효율적인 탐색을 위한 위상 인지 Chord 시스템

김진홍* · 조인준* · 김승해**

*배재대학교 컴퓨터공학과 · **한국과학기술정보연구원

Topology-aware Chord system for efficient lookup

Jin-Hong Kim* · In-June Jo* · Seung-Hae Kim**

*Dept. of Computer Engineering, Paichai Univ. · KISTI**

E-mail : {jhkim@mail.pcu.ac.kr, injune@mail.pcu.ac.kr, shkim@kisti.re.kr}

요 약

현재까지 중앙집중형 시스템인 Napster와 비구조적 시스템인 Gnutella가 P2P 애플리케이션의 주종을 이룬다. 중앙집중형 시스템과 비구조적 시스템은 네트워크의 규모가 커질 경우 시스템이 정상적인 동작을 하지 못하는 문제점을 안고 있다. 이러한 확장성 문제를 해결하기 위해 구조적 P2P 시스템이 활발히 연구되고 있다. 그 대표적인 시스템으로 Chord, CAN, Pastry, Tapestry가 있다. 그러나 구조적 P2P 시스템은 분산 해쉬 테이블을 사용함으로써 물리적인 근접 노드를 인식하지 못하는 문제점을 가진다. 노드를 IP가 아닌 해쉬값으로 식별하기 때문에 해당 노드의 물리적 위치를 알지 못하기 때문이다.

제안하는 시스템은 Chord에서 근거리 네트워크의 개념을 이용하여 근접 노드들 간의 통신을 가능하게 한다. 이로 인해 기존의 Chord 시스템보다 효율적인 탐색을 할 수 있도록 한다. 또한 Chord 네트워크에서의 통신을 서브 네트워크로 분산시킴으로써 인터넷 트래픽을 줄이는 효과를 볼 수 있다.

ABSTRACT

Centralized P2P system, Napster and unstructured P2P system, Gnutella accomplish main current. Centralized system and unstructured system have a restriction in a scalability. To solve the this problem, structured system is appeared. CAN, Chord, Pastry and Tapestry are delegation of this system. Although structured system don't aware physical proximity of node because it uses Distribute Hash Table.

In proposing system, a node can communicate with physical proximity of node using concept of LAN. Internet traffic is also decreased because communication in the Chord network divide in two(original network and sub network).

키워드

chord, topology, lookup, Peer to Peer, sub network

I. 서 론

P2P는 기존의 클라이언트/서버 개념에서 벗어나 P2P 시스템에 참여하는 노드들끼리 직접 연결하여 자원을 공유하고 검색하는 과정에 참여함으로써 모든 참여자가 클라이언트이면서 서버의 역할을 수행한다. 그러므로 P2P 네트워크에서는 네트워크에 참여한 노드를 피어(peer) 혹은 servant라고 한다. 이러한 P2P 네트워크에서 피어는 자율적으로 관리되고 네트워크 내에 존재하는 데이터는 네트워크에 존재하는 모든 피어에서 접속 가능하다. 또한 피어들은 임의로 가입하고 임의로 네트워크를 떠날 수 있다. 때문에 P2P 시스템에서 핵심이 되는 기능은 네트워크에 존재하는 데이터에 대한 정보를 분산하는 방법, 분산된 데이터를 탐색하는 방법이라 할 수 있다.

현재 많이 사용되어지는 P2P 시스템으로 Napster[1], Gnutella[2]가 있다. 하지만 이러한 두 시스템은 확장성에 제약을 가지고 있다.

이를 해결하기 위해 overlay network을 활용하는 구조적인 P2P 시스템들이 제안되고 있다. 그 대표적인 시스템으로 Chord[3], CAN[4], Pastry[5], Tapestry[6]가 있다. 이러한 시스템에서는 피어의 식별자를 해쉬함수를 사용하여 나타낸다. 이로 인해 탐색하고자 하는 데이터가 물리적으로 이웃한 노드에 있음에도 광범위한 Chord 네트워크에서 탐색을 수행하게 된다. 이러한 비효율성을 해결하고자 근접한 노드를 서브 네트워크로 구성하여 서브 네트워크에서 먼저 탐색을 수행하는 시스템을 제안한다.

