Journal of Digital Contents Society Vol. 12 No. 1 Mar. 2011(pp. 133-139)

구조화된 Peer-to-Peer 네트워크를 위한 적응적 링크 복구 주기 결정 알고리듬

김석현*, 김태은**

요 약

구조화된 P2P (Structured Peer-to-peer) 네트워크는 지금까지 많은 관심을 받아왔다. 중앙 서버가 없는 분산 환경에서 구조화된 P2P 네트워크를 사용하면 원하는 자료를 로그 스케일의 검색 시간 안에 찾을 수 있다. 구조화된 P2P 네트워크의 이러한 특성을 유지하려면 그것의 링크 구조를 유지해야 한다. 네트워크에 노드가 빈번하게 참여, 이탈하는 상황에서 구조화된 P2P 네트워크를 운용하면 유효하지 않은 (unavailable) 링크가 발생하게 되고, 이러한 링크가 많아지면 검색 성능이 점차 떨어지게 된다. 구조화된 P2P 네트워크의 특성을 유지하기 위해 주로 사용 되어온 방법은 주기적으로 링크를 복구하는 것이다. 그러나 이 방법은 노드의 참여, 이탈 속도가 빈번하게 변하는 환경에서 충분히 링크를 복구하지 못하거나, 필요 이상의 메시지를 사용할 수 있는 단점을 가지고 있다. 본 논문은 노드의 참여, 이탈 속도에 대응하여 링크 복구 주기를 결정함으로써 노드가 참여, 이탈하는 속도가 동적으로 변화하는 환경에서 구조화된 P2P 네트워크의 특성을 유지할 수 있는 알고리듬을 제시한다. 시뮬레이션 결과 제시한 알고리듬은 다양한 노드의 이탈 속도에 대해 적절한 QoS를 유지함을 확인할 수 있었다.

Adaptive Link Recovery Period Determination Algorithm for Structured Peer-to-peer Networks

Seokhyun Kim*, Tae-eun Kim**

Abstract

Structured P2P (peer-to-peer) networks have received much attention in research communities and the industry. The data stored in structured P2P networks can be located in a log-scale time without using central severs. The link-structure of structured P2P networks should be maintained for keeping log-scale search performance of it. When nodes join or leave structured P2P networks frequently, some links become unavailable and search performance is degraded by these links. To sustain search performance of structured P2P networks, periodic link recovery scheme is generally used. However, when the link recovery period is short or long compared with node join and leave rates, it is possible that sufficient number of links are not restored or excessive messages are used after the link-structure is restored. We propose the adaptive link recovery determination algorithm to maintain the link-structure of structured P2P networks when the rates of node joining and leaving are changed dynamically. The simulation results show that the proposed algorithm can maintain similar QoS under various node leaving rates.

Keywords : Structured Peer-to-peer networks, link recovery, adaptive algorithm

1. 서론

구조화된 p2p (Structured peer-to-peer) 네트워크는 효율성 (efficiency), 결함허용성 (fault tolerance), 범위성 (scalability)과 같은 좋은 특성을 가지고 있기 때문에 많은 주목을 받아 왔

※ 제일저자 : 김석현
접수일:2011년 03월 08일, 완료일:2011년 03월 28일,
완료일:2011년 03월 30일
* 남서울대학교 멀티미디어학과
shkim@nsu.ac.kr
**남서울대학교 멀티미디어학과
▣ 본 논문은 남서울대학교 2010년 교내연구공모과제 연구비 지원에 의해서 연구되었습니다.

다 [1,2,3]. 구조화된 p2p 네트워크는 DHT (Distributed hashing Tables) 라는 용어로도 흔히 불리 운다. 이는 P2P 네트워크에 저장된 자료를 찾기 위해, 이 자료의 해쉬 값을 구하고, 이 값으로 분산 검색을 통해 원하는 자료를 찾을 수 있기 때문이다. DHT에 참여하는 노드의 수를 N이라 할 때, 원하는 자료를 찾기 위해 방문해야 하는 노드 수는 일반적으로 O(log N)이다.

DHT의 긍정적인 특성은 DHT의 구조가 정확히 유지될 때 보장될 수 있다. DHT에 참여하는 노드가 빈번하게 네트워크에 참여하거나 이탈하는 상황 하에서는 DHT의 좋은 특성을 유지하기 어렵다 [4,5,6].

