The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 16, No. 6, pp.65-73, Dec. 31, 2016. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2016.16.6.65
JIIBC 2016-6-8

Super-Peer 네트워크에 기반을 둔 Peer-to-Peer 시스템의 계층적 구성

A Hierarchical Construction of Peer-to-Peer Systems Based on Super-Peer Networks

정원호*

Won-Ho Chung*

요 약 수퍼피어 네트워크에 기반을 둔 P2P 시스템은 기존의 하이브리드 P2P 시스템과 순수 P2P 시스템이 결합된 장점을 나타내고 있다. 수퍼피어는 어떤 일반 피어들의 집단에 대해 서버처럼 동작하는 특수한 피어이다. 수퍼피어들의 네트워크를 구성하는 문제는 수퍼피어 네트워크에 기반을 둔 P2P 시스템에 있어서 중요한 문제 중의 하나이다. 기존의 P2P 시스템들은 2 계층으로 구성된 피어들에 기반을 두고 있다. 하나는 일반피어들로 구성된 계층이고 다른 하나는 수퍼피어들로 구성된 계층이다. 수퍼피어 네트워크는 랜덤 그래프의 형태를 가지고 있는 것이 일반적이다. 그러나 대규모 일반 피어들을 수용하기 위해서는 수퍼피어 네트워크 또한 그에 맞도록 확장되어야 한다. 본 논문에서는 이러한 대규모 P2P 시스템을 위한 트리 기반의 수퍼피어 네트워크의 계층적 구성 방법이 제안된다. 먼저 두 개의 계층으로 구성되는 단순 수퍼피어 네트워크의 구성이 소개되고, 그것을 일반화 그리고 확장 시키면서 다중 레벨 수퍼피어 네트워크로 확장하는 알고리즘이 제안된다. ***단순 수퍼피어 네트워크***도 좋은 특징을 가지고는 있으나, 제한된 레벨의 수 때문에 규모성에 문제를 나타낼 수 있어, 좋은 규모성과 클라이언트 노드들에 관한 관리의 용이성을 보여주는 ***확장 수퍼피어 네트워크***라고 하는 k-레벨의 수퍼피어 트리로 확장 시킨다.

Abstract Peer-to-Peer (P2P) systems with super-peer overlay networks show combined advantages of both hybrid and pure P2P systems. Super-peer is a special peer acting as a server to a cluster of generic peers. Organizing a super-peer network is one of important issues for P2P systems with super-peer networks. Conventional P2P systems are based on two-level hierarchies of peers. One is a layer for generic peers and the other is for super-peers. And it is usual that super-peer networks have forms of random graphs. However, for accommodating a large-scale collection of generic peers, the super-peer network has also to be extended. In this paper, we propose a scheme of hierarchically constructing super-peer networks for large-scale P2P systems. At first, a two-level tree, called a *simple super-peer network*, is proposed, and then a scheme of generalizing and then extending the simple super-peer network to multi-level super-peer network is presented to construct a large-scale super-peer network. We call it an extended super-peer network. The simple super-peer network has several good features, but due to the fixed number of levels, it may have a scalability problem. Thus, it is extended to k-level tree of a super-peer network, called *extended super-peer network*. It shows good scalability and easy management of generic peers for large scale P2P system.

Key Words : Hierarchical Construction, Peer-to-Peer System, Super-Peer Network, Simple/Extended Super-Peer Network

*정회원, 덕성여자대학교 디지털미디어학과
접수일자: 2016년 8월 3일, 수정완료: 2016년 10월 5일
게재확정일자: 2016년 12월 9일

Received: 3 August, 2016 / Revised: 5 October, 2016
Accepted: 9 December, 2016
*Corresponding Author: whchung@duksung.ac.kr
Dept. of Digital Media Science, Duksung Women's University, Korea

