실시간 환경에서 가용 대역폭과 거리를 고려한 개선된 Kademlia 프로토콜

박재완*, 맹주현*, 이동혁*, 조인휘*
*한양대학교 컴퓨터 소프트웨어학과
e-mail:xp0207@hanyang.ac.kr

An Improved Kademlia Protocol considering Available Bandwidth and Physical Distance in the Real-Time Environment

Jae-Wan Park*, Ju-Hyun Maeng*, Dong-Hyuk Lee*, In-Whee Joe*
*Dept of Computer Software, Hanyang University

요 약

분산 해시 테이블은 {Key-Value} 형태의 해시 테이블을 시스템 내 노드들이 나누어 가지는 분산 시스템이다. 분산 해시 테이블 중 Kademlia는 Binary Tree 구조를 통해 노드 확장성을 가지고 XOR Metric을 이용하여 빠른 노드 탐색으로 다양한 분야에서 활용되고 있다. 하지만 노드 탐색 시 실제 상황을 배제하고 논리적인 거리만을 고려하여 라우팅 경로를 설정한다는 문제점을 가진다. 본 연구에서는 이 문제를 해결하기 위해 노드 탐색 시 노드 간의 대역폭과 물리적 거리를 고려하여 라우팅 테이블을 생성하는 Kademlia의 효율적인 노드 탐색 기법을 제안한다. 기존의 Kademlia와 유사한 수치의 Lookup Success Ratio와 Lookup Overhead Rates를 보이지만, End-to-End Delay가 감소한 것을 시뮬레이션을 통해 확인하였다.

1. 서론

분산 해시 테이블이란 시스템 내의 각 노드가 {Key-Value} 형태의 해시 테이블을 나눠 가지는 분산 시스템을 말한다. 분산 해시 테이블은 크게 키 분할(Keyspace partitioning)과 오버레이 네트워크(Overlay Network)로 두 가지 기능으로 나뉘는데, 해시함수를 통해 생성된 해시 테이블을 키 분할을 통해 시스템 내부의 각 노드들에 분산시킨다. 분산된 해시 테이블의 키 값을 통해 엔트리 포인트에 상관없이 분산 해시 테이블 시스템 내부의 오버레이 네트워크상에서 목적지 위치를 찾아간다[1].

분산 해시 테이블은 해시함수를 통해 네트워크상에 존재하는 다양한 콘텐츠들을 각 노드의 해시 테이블에 분산 배치시켜 빠르고 정확히 검색할 수 있다. 또한 노드의 참가와 이탈을 관리자가 직접 처리할 필요가 없어 관리비용을 줄일 수 있다는 장점이 있다. 이러한 이유로 분산 해시 테이블은 다양한 서비스에서 활용되고 있다. 예를 들면, 분산 해시 테이블 알고리즘 중 하나인 Chord는 CFS(Cooperative File System)에서 Block을 찾기 위한 라우팅 테이블을 유지하는 데 사용되었고, Pastry는 분산 파일 시스템인 PAST와 멀티캐스트 시스템인 SCRIBE에 활용되었다. CAN은 분산 파일 관리 시스템인 OceanStore에서 응용되었다 [1]. 마지막으로 Kademlia는 Fog/Edge Computi-ng의 IPFS(InterPlanetary File System)에서 사용되며, P2P 파일 공유 시스템인 Bittorrent[2], eMule 등에서 응용되고 있다. 또한 블록체인에서는 Ethereum[3]의 노드 탐색 알고리즘으로 활용되고 있다.

2. 관련연구

2.1. 분산 해시 테이블 (Distributed Hash Tables)

분산 해시 테이블을 활용하는 프로토콜은 Ring형 구조, Tree형 구조, Hypercube형 구조로 다양하게 존재하고 있다. 첫 번째로 Ring형 구조에는 Chord[4]가 있다. Chord는 해시함수를 통해 생성된 노드 식별자를 m-bit의 원형 주소 공간에 할당시키고, 노드를 탐색하기 위해서 Finger table이라는 라우팅 테이블을 이용하여 시계방향으로 위치하는 후속 노드들의 테이블 정보를 통해서 목적지 노드를 탐색한다. 두 번째로 Tree형 구조에는 Pastry[5], Kademlia[6], Tapestry[7] 등이 있다. Pastry는 Chord와 같이 원형 주소 공간을 사용하지만, 주소 공간의 크기가 128-bit로 고정되어 있으며, 2진법을 사용하는 Chord와 달리 2b 진법이라는 임의의 진법을 사용한다 [1]. Kademlia는 해시함수를 통해서 160-bit의 선형 주소 공간에 노드를