본 논문은 II장에 관련 연구, III장에 제안 시스템 구조, IV장에 결론을 다루었다.

II. 관련연구

Chord 시스템은 P2P 응용을 위한 분산처리 자원탐색 프로토콜이며 키가 주어지면 키를 이용하여 노드를 매핑하는 하나의 오퍼레이션만을 지원한다.

Chord는 2^m 크기의 원형 식별자 공간을 사용하며 각 노드는 IP주소를 SHA-1과 같은 해쉬함수로 해쉬하여 m-bit 노드-ID를 구한 다음 원형 식별자 공간의 해당 노드-ID에 위치한다. 데이터의 위치 정보는 (key, value) 쌍으로 표현한다. 여기에서 key는 데이터의 이름을 나타내고 value는 데이터의 위치정보를 뜻한다. 위치정보 (key, value)가 저장될 위치는 키를 해쉬한 값에 의해 정해진다. 원형 식별자 공간에서 해쉬된 키 값과 같은 노드-ID를 가지는 노드에 저장하거나 같은 노드-ID를 가진 노드가 없을 때는 바로 뒤의 노드에 저장한다. 이 노드를 후계자 노드라 부른다. 그림 1은 3-bit의 노드-ID로 구성되는 간단한 Chord 시스템이다. 0, 1, 3은 노드-ID이고, 1, 2, 6은 object-ID이다. object-ID 6에 대한 후계자 노드는 0이다. object-ID 1에 대한 후계자 노드는 1이다. object-ID 2에 대한 후계자 노드는 3이다. 각각의 후계자 노드가 해당하는 데이터 위치정보를 관리하게 된다.

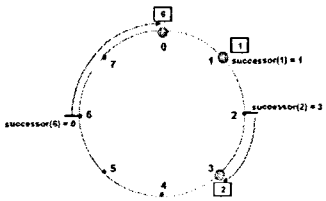


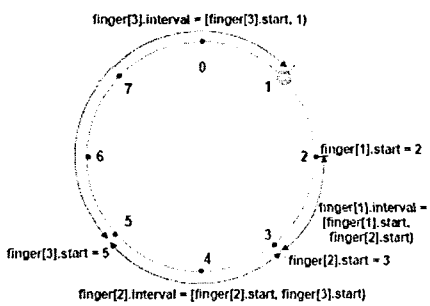
그림 1. successor와 키 분산

각 노드는 전체 네트워크 상의 노드에 대한 정보가 분산된 동적 환경에서 효과적으로 데이터를 삽입, 탐색하기 위해 finger table을 이용한다.

finger table은 모든 노드의 정보를 가지고 있는 것이 아니라 2의 지수 간격으로 모두 m개 (m-bit 해쉬 함수를 사용했을 때)의 entity를 가진다. 그림 2에서 finger table의 구성 요소와 interval에 대해서 나타내고 있다. interval 범위에 해당하는 object-ID가 주어질 경우 이 interval에 대응하는 후계자 노드에게 탐색 메시지를 전달하게 된다. 그림 2의 (b)에서 보듯이 한 노드에서 interval이 지수승으로 커지면서 식별자 공간을 전부 나타내게 된다.

Notation	Definition
finger[k].start	$(n + 2^{k-1}) \bmod 2^m,\ 1 \le k \le m$
finger[k].interval	$[finger[k].start, finger[k+1].start)$, if $1 \le k < m$ $[finger[k].start, n)$, if $k = m$
finger[k].node	first node whose identifier is equal to or follows $n.finger[k].start$
successor	immediate successor of node n on the identifier circle: $successor = finger[1].node$
predecessor	immediate predecessor of node n on the identifier circle

(a)



(b)

그림 2. (a)finger table의 요소 (b)finger table의 interval

Chord 네트워크에 조인하고자 하는 노드는 부트스트랩 노드에 자신의 노드-ID 값을 알려주고, 부트스트랩 노드는 이 ID 값으로 네트워크에 탐색 메시지를 보냄으로써 해당 노드-ID의 후계자 노드가 어느 노드인지 알게 된다. 조인하고자 하는 노드는 후계자 노드에게 자신이 네트워크에 조인함으로써 관리하게 될 object-ID를 넘겨받고 네트워크는 finger table을 갱신하게 된다.