DHT를 구성하는 노드들이 빈번하게 참여, 이탈하는 상황에서 각 노드의 라우팅 테이블은 (routing table) 일관성을 (consistency) 유지하기 어렵다. 이는 검색 성능 저하, 잘못된 라우팅, 시스템의 분리 등을 유발할 수 있다 [7,8].

일반적으로 노드가 빈번하게 참여, 이탈하는 상황을 극복하기 위하여 주기적인 링크 복구가 시행된다 [8,9]. 주기적으로 링크를 복구하면서 각 노드의 라우팅 테이블이 현재 네트워크의 정확한 상태를 유지하도록 하는 것이다.

각 노드에서 시행되는 링크 복구 알고리듬의 주기는 보통 시스템 전체에 통용되는 파라미터로써 (parameter) 설정된다. 하지만 만약 노드가 참여, 이탈하는 속도가 동적으로 변한다면 이러한 접근 방식은 효율적으로 노드의 라우팅 테이블을 복구하기 어렵다. 노드들의 참여, 이탈 속도에 비해 링크 복구 주기가 늦다면 DHT는 자신의 구조를 적절한 상태로 유지하기 어렵다. 반대로 노드의 참여, 이탈 속도에 비해 링크 복구 주기가 빠르다면 노드 사이에 필요 이상의 메시지를 주고받음으로써 네트워크 대역폭을 낭비할 가능성이 크다.

이러한 주기적인 링크 복구 알고리듬의 한계를 극복하기 위해 가십 프로토콜을 (gossip protocol) 응용한 적응적인 링크 복구 알고리듬을 제안한다.

가십 프로토콜은 적은 오버헤드로 노드의 참여, 이탈 속도를 추정하기 위해 사용된다. DHT는 중앙 서버 없이 자율적인 노드들의 분산 구조로 작동하는 시스템이기 때문에 전체 시스템의 정보를 수집하는 중앙 서버와 같은 구조를 가정할 수 없다. 가십 프로토콜은 이러한 상황에서 노드들이 메시지를 주고받는 과정을 통해 전체 시스템에 대한 어떤 특성에 대한 통계치를 추정할 수 있는 방법이다 [10].

적응적인 링크 복구 알고리듬은 두 단계로 이루어진다. 먼저 가십 프로토콜을 이용하여 DHT 전체에서 노드가 참여, 이탈하는 속도의 평균치를 추정한다. 다음으로 이 추정치를 바탕으로 각 노드에서 깨어진 링크의 수를 추정하고, 깨어진 링크의 수가 일정 비율에 도달하면 링크 복구 루틴을 수행한다. 제안된 알고리듬의 효율성은 시뮬레이션을 통하여 입증하였다. 실험에 사용한 DHT는 DKS 시스템 [11] 이다.

본 논문은 DKS 시스템에 적응적인 링크 복구 알고리듬을 적용함으로써 다양하게 변화하는 노드의 참여, 이탈 속도에 각 노드가 적응하여 링크 복구 주기를 자율적으로 결정하는 알고리듬을 제시한다. 본 논문에서 제시한 알고리듬의 성과는 다음과 같이 요약할 수 있다.

- DHT에서 분산 방식으로 동적으로 변화하는 노드의 참여, 이탈에 대응하는 방안 제시
- DHT의 한 종류인 DKS 시스템에 제안한 알고리듬을 적용
- 기존 DHT에 일반적인 주기적 복구 알고리듬과의 비교 분석

실험 결과 본 논문이 제안한 적응적인 링크 복구 알고리듬은 동적으로 변화하는 노드의 참여, 이탈 속도에 대응하여 링크 복구 주기를 변화시킴으로써 전체 DHT의 QoS를 일정하게 유지할 수 있음을 확인하였다.

본 논문의 나머지 부분은 다음과 같이 구성되어있다. 2장에서 관련 연구를 요약한다. 3장에서 가십 프로토콜을 DHT에 적용한 방법을, 4장에서 추정된 노드의 이탈 속도에서 링크 복구 주기를 결정하는 알고리듬을 설명한다. 5장에서 시뮬레이션 결과를 제시하고, 6장에서 결론을 맺는다.