(표 1) 분산 해시 테이블 기반 프로토콜 분류 및 성능 비교 [9]

	Ring Structure	Tree Structure			Hypercube Structure
DHT Algorithm	Chord	Pastry	Kademlia	Tapestry	CAN
System Parameter	N-number of peers in network.	N-number of peers in nework, b-number of bits (B=2^b) used for the base of th chosen identifier.	N-number of peers in network b-number of bits (B=2^b) of NodeID.	N-number of peers in nework, B-base of the chosen peer identifier.	N-number of peers in network, d-number of dimensions
Routing Performance	$O(logN)$	$O(logN)$	$O(log_BN)$ + c *where* c = *small constant*	$O(log_BN)$	$O(d..N^{1/d})$
Routing State	$logN$	$Blog_BN$+$Blog_BN$	$Blog_BN$+B	log_BN	$2d$
Peers Join/Leave	$(logN)^2$	log_BN	log_BN + c *where* c = *small constant*	log_BN	$2d$

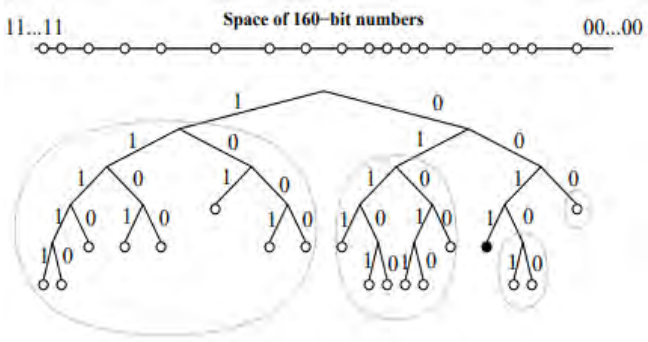
배치시키고, 인접한 노드들의 정보를 XOR Metric을 이용하여 노드를 탐색한다. Tapestry는 네트워크 지역성을 고려하여 탐색 시간을 기준으로 자신에게 가까운 이웃 노드의 정보를 유지하고, 다중 레벨로 구성된 이웃 맵을 구성하고 노드 탐색 시 이를 사용한다. 세 번째로 Hypercube 형 구조에는 CAN(Content Addressable Network)[8]이 있다. CAN은 앞서 설명한 Tree형 구조의 분산 해시 테이블 알고리즘과 다르게 d-차원의 데카르트 좌표계에 기반을 둔 주소 공간을 활용한다. 각 노드의 ID와 Key는 d-차원의 한 점으로 맵핑되고, 각각의 노드는 전체 주소 공간을 균등하게 배분받아 관리하게 된다 [1].

2.2. Kademlia

Kademlia는 분산 해시 테이블을 기반으로 하는 구조화된 시스템의 하나로, 확장성이 용이한 P2P 탐색 프로토콜이다. Kademlia는 각 노드가 시스템 내 존재하는 모든 노드에 대한 정보를 유지하지 않고, 자신에게 인접한 노드에 대한 라우팅 정보만을 유지한다. Kademlia는 160-bit SHA-1 해시함수를 사용하여, 노드 ID(식별자)를 생성하고 시스템 내의 모든 노드를 연결하는 오버레이 네트워크를 구성한다. 이때 Kademlia에 참여하는 노드의 IP주소를 해시함수를 통해 노드 ID로 전환하며, 노드 ID를 할당받은 노드는 자신의 노드 ID와 일치하는 공간에 배치된다.

또한, XOR Metric을 이용하여 간단하고 빠른 탐색 성능을 가진다. 해시함수를 통해 생성된 160-bit의 노드 ID를 사용하여, 오버레이 네트워크의 동일 선상 위에 노드가 정렬된다. 그리고 자신의 노드와 목적지 노드 간의 거리를 XOR Metric을 통해 판단한다. 연산결과 값이 0에 가까울수록 목적지 노드와 가깝다고 판단하며, 반대로 값이 클 때 목적지 노드와 멀다고 판단한다.

Kademlia는 K-bucket을 통해서 라우팅 테이블을 관리한다. K-bucket은 자신에게 인접한 노드의 ID를 Radix Binary Tree 구조 형태로 저장한다. 이러한 K-bucket을 통해서 전체적으로 노드 ID에 대한 Binary Tree를 구조를 만들 수 있으며, 자신의 Sub-Tree를 나타낼 수 있다. (그림 1)은 Kademlia의 Binary Tree 구조를 보여주고 있다. 검정색 점은 0011의 위치를 나타내며, 회색 원은 0011의 Sub-Tree를 나타낸다.