Chord 네트워크에 조인되어 있던 노드가 떠날 때 이 노드는 자신이 관리하던 키 값을 후계자 노드에게 넘기고 이때도 역시 네트워크에서는 finger table의 갱신이 이루어진다.

Chord 시스템은 시스템을 자유롭게 조인하거나 떠나는 노드에서 효율적이며 시스템이 지속적으로 변경되는 과정에서도 탐색을 수행 할 수 있다.

III. 제안 시스템 구조

overlay network에서는 해쉬 함수의 특성상 물리적인 네트워크를 고려하지 않아 물리적으로 이웃한 노드가 원하는 데이터를 가지고 있음에도 불구하고 object-ID를 찾기 위해 광범위한 Chord 네트워크의 여러 노드들을 거쳐 결국에는 물리적으로 이웃한 노드에서 데이터를 받게 된다. 이렇게 원하는 데이터를 물리적으로 근접한 노드에서 가지고 있음에도 불구하고 여러 노드를 거침으로써 비효율적인 탐색 과정을 수행한다.

이해를 돕기 위해 Chord 시스템에 참여하는 모든 노드로 구성되는 네트워크를 원래 네트워크라 하고 이웃한 노드로 구성되는 네트워크를 서브 네트워크라고 하겠다.

제안하는 시스템에서는 111.112.113.114라는 임의의 IP에서 111.112.113까지의 IP주소를 해쉬하여 111.112.113.0 네트워크에 있는 노드들을 하나의 서브 네트워크로 구성하여 원래 네트워크에 탐색하기 전에 먼저 서브 네트워크에서 탐색함으로써 효율적인 탐색을 기대할 수 있도록 한다. 더불어 데이터 교환과 탐색 메시지를 서브 네트워크로 유도함으로써 인터넷 트래픽을 줄이는 이점도 얻을 수 있다.

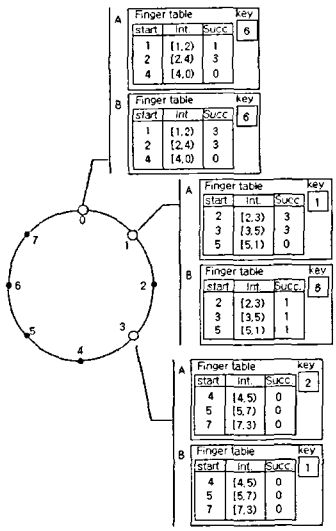
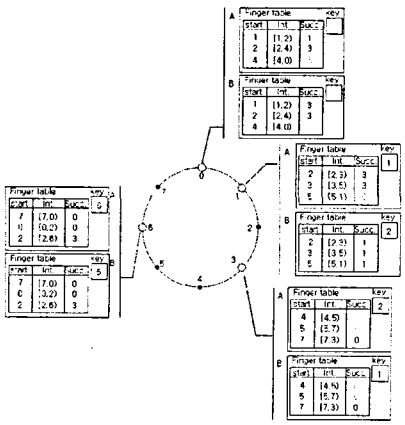


그림 3. 제안시스템의 기본 구조

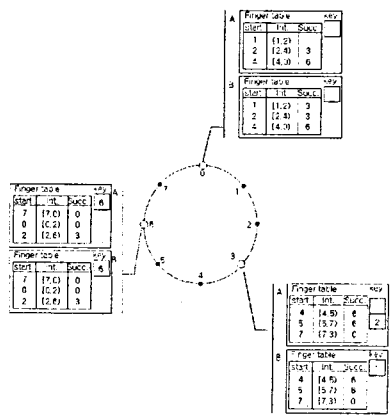
3.1 노드의 조인(join)과 리브(leave)