2. 관련 연구

2.1 DHT의 링크 복구 알고리즘

Ou는 Kademlia DHT 위에서 Peer-to-Peer

프로토콜 (P2PP)을 시그널링 프로토콜로 사용하여 노드의 참여, 이탈이 빈번한 상황에서 성능 테스트를 시행 하였다 [8]. 이 논문에서 제안한 링크관리 알고리즘은 주기적으로 랜덤하게 선택된 노드들의 라우팅 테이블 정보를 교환하면서 서로 새로운 링크 정보를 얻게 하는 것이다.

Rhea는 Chord나 Pastry와 같은 DHT들이 노드의 참여, 이탈이 빈번한 상황에서 여러 가지 문제를 야기함을 보이고 이를 극복하기 위해 Bamboo라는 새로운 DHT를 제안하였다 [9]. Bamboo는 reactive 방식이나 proactive 방식으로 링크 정보를 수정하며, 이러한 링크 복구 메커니즘을 주기적으로 실행한다.

Ghodsi는 DKS 시스템 이라는 DHT를 제안하였다 [11]. 이 DHT는 노드가 참여, 이탈할 때 분산 방식으로 그러한 변화를 알아내고 관련 노드들의 라우팅 테이블에 이를 바로 반영하는 방식을 사용한다. 이 기법은 라우팅 테이블에 수정이 필요한 경우에만 작동하므로 효율적이지만, 다른 DHT에 적용할 수 없는 단점을 가지고 있다.

2.2 가십 프로토콜

가십 프로토콜은 여러 분야에서 다양한 목적으로 사용되며 다양한 알고리듬이 존재하는 방대한 분야이다. 여기에서는 P2P 네트워크와 관련된 주요 연구 두 가지만 간략히 언급한다.

Kempe는 공간적인 가십을 (spatial gossip) 제안하였다 [12]. d 차원 공간상에 점들이 배치되어 있다고 하고, 점 사이의 거리의 거듭제곱의 역수에 비례하는 확률로 정보를 전달한다. 이 경우 전체 점들의 수에 대한 로그 스케일로 정보가 빠르게 전달된다.

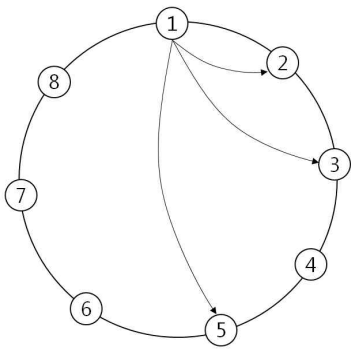
Jelasity는 가십 프로토콜을 위한 프레임워크를 제안하였다 [13]. 구조화되지 않은 P2P (unstructured P2P) 네트워크에 가십 프로토콜을 적용하는 것은 쉽지 않은 일이다. 이 연구는 이러한 상황에서도 가십 프로토콜을 적용할 수 있도록 하여 P2P 네트워크에서 가십 프로토콜의 활용도를 높였다.

3. 가십 프로토콜을 통한 노드의 이탈 속도 추정

3.1 가십 프로토콜을 적용한 DHT

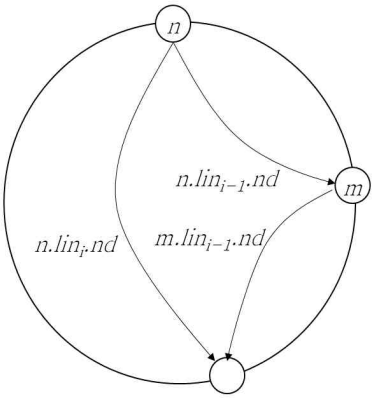
본 논문에서 제안하는 알고리즘을 적용한 DHT는 DKS 시스템 이다 [11]. DKS 시스템의 각 노드는 노드 사이의 홉 수를 기준으로 링크 테이블을 관리한다. 이로 인해 각 노드의 링크 수가 동일하고 여러 통계적인 분석을 수행하기가 용이하다. 하지만 노드의 참여, 이탈이 빈번한 환경에서 정확한 링크 테이블을 유지하려면 링크 관리 비용이 증가한다. 본 논문에서는 통계적 접근의 용이함을 위하여 DKS 시스템을 사용하였다.

DKS 시스템은 몇 가지 parameter를 가지고 있다. 대표적인 파라미터가 검색 arity, k이다. DKS 시스템의 검색 성능은 전체 노드 수가 N이라 할 때 $O(\log_k N)$이 된다. 이후로 k=2인 DKS 시스템을 DKS_2라 하자. DKS_2의 각 노드는 그림 1과 같은 링크 구조를 가지고 있다.