Ⅰ. 서 론

음악 파일 공유를 위해 최초로 구현된 Napster[1]는 대표적인 하이브리드 Peer-to-Peer(P2P) 시스템이다. 하나의 중앙 서버에 다수 개의 클라이언트들이 연결된 2 레벨 트리 구조이며, 그 서버에 공유 메타 정보를 비롯하여 피어들의 등록 및 자료 검색에 대한 경로 정보 등이 저장되어 있어, 모든 자료 검색은 여기에서 수행되나, 실제 콘텐츠의 전달은 클라이언트 사이에 직접 일어난다. 그러므로 효율성 측면에서는 양호한 특성을 보이나, 다수의 클라이언트들에 의한 동시 검색 요청 시, 중앙 서버에 가해지는 부하의 증가와 서버 오류에 대한 서비스 중지 문제가 단점이라고 할 수 있다. 이와는 반대로, 순수 P2P 방식은, 중앙 서버의 개념이 존재하지 않는 구조로, 검색 요청 및 자료의 전송이 피어들에 의해 구성되는 트리 형상을 따라 전파된다. 그러므로 전달 중 발생하는 중복 피어 참여와 그에 따른 사이클의 형성과 그로 인한 스팸 메시지가 남발될 수 있으며, 또한 피어 간 접속을 해당 트랜잭션이 완료될 때까지 유지하고 있어야하므로, 네트워크 대역폭의 낭비가 심하다. 그러나 어떤 피어의 결함이 끼치는 영향 범위는 최소화 될 수 있다는 장점을 가지고 있다. 초기의 Gnutella[2]와 Freenet[7]이 이에 해당한다. 기존의 P2P 시스템에 관해 Schollmeier가 간략하고 명료하게 정의 분류하였으며[5], 그들의 유형에 따른 분류 및 장단점 등이 많은 연구를 통해 이루어졌다[6].

그러나 최근에 개발되고 있는 대부분의 P2P 시스템은 수퍼피어 네트워크(Super-Peer Network, SPN)를 기반으로 하고 있는데, 이러한 SPN에 기반을 둔 P2P 시스템이 기존의 하이브리드 P2P 시스템과 순수 P2P 시스템의 장점들을 보여주고 있기 때문이다. 서버 역할을 하는 특별한 피어인 수퍼피어(SP)에 일반피어(Generic Peer, GP)들의 그룹이 연결되어 클러스터를 이루고, 이러한 SP들이 상호 연결된 SPN을 기반으로 하는 P2P 시스템이다. 최근 대부분의 P2P 응용은 KaZaA[4]와 Gnutella-2[3]로 대표되는 SPN 기반의 P2P 시스템이다. SPN 기반의 P2P 시스템은 2개의 오버레이 네트워크로 구성된다. 하나는 GP들로 구성되는 계층이며, 다른 하나는 SP들로 구성되는 계층이다. 이러한 SPN 패러다임은 이제 P2P 시스템을 구축하는 일반적인 접근 방식으로 간주되고 있으나, 그들의 구성에 대한 세부 사항들에 관한 표준은 없으며, 응용에 따라 많이 달라진다. 이러한 SPN은 일반적으로 랜덤그래프로 구성하는 것이 보통이지만[8], 응용, 거리, 탐색 콘텐츠의 유사성 등 네트워크 구성 기준에 따라 다양한 형태의 토폴로지를 가지기도 한다[9, 11-15]. 그리하여 효율적인 SPN의 설계를 위한 일반적 과정, 성능의 trade-off 그리고 현실적 가이드라인 등에 관한 필요성이 제기되었다[4, 14]. 그리고 가십 프로토콜과 같은 단순 정보 교환 방식을 사용한 SP의 선정 및 SPN의 구성에 관한 많은 연구가 이루어졌다[16, 17]. 그러나 피어 간의 거리는 콘텐츠의 유사성 등과 같은 척도들과는 달리 시스템의 성능 향상을 위한 불변적 특성이라고 할 수 있다. 피어간의 거리를 측정하는 효율적 방법이 많이 제안되었으나, 수학적 이론을 기반으로 하고 있으며 또한 시뮬레이션 결과를 보여주고 있어, 실제로 적용하는 데는 많은 무리가 있는 것도 사실이다[18]. 그리하여 세계 표준의 국제 전화번호를 기반으로 하는 근접 기반의 SP 선정과 SPN을 구성하는 효율적인 알고리즘이 제안되었다[10]. 그러나 대규모 피어들을 수용하기 위해서는 SPN 또한 그에 맞도록 확장되어야 한다.

본 논문에서는 이러한 대규모 P2P 시스템을 위한 SPN의 계층적 구성 방법이 제안된다. 먼저 단순 SPN라고 하는 2 계층의 수퍼피어 트리 구성이 소개되고, 다음에 그것을 일반화 하면서 단순 SPN을 확장하는 방법을 보여준다. 단순 SPN 자체도 좋은 특징을 가지고는 있으나, 제한된 레벨의 수 때문에 규모성에 문제를 나타낼 수 있어, 좋은 규모성과 클라이언트 노드들에 관한 관리의 용이성을 보여주는 확장 수퍼피어 네트워크라고 하는 다중 레벨의 네트워크로 일반화 시킨다. 이를 확장 SPN이라고 하며, 이를 구성하는 알고리즘이 제안된다. 이는 단순 SPN 구성 알고리즘을 임의의 레벨까지 일반화시킨 것으로, 대규모 P2P 네트워크로의 확장성 및 유연성을 가지고 있으며, 또한 컴퓨팅 자원을 효율적으로 공유하고 관리, 전송하기 위한 유용한 기술이라 할 수 있다.

