실제 상황에선 사용할 수 없기 때문에 플러딩을 최소한으로 사용하면서 가장 효율적인 값을 방법을 찾아야 한다. 또한 플러딩은 5 회를 초과해서 사용하는 것이 효율이 매우 떨어진다. 랜덤 워크만 진행하게 되면, 50%의 확률을 얻기 위해 약 700 번의 탐색이 필요하다. 따라서 목표치를 50%로 설정하고 플러딩과 랜덤 워크를 이용하여 훨씬 적은 탐색과 대역폭을 사용한다. 이를 통해 얻게 된 두 탐색 방법의 다이나믹한 사용 방법을 통해 탐색했을 때의 결과를 그림 5 와 같다.

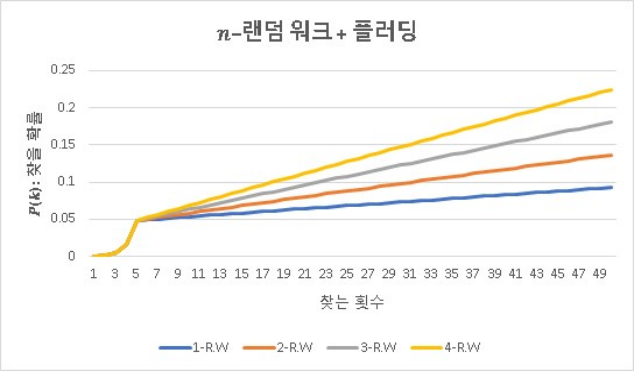

그림 5. 플러딩과 랜덤워크의 다이나믹 그래프

이는 플러딩을 5 회 진행 후 $n-$랜덤 워크를 진행한 그래프이다. n 이 4 일 때 기준으로 약 95 번 만에 50%정도에 도달했기 때문에 단순히 랜덤 워크를 진행했을 때 보다 훨씬 빠르게 수행이 가능하다. 또한 플러딩보다 밴드위드를 훨씬 덜 사용하게 된다.

Ⅳ. 결론

비구조적인 P2P 네트워크에서 노드를 탐색할 때 플러딩을 통해 일정 비율의 노드를 탐색하고, 이후 각 노드들이 동시에 랜덤 워크를 진행하는 n – 랜덤 워크 제안하였다. 적은 통신 대역폭을 이용하여 빠른 탐색이 가능함을 분석적으로 확인하였다.

ACKNOWLEDGMENT

이 논문은 2018 년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (NRF-2018R1D1A1B07040573).

참 고 문 헌

[1] M. Steen and A. Tanenbaum, Distributed Systems 3rd edition (2017), pp. 85-87.

[2] Cohen E. and Shenker S. Replication Strategies in Unstructured Peer-to-Peer Networks. In SIGCOMM, pages 177-190, New York, NY, Aug. 2002. ACM, ACM Press.

[3] Lv Q., Cao P., Cohen E., Li K., and Shenker S. Search and Replication in Unstructured Peer-to-Peer Networks. In 16th International Conference on Supercomputing, pages 84-95, New York, NY, June 2002. ACM, ACM Press.