노드가 조인할 경우 각 노드는 IP의 왼쪽으로부터 3개의 숫자를 해쉬하여 참여하게 될 근거리 네트워크를 찾게 된다. 서브 네트워크를 구분 짓는 이 해쉬값은 처음 서브 네트워크를 생성한 노드가 관리한다. 이 노드가 네트워크를 떠날 경우에는 후계자 노드에게 서브 네트워크를 구분 짓는 해쉬값을 넘긴다.

노드는 원래 네트워크에 조인하면서 동시에 서브 네트워크에 조인한다. 서브 네트워크에 조인하는 방법은 다음과 같다. 우선 자신이 참여하게 될 서브 네트워크 식별자(해쉬값)를 생성하여 시스템에 서브 네트워크가 있는지 확인한 후 서브 네트워크 해쉬값을 가지고 있는 노드가 있을 경우 그 노드를 부트스트랩 노드로 하여 서브 네트워크에 조인하게 되고 해쉬값을 가지는 노드가 없을 경우 자신이 서브 네트워크에 대한 부트스트랩 노드가 된다. 그림 3에서 노드 0과 3은 같은 서브 네트워크에 있다고 가정한다. A 테이블은 원래 네트워크의 finger table이고, B 테이블은 서브 네트워크의 finger table이다. 기존과 다르게 한 노드에서 finger table을 두 개 유지하게 된다. 그림 4의 (a)에서 동일한 서브 네트워크의 노드 6이 조인할 경우를 보인다. (b)는 다른 서브 네트워크를 구성하는 노드 1이 네트워크를 떠났을 때의 테이블이다.



(a)



(b)

그림 4. (a)노드 6의 조인, (b)노드 1의 나감

3.2 데이터의 삽입과 탐색

노드에서 공유하고자 하는 데이터가 있을 경우 데이터에 대한 위치정보는 원래 네트워크와 서브 네트워크에 각각 저장된다.

데이터 탐색의 경우 서브 네트워크에서 먼저 데이터를 탐색하여 원하는 데이터를 찾을 수 없을 경우 원래 네트워크에서 탐색을 수행한다. 원래 네트워크에서 데이터를 찾았다면 이 데이터를 서브 네트워크에 다시 공유한다. 이후부터는 서브 네트워크에서 다른 노드들이 같은 데이터를 찾을 경우 원래 네트워크에서 탐색을 수행하지 않고 빠른 근거리 네트워크를 통해 통신이 가능하다. 또한 서브 네트워크에서 데이터 공유가 활발해질수록 인터넷 트래픽이 저하되는 효과는 커지게 된다.

IV. 결 론

컴퓨터의 성능 발전과 초고속 통신망의 발전으로 인해 인터넷 망에는 많은 트래픽이 생기게 된다. 이로 인해 물리적으로 가까운 근거리 네트워크에서의 통신이 빠른 것이 일반적이다.

제안하는 시스템에서는 물리적으로 이웃한 노드들로 구성된 서브 네트워크에서의 탐색 메시지 전달, 데이터 교환으로 전반적인 Chord 시스템의 성능 향상과 함께 기존의 Chord 시스템보다 인터넷 트래픽의 감소라는 이점을 가지고 있다.

참고문헌

[1] Napster. http://www.napster.com

[2] Gnutella. http://www.gnutella.com

[3] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H.Balakrkshnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," In Proceedings of SIGCOMM (August 2001)

[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalable Content-Addressable Network," In Proceedings of SIGCOMM (August 2001), ACM.

[5] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peerto-peer Systems," In Proceedings of IFIP/ACM Middleware 2001 (November 2001).

[6] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," Tech. Rep. UCB/CSD-01-1141, UC Berkeley, EECS,2001.

[7] 팀 오라이리 외 24인, Peer-to-Peer 차세대 인터넷 P2P, 한빛미디어 출판사, 2001