Ⅱ. 단순 수퍼피어 네트워크와 수퍼피어 선정

1. 단순 수퍼피어 네트워크 모델

단순 수퍼피어 네트워크(Simple Super-Peer Network, 이하 SSPN)는 그림-1에서 보여준 바와 같이, 다른 SP들에 대해 서버 역할을 하는 하나의 특정 피어를 부모노드

로 하여 다수의 SP들이 자식노드들이 되어 2-레벨 트리 형태로 연결된 네트워크이다. 여기서 서버 역할을 하는 특정 피어를 루트피어(Root Peer, RP)라고 한다.

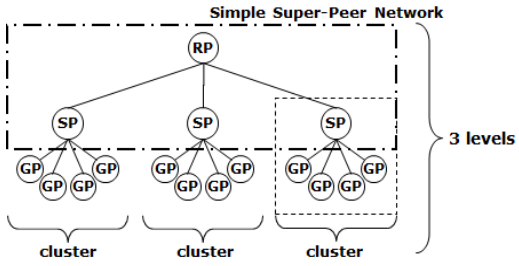

그림 1. 단순 수퍼피어 네트워크)
Fig. 1. Simple Super-Peer Network(SSPN)

RP는 자신에게 연결된 자식 SP들의 등록 및 해지가 핵심 기능이며, 응용에 따라 필요한 경우, 그 외 다양한 메타 정보를 유지 관리할 수 있다. 그리고 SP의 핵심 기능은 자신의 클러스터에 포함된 일반 피어(generic peer, 이하 GP)들의 등록 및 해지이며, 각 SP의 능력에 따라 자신의 클러스터에 포함할 수 있는 GP들의 최대 수를 설정할 수 있는데, 이를 그 SP의 ***capacity***라고 한다. 보통, capacity는 각 SP의 하드웨어 자원, 예를 들면 CPU 성능, 메모리 용량, 네트워크 대역폭, 등에 의해 결정될 수 있다.

하나의 SP와 그에게 등록된 GP들의 집합을 클러스터라고 하면, SSPN 기반의 P2P 시스템은 RP에 연관된 클러스터들의 집합으로 정의할 수 있다. P_{SSPN}를 SSPN 기반의 P2P 시스템이라 하고, $Cluster_i$를 i 번째 SP, 즉 SP_i에 의해 관리되는 클러스터라고 하면,

$$P_{SSPN} = RP \cup \{Cluster_i | i = 1, 2, ..., n\},$$
$$Cluster_i = SP_i \cup \{GP_k | k = 1, 2, ..., m\} \text{ 이다.}$$

여기서 n은 시스템을 구성하고 있는 클러스터의 총 개수 이며, SP들의 참가와 탈퇴가 자유롭게 일어나므로 그 수는 시간에 따라 가변적이다. $Cluster_i$는 SP_i를 포함하는 GP들의 집합이다. C_i를 (SP를 포함하는) SP_i의 *capacitiy*라고 하면, $1 \leq |Cluster_i| \leq C_i$ 이다. 그것은 $Cluster_i$에는 최소한 하나의 SP를 포함하고 있으며, C_i는 (SP를 포함하여) SP_i가 관리할 수 있는 최대 허용 GP의 수이기 때문이다. 온라인 상태에 있는 각 GP는 어느 하나의 SP에는 반드시 등록 되어야 한다.

SPN기반의 P2P 시스템의 구성을 위해서는 SP의 선정 문제가 필수적으로 대두되는데, 이러한 SP의 선정 문제는 세 갈래 방향으로 해석되어 진다. 1) SP로서의 자격 부여를 위한 SP의 선발(SP election) 문제, 2) 등록할 SP의 선택(SP selection) 문제: 참여를 원하는 각 GP가 자신을 등록할 SP를 선택하는 기준을 의미하며, 마지막으로 3) SPN 구성을 위한 SP 선택(SPN organization) 문제 : 새로 선발된 SP를 기존의 SPN에 연결하기 위해 자신을 연결시킬 SP들을 선정하는 문제를 의미한다.