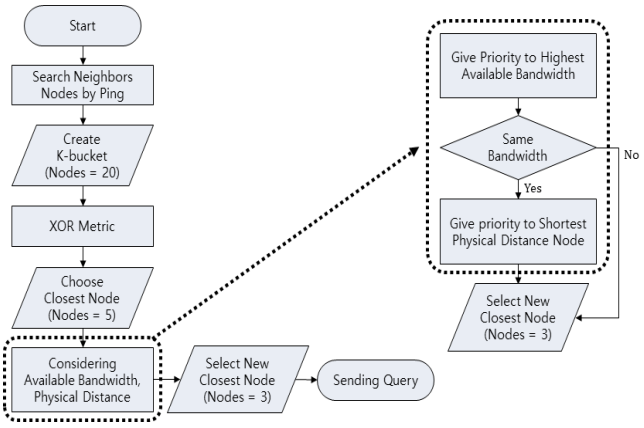
(그림 1) Kademlia Binary Tree 구조 [6]

3. 제안하는 알고리즘

Kademlia는 Binary Tree 구조를 통해서 노드 확장성을 가지며, XOR Metric을 통한 빠른 노드 탐색이 가능하다는 장점을 가지고 있다. 이러한 이점으로 P2P File Sharing, Fog/Edge Computing, 블록체인 등의 다양한 서비스에서 활용된다. 하지만 해시함수를 통해서 생성된 노드 ID를 160-bit 선상 위에 배치하고 노드 탐색 시 XOR Metric을 통해서 노드 간의 논리적인 거리를 판단하기 때문에 실제 물리적인 거리를 고려하지 않게 된다. 이러한 이유로 오버레이 네트워크상에서 논리적인 거리가 가깝다고 해도 물리적인 거리가 가깝다고 판단할 수 없다. 따라서 실제 자신에게 가까운 노드를 탐색하는 시간이 늘어난다는 문제점을 가질 수 있다.

본 연구에서는 Kademlia의 효율적인 노드 탐색을 위해서 (그림 2)와 같이 노드의 대역폭과 노드 간의 물리적인 거리를 고려한 방식을 제안한다. 앞서 설명하였듯이

Kademlia는 논리적 거리를 기반으로 노드 간의 거리를 판단하기 때문에 실제 환경을 고려되지 않는다. 따라서 동적인 네트워크 환경에서 최적 경로가 선정되지 않아 탐색 시간이 늘어날 수 있다. 하지만 제안하는 방식에서는 XOR Metric을 통해 나온 논리적 거리를 기반으로 선택된 노드들의 가용 대역폭과 물리적인 거리를 고려하여 목적지를 찾기 위한 Query를 보낼 노드들을 선정한다. 각 노드의 가용 대역폭과 노드 간 물리적 거리를 활용하여 노드 탐색을 진행하면 노드 탐색 시간을 최소화시킬 수 있다고 기대한다.



(그림 2) 제안하는 Kademlia 노드 탐색 순서도

기존의 Kademlia의 노드 탐색 방식은 K-bucket에서 저장된 자신과 인접한 20개의 노드 정보를 기반으로 시작한다. 목적지 노드를 탐색하기 위해서 자신이 보유하고 있는 K-bucket에 저장된 자신과 이웃한 노드들과 목적지 노드 간의 거리를 판단하기 위해서 XOR Metric을 진행하며, 이때 계산된 거리를 통해서 목적지 노드와 논리적인 거리가 가장 짧은 3개의 노드를 선정하고 해당 노드들에게 Query를 보내게 된다.

본 연구에서 제안하는 방식은 (그림 2)의 순서도에서와 같이 목적지 노드와 논리적인 거리가 가까운 5개의 노드를 선정하고, 선정된 노드들 중 가용 대역폭이 가장 큰 노드 3개를 선정한다. 만약 같은 가용 대역폭을 갖는 노드가 존재하는 경우에는 물리적인 거리를 계산하고 제일 가까운 노드를 선정한다. 이때 물리적인 거리는 노드의 위도, 경도 값을 Haversine Formula (1)을 이용하여 대권거리 (Great Circle Distance) (2)를 계산한 후 대권거리와 지구의 평균 반지름 (약 6,371km)을 곱하여 (3) 노드 간의 물리적 거리를 계산한다. 이러한 조건으로 선정된 3개의 노드에게 Query를 보낸다.