(그림 1) DKS_2 노드의 링크

각 노드는 자신으로부터 2^i 홉 떨어진 노드에 대한 링크를 갖는다. 이 때 i의 범위는 $0 \le i \le \lfloor \log N \rfloor$ 이다.



(그림 2) DKS_2의 링크 업데이트

DKS_2의 링크 업데이트 프로토콜은 그림 2의 관계를 이용하여 쉽게 설계할 수 있다. 먼저 각 노드는 자신의 인접한 이웃 노드에 대한 링크를 알고 있다고 가정한다. 만약, 어떤 노드가 이웃 노드에 대한 링크를 모른다면 DKS_2는 그 지점에서 단절된다. DKS_2가 단절되지 않도록 관리하는 프로토콜은 본 논문이 다루는 주제를 벗어나므로 자세히 언급하지 않는다.

DKS_2의 각 노드는 자신과 2^1=2 홉 떨어진 노드에 대한 링크를 인접 노드, $n.lnk_1.nd$를 통해 다음과 같이 얻을 수 있다.

$$n.lin_2.nd = (n.lnk_1.nd).lnk_1.nd$$

마찬가지로 2^2=4, 2^3=8, ..., 2^i 홉 떨어진 노드에 대한 링크를 얻을 수 있으며 이 관계를 일반화하여 다음과 같이 나타낼 수 있다.

$$n.lin_i.nd = m.lin_{i-1}.nd \text{ where } m = n.lin_{i-1}.nd$$

DKS_2의 모든 노드가 이러한 방식으로 링크를 업데이트 하면 한 노드당 O(logN) 개의 메시지가 필요하다. 따라서 전체 노드의 링크 업데이트에 필요한 메시지 수는 O(NlogN)이다. 모든 링크를 업데이트하는데 필요한 시간은 한 노드가 가지는 링크 수에 비례하므로 O(logN)이 된다.

3.2 DKS_2에 적용된 가십 프로토콜

각 노드 n은 자신의 이웃 노드의 위치에 노드들이 들어오고 나가는 이벤트를 관찰하여 노드들이 떠난 시점을 모두 로그(log)로 남겨 놓는다. 이 로그에서 n은 자신이 측정한 노드의 이탈 속도 $n.\lambda_l$을 유지한다.

모든 노드가 DKS_2를 이탈할 때 이 사실을 이웃 노드에게 알려준다고 가정한다. 이 때 노드 n이 현재까지 측정한 $n.\lambda_l$ 값을 이웃에게 알려준다. 새로운 노드가 에 들어올 때는 이웃노드에서 유지하는 λ_l 값을 그대로 사용한다. 표 1은 이렇게 노드가 참여하거나 이탈할 때 λ_l을 관리하는 알고리듬을 보여준다.

<표 1> 노드 참여, 이탈 시 λ_l 관리 알고리듬

노드 이탈 시 λ_l 관리	노드 참여 시 λ_l 관리
n.notifyLeave()	***n.notifyJoin()***
v = *n.predecessor*.λ_l	*n.predecessor*.λ_l = $n.\lambda_l$
$n.\lambda_l$ = ($n.\lambda_l$ + v) / 2	

가십 프로토콜의 핵심은 측정하고자 하는 특성 값을 각 노드에서 측정하고, 이렇게 측정된 지역 값을 전체 노드들이 분산 방식으로 지속적으로 '혼합'함으로써 실제 평균에 근접해 가는 것이다.

본 논문은 이러한 가십 프로토콜의 특성을 이용하여 각 노드가 push-pull 가십 프로토콜을 이용하여 노드들이 개별적으로 측정한 λ_l을 혼합하여 전체 노드의 λ_l값이 평균치에 근접하도록 한다. Push-pull 가십 프로토콜은 두 노드 n_1, n_2가 통신한다고 할 때, n_1에서 가십의 대상이 되는 값을 보내주면 n_2역시 해당되는 값을 n_1에게 전송하는 방식이다. 표 2는 이러한 push-pull 가십 프로토콜을 이용하여 각 노드에서 측정한 λ_l을 섞어주는 과정을 보여준다.