2. Demand-Driven 수퍼피어 선발

첫 번째 문제인 SP로서의 자격 부여를 위한 SP 선발은 어떤 피어가 SP로서의 자격을 가질 수 있느냐하는 기준을 설정하는 문제를 의미한다. 다시 말해서 새로운 SP의 생성 문제라고 할 수 있는데, 피어의 성능을 포함하는 다양한 사항 등을 고려하여 그 자격을 부여하는 것이 일반적이며, 보통 하드웨어 자원, 예를 들면 CPU 성능, 메모리 용량, 네트워크 대역폭 등과 참여 유지시간(uptime) 등을 기준으로 고려한다. 이 문제는 자격 문제도 그렇지만 언제 SP의 생성이 이루어져야 하는가 하는 시점 문제와 동반되는 문제이다. 새로운 SP가 필요하다는 결정이 되어야 SP의 생성이 이루어질 것이기 때문이다. 일반적으로 이러한 시점 문제를 다루지 않고 임의로 혹은 주기적으로 현재 참여중인 모든 GP들을 대상으로 SP 자격부여를 통해 SP를 생성하는 것이 보통이다[13]. 이는 시스템 성능을 저하시키는 오버헤드로 작용하므로 가능한 그 시간을 줄이는 것이 바람직하다. 그리하여 본 논문에서는, SP가 필요한 시점에 자격부여 심사를 통해 생성을 하는 통합 capacity 기반의 demand-driven SP 생성이 제안된다. 통합 capacity, C_{Max}는 다음과 같이 표현된다.

$$C_{Max} = \sum_{i=1}^{n} |Cluster_i| = \sum_{i=1}^{n} C_i \qquad (1)$$

그러므로 현재 참여중인 총 피어들의 수를 C_T라고 할 때, $C_T < C_{Max}$ 일 경우에는 새로운 SP의 생성이 필요하지 않으며, 그렇지 않을 경우, 즉 $C_T = C_{Max}$일 경우에는 GP들의 참여를 위해서 추가적인 SP가 필요하

므로 새로운 SP의 선발이 필요하게 된다. 하나의 방법으로, 새로운 SP의 생성을 위해서, 기존의 방식을 따라, 현재 참여중인 모든 GP들을 대상으로 SP 선발 작업을 수행할 수도 있는데, 본 논문에서 제안하는 방법은, $C_T = C_{Max}$가 되는 시점 이후, 참여를 원하는 GP들만을 대상으로 SP 선발 작업을 수행하는 것이다. 전자의 경우가 모든 GP를 대상으로 선발 작업을 하므로 별도의 많은 오버헤드가 필요하나, 후자의 경우에는 필요 시점 이후에 참여를 원하는 GP에 대해서만 SP 선발 작업을 수행하므로 전자에 비해서는 오버헤드가 적다는 장점을 가진다. 이를 ***demand-driven SP 선발***이라고 하며, 새로운 SP의 선발 요청은 $C_T = C_{Max}$가 되는 시점에서 이루어진다. 그리하여 일단 SP가 선발되면 capacity를 결정하고, GP들이 등록 할 수 있는 SP 집합에 포함시킨다. 만약 SP 자격부여가 어려우면 해당 GP는 GP로 등록시키거나 참여를 보류시킨다.

3. 수퍼피어 선택

SP의 선정에 관한 문제는 두 번째, 세 번째 문제의 해결을 위한 효율적 방법에 관한 연구가 주를 이루고 있는데, 이 중, 등록 SP의 선택은 선택 가능한 SP들의 집합을 대상으로 다양한 기준을 가지고 이루어진다. GP에게 가장 가까운 SP를 선택하는 거리 혹은 근접 기반의 선정 방식[17-18], 콘텐츠 유사성을 기반으로 선정하는 방법[13], 피어들 간의 간단한 정보 교환을 통한 선정 방식[16-17] 등을 들 수 있다. 세 번째 문제는 SPN의 토폴로지에 따라 달라질 수 있다. 보통 SPN을 랜덤 토폴로지로 가정하므로 새롭게 생성된 SP가 기존의 SPN에 참여하여 새로운 SPN을 구성하기 위해서는 어떤 SP에 연결되어야 하는 문제가 중요한 문제로 대두된다. 그러나 SSPN과 같이 트리 구조를 가진다면 참여를 원하는 모든 SP들은 하나의 RP에 연관되므로 이 문제는 간단하게 해결될 수 있으므로, 오히려 대규모 시스템 구축을 위한 SSPN의 확장에 관한 문제가 주요 이슈가 된다. 자발적 수퍼피어(