$$a = \sin^2(\triangle\phi/2)^2 + \cos\phi_1 \cdot \cos\phi_2 \cdot \sin^2(\triangle\lambda/2) \quad (1)$$

$$c = 2 \cdot \tan^{-1}\left(\frac{\sqrt{a}}{\sqrt{1-a}}\right) \quad (2)$$

$$Distance = R \cdot c \quad (3)$$

where, ϕ is latitude, λ is longitude, R is earth's radius (mean radius = 6,371km)

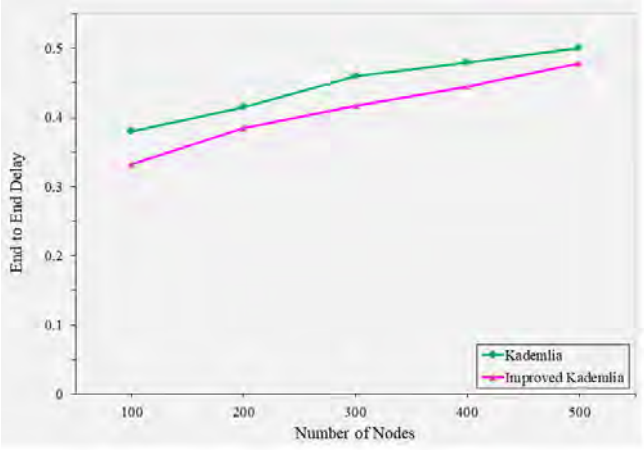
4. 시뮬레이션

Oversim[10]은 OMNeT++[11] 기반의 오픈소스 P2P 네트워크 시뮬레이션 프레임워크로 구조적/비구조적 P2P 프로토콜을 지원하며, 많은 P2P 연구 분야에서 사용되었다. 따라서 본 연구에서는 제안하는 Kademlia의 성능을 평가하기 위해서 Oversim을 사용하였으며, 시뮬레이션에서 사용된 Parameter는 (표 2)와 같다.

(표 2) Simulation Parameters

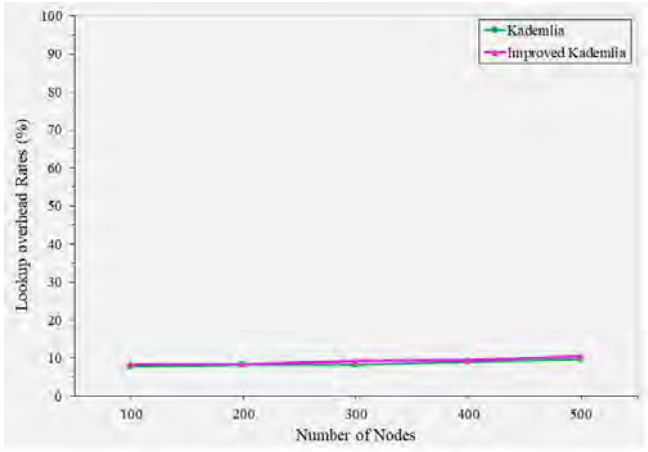
Parameter	Unit	Value
Bandwidth	Mbps	100
Latency	ms	10 ~ 50
Distance	km	0 ~ 20,014
Latitude	Degree	-90.00 ~ 90.00
Longitude	Degree	-180.00 ~ 180.00
Node	Number	100 ~ 500

첫 번째 실험에서 노드 수의 증가에 따른 End-to-End Delay를 측정하였다(그림 3). 이 실험에서 개선된 Kademlia는 기존의 방식과 유사하게 노드 수에 따른 End-to-End Delay 증가가 발생하는 것을 확인할 수 있다. 하지만 기존의 Kademlia보다 개선된 방식의 End-to-End Delay가 적게 발생하는 것으로 나타났다. 이를 통해서 제안하는 가용 대역폭과 실제 환경을 고려할 경우, 기존의 Kademlia가 논리적인 거리만을 고려할 때보다 더 나은 성능을 보임을 확인할 수 있다.