<표 2> Push-pull 가십 프로토콜

λ_l Push 함수	λ_l Pull 함수, Push-메시지 핸들러
n.pushGossip()	***n.pullGossip(m: node)***
randomly get	*send* $n.\lambda_l$ *to m*
$m \in \{n.links\}$	***n.processGossip(from:node)***
send $n.\lambda_l$ *to m*	**if** *n = from* **then return**
	n.pullGossip(from)
	$n.\lambda_l$=($n.\lambda_l$ + *from*.λ_l) / 2

노드 n, m이 표 2의 방식으로 $n.\lambda_l$, $m.\lambda_l$을 주고받는다고 하자. 표 2의 알고리듬의 결과 $n.\lambda_l$, $m.\lambda_l$은 다음과 같은 값이 된다.

$$n.\lambda_l = n.\lambda_l = (n.\lambda_l + n.\lambda_l) / 2$$

표 2의 push-poll 가십 알고리듬은 노드의 λ_l 값이 바뀔 때, 그 노드에 의해 실행된다. 노드가

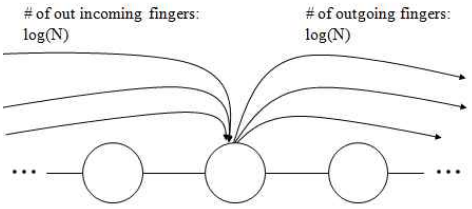
빈번하게 이탈하는 상황을 가정하고 있으므로 이러한 방식으로 전체 노드의 λ_l 값을 충분히 혼합할 수 있다.

4. 적응적인 링크 복구 주기 결정 알고리듬

노드가 이탈하는 속도를 3장에서 제시한 가십 알고리듬에 의하여 추정하였다. 이렇게 계산된 추정치를 통하여 각 노드에서 깨어진 링크의 기대 값을 구할 수 있다.

그림 3은 DKS_2의 한 노드에서 들어오는 링크와 나가는 링크의 수를 보여준다. 이는 3.1절에서 설명한 DKS_2의 링크 구조를 통해 쉽게 도출할 수 있다.

이제 3장에서 설명한 방식으로 추정한 λ_l에서 적응적인 링크 업데이트 주기를 결정하는 방식을 기술한다.



(그림 3) DKS_2 노드의 링크 수

Δt의 시간동안 전체 네트워크를 이탈한 노드 수는 $N\lambda_l\Delta t$이다. 이는 λ_l이 개별 노드에서 측정한 노드 이탈 속도이기 때문에 전체 네트워크를 대상으로 할 때는 전체 노드의 수를 곱해야 하기 때문이다.

어떤 노드가 Δt 동안 유효하지 않은 상태(unavailable state)가 될 확률은 Δt 시간 동안 네트워크를 이탈한 전체 노드 수를 N으로 나눈 것이다.

$$\Pr\{\ a\ node\ is\ unavailable\ during\ \Delta t\ \}$$
$$= total\ leaving\ nodes\ during\ \Delta t\ /\ N$$
$$= N\lambda_l\Delta t\ /\ N$$
$$= \lambda_l\Delta t$$

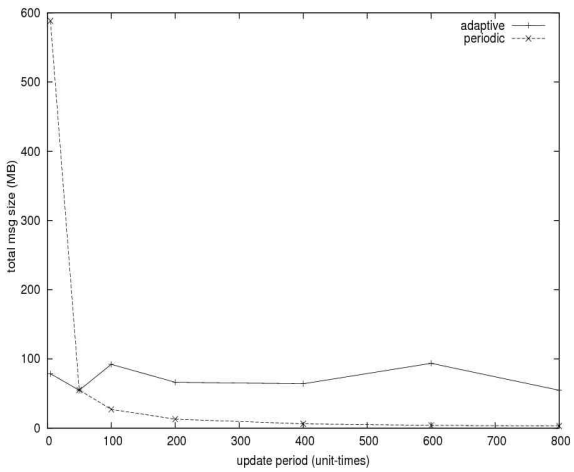
그림 3과 같이 DKS_2의 각 노드는 logN개의 다른 노드를 향한 링크를 가지고 있다. 한 노드가 네트워크를 이탈할 확률은 $\lambda_l\Delta t$ 이므로 logN개의 링크 중, 노드 이탈로 인하여 유효하지 않은 상태가 되는 링크의 수는 $(\lambda_l\Delta t\log N)$이다. 따라서 하나의 링크가 깨어지기 까지 걸리는 시간은 이 값의 역수인 $(\lambda_l\Delta t\log N)^{-1}$이 된다. 결국 각 노드는 $(\lambda_l\Delta t\log N)^{-1}$를 주기로 하여 링크 복구 알고리즘을 구동 시키고, 이 링크 복구 알고리즘에서 유효하지 않은 링크 하나를 복구 하면 전체 노드의 라우팅 테이블을 유효한 상태에 가깝게 유지할 수 있다.