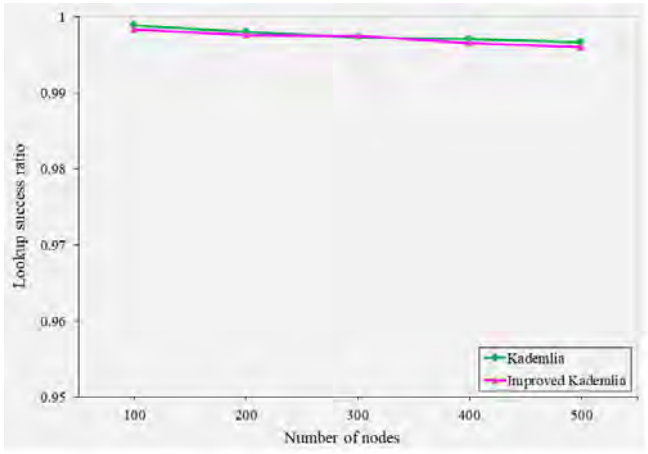
(그림 3) 노드 수 증가에 따른 End-to-End Delay

두 번째 실험에서는 노드 수의 증가에 따른 Lookup Overhead Rates를 측정하였다(그림 4). 이 실험에서는 개선된 Kademlia와 기존의 Kademlia에서 발생하는 전체 Overhead에서 Lookup이 차지하는 비중이 유사하다는 것을 확인할 수 있다. 또한 노드 수 증가에 따라서 Lookup Overhead Rates가 소폭으로 증가하는 것을 볼 수 있다.



(그림 4) 노드 수 증가에 따른 Lookup Overhead Rates

세 번째 실험에서는 노드 수 증가에 따른 Lookup Success Ratio를 측정하였다(그림 5). 이 실험에서는 노드 수 증가에 따라서 Lookup Success Ratio가 조금씩 감소한다. 또한 기존의 방식과 비교하였을 때, 제안하는 개선된 Kademlia는 유사한 수치의 Lookup Success Ratio를 보이는 것을 확인 할 수 있다.



(그림 5) 노드 수 증가에 따른 Lookup Success ratio

5. 결론

기존의 Kademlia는 XOR Metric을 통해 산출된 논리적 거리만을 이용하여 노드 탐색을 진행하기 때문에 실제 환경과 가용 대역폭이 고려되지 않는다. 따라서 실제 목적지와의 End-to-End Delay가 커진다. 하지만 제안하는 개선된 Kademlia는 실제 환경과 가용 대역폭을 고려하기 때문에 동적인 네트워크 환경에서 목적지를 찾고 최적 경로를 통해 데이터를 송·수신하게 된다.

실험결과를 보면 개선된 Kademlia는 기존 방식과 유사한 수준의 Lookup Overhead Rates와 Lookup Success Ratio를 가지지만, End-to-End Delay가 향상된 것을 확인할 수 있다. 이를 통해서 개선된 Kademlia는 실제 환경이 고려된 최적 경로가 선정되어 성능이 향상되었음을 볼 수 있다. 하지만 개선된 Kademlia를 대규모 네트워크 환경에서 적용하였을 때, 오버레이 네트워크상의 논리적 거리 증가가 성능에 어떠한 영향을 미칠지 알 수 없다. 따라서 논리적 거리와 실제 환경이 고려된 Kademlia Binary Tree 구조의 동적 업데이트에 관한 연구가 필요하다.

참고문헌

[1] 김병오, 이일우, 박호진 . "분산 해시 테이블 기반 P2P 기술 동향." [ETRI] 전자통신동향분석,(2018)

[2] Pouwelse, Johan, et al. "The bittorrent p2p file-sharing system: Measurements and analysis." *International Workshop on Peer-to-Peer Systems*. Springer, Berlin, Heidelberg, 2005.

[3] Wood, Gavin. "Ethereum: A secure decentralised generalised transaction ledger." *Ethereum project yellow paper* 151 (2014): 1-32.

[4] Stoica, Ion, et al. "Chord: a scalable peer-to-peer lookup protocol for internet applications." *IEEE/ACM Transactions on Networking (TON)* 11.1 (2003): 17-32.

[5] Rowstron, Antony, and Peter Druschel. "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems." *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer, Berlin, Heidelberg, 2001.

[6] Maymounkov, Petar, and David Mazieres. "Kademlia: A peer-to-peer information system based on the xor metric." *International Workshop on Peer-to-Peer Systems*. Springer, Berlin, Heidelberg, 2002.

[7] Zhao, Ben Y., et al. "Tapestry: A resilient global-scale overlay for service deployment." *IEEE Journal on selected areas in communications* 22.1 (2004): 41-53.

[8] Ratnasamy, Sylvia, et al. *A scalable content-addressable network*. Vol. 31. No. 4. ACM, 2001.

[9] Lua, Eng Keong, et al. "A survey and comparison of peer-to-peer overlay network schemes." *IEEE Communications Surveys and tutorials* 7.1-4 (2005): 72-93.

[10] Baumgart, Ingmar, Bernhard Heep, and Stephan Krause. "OverSim: A flexible overlay network simulation framework." *2007 IEEE global internet symposium*. IEEE, 2007.

[11] Varga, Andras. "OMNeT++." *Modeling and tools for network simulation*. Springer, Berlin, Heidelberg, 2010. 35-59.