5. 시뮬레이션 결과

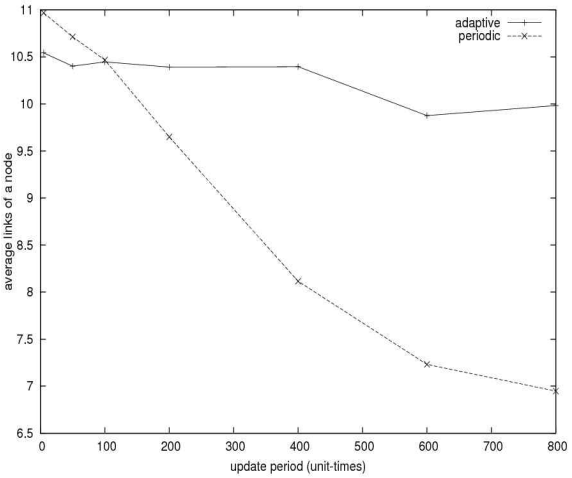
제안한 알고리즘을 평가하기 위하여 c++을 이용해 시뮬레이터를 작성하였다. 모든 시뮬레이션은 1,000 개의 노드를 기반으로 행해졌다. 노드가 네트워크를 이탈하는 속도, λ_l은 0.1로 놓았다. 실험은 다양한 링크 복구 주기에 대하여 이루어졌다.

실험 그룹 1은(adaptive group) 본 논문에서 제안한 알고리즘을 실행하는 그룹이다. 각 노드는 미리 설정한 λ_l을 모르는 체로 작동한다. 처음에는 주어진 복구 주기로 알고리듬이 작동하지만, 노드들이 각자 노드의 네트워크 이탈 속도를 추정한 다음, 적응적으로 노드 복구 주기를 변화 시킨다.

실험 그룹 2는 (periodic group) 실험 대조군으로써, 주기적으로 같은 수의 링크를 복구하는 방식으로 작동하였다.



(그림 4) 알고리듬 실행 주기와 복구비용



(그림 5) 알고리듬 실행 주기와 복구된 링크 수

그림 4는 알고리듬의 실행 주기와 복구비용의 관계를 보여준다. 실험 그룹 1은 알고리듬의 실행 주기와 관계없이 일정한 양의 메시지를 링크 복구를 위하여 사용하고 있다. 반면에 실험 그룹 2는 링크 주기의 길이에 반비례하는 복구비용을 보여주었다. 특히 링크 복구 주기가 0에 가까워질수록 복구비용이 급격히 증가하는데, 이는 실험 그룹 2의 복구비용이 링크 주기에 반비례하는 특성을 잘 보여준다.

그림 5는 각 실험 그룹에 의해 복구된 링크의 수를 보여준다. 본 논문에서 제안한 실험 그룹 1은 알고리듬의 실행 주기와 관계없이 유사한 수의 링크를 복구한다. 만면에 실험 그룹 2는 실행 주기가 증가할수록 더 적은 수의 링크를 복구한다.

DHT가 자신의 특성을 유지하기 위해서는 일정한 수의 링크를 유지해야 한다. 본 논문에서 제안한 기법은 변화하는 링크 복구 주기와 관계없이 필요한 만큼의 링크를 복구함으로써 일정한 QoS를 보장함을 확인할 수 있었다.

6. 결론

본 논문은 구조화된 P2P (structured peer-to-peer) 네트워크에서 활용할 수 있는 적응적인 링크 복구 주기 결정 알고리듬을 설명하였다. 구조화된 P2P 네트워크는 빈번한 노드의 참여, 이탈로 인하여 링크 구조가 깨어지면서 구조화된 P2P 네트워크의 여러 가지 좋은 특성들을 잃어버리게 된다. 본 논문에서 제안한 기법은 다양한 상황 아래에서 적응적으로 필요한 수만큼의 링크를 복구할 수 있게 함으로써 구조화된 P2P 네트워크의 구조를 적은 비용으로 유지할 수 있는 알고리듬을 제시 하였다. 실험 결과 본 논문에서 제시한 알고리듬은 다양한 상황에서 유사한 QoS를 보여줌을 볼 수 있었다.

참 고 문 헌

[1] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. Chord: A scalable peer-to-peer lookup service for internet applications. SIGCOMM Comput. Commun. Rev. 31, 4 (Oct. 2001), 149-160.

[2] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S. 2001. A scalable content-addressable network. In Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols For Computer Communications (San Diego, California, United States). SIGCOMM '01. ACM, New York, NY, 161-172.

[3] Rowstron, A., and Druschel, P. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (MIDDLEWARE '01) (Heidelberg, Germany). Springer, LNCS 2218., 2001, 329-350.

[4] Locher, T., Schmid, S. and Wattenhofer, R. eQuus: A Provably Robust and Locality-Aware Peer-to-Peer System. In Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing (P2P '06), vol., no., Sept. 2006, 3-11.

[5] Anceaume, E., Ludinard, R., Ravoaja, A. and Brasileiro, F. PeerCube: A Hypercube-Based P2P Overlay Robust against Collusion and Churn. In Proceedings of the Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems, 2008 (SASO '08) , vol., no., Oct. 2008, 15-24.

[6] Anceaume, E., Ludinard, R. and Sericola, B. Analytic Study of the Impact of Churn in Cluster-Based Structured P2P Overlays. In Proceedings of the IEEE International Conference on Communications (ICC' 10), vol., no., May 2010, 1-5.

[7] Li, X., Misra, J., and Plaxton, CG., Active and Concurrent Topology Maintenance. In Proceedings of the 18th annual Conrference on Distributed Computing

(DISC '04), (Trippenhuis, Amsterdam, Netherlands). Springer, LNCS 3274., 2004, 320-334.
[8] Rhea, S., Geels, D., Roscoe, T., and Kubiatowicz, J. Handling churn in a DHT. In Proceedings of the Annual Conference on USENIX Annual Technical Conference (Boston, MA, June 27 - July 02, 2004). USENIX Annual Technical Conference. USENIX Association, Berkeley, CA, 2004, 10-10.
[9] Ou, Z., Harjula, E., Kassinen, O., and Ylianttila, M. Performance Evaluation of a Kademlia-based Communication-oriented P2P System Under Churn, Computer Networks, Elsevier, Vol 54., 2010, 689-705.
[10] Shah, D. Gossip Algorithms. Found. Trends Netw. 3, 1 (Jan. 2009)
[11] Ghodsi, A., Alima, L. O., and Haridi, S. Low-Bandwidth Topology Maintenance for Robustness in Structured Overlay Networks, In Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05), (Big Island, Hawaii). IEEE Computer Society, Los Alamitos, CA, USA, 2005, 302a.
[12] Kempe, D., Dobra, A. and Gehrke, J. Gossip-Based Computation of Aggregate Information, In Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS'03). IEEE Computer Society, Los Alamitos, CA, USA, 2003, 482.
[13] Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A., and van Steen, M. Gossip-based peer sampling. ACM Trans. Comput. Syst. 25, 3 (Aug. 2007), 2007, 8.

김 석 현

2001년 : 서울대학교 재료공학부 (학사)
2008년 : 서울대학교 컴퓨터공학부 (석사)
2008년～현 재 : 서울대학교 컴퓨터공학부 (박사과정)
2001년～2004년: (주) 레드덕
2004년～2005년: (주) 재미 인터랙티브
2008년～현 재: 남서울대학교 멀티미디어학과 교수
관심분야 : 피어투피어 컴퓨팅(Peer-to-peer Computing), 클라우드 컴퓨팅, 시스템 보안, 게임 엔진 구조, 임베디드 시스템 등

김 태 은

1992년 : 중앙대학교 대학원 전자공학과 졸업(공학석사)
1997년 : 중앙대학교 대학원 전자공학과 졸업(공학박사)
1995년 삼성전자 휴먼테크논문 은상수상
1997년～현재 : 남서울대학교 교수
관심분야 : 영상통신네트워크공학, 증강현실, 멀티미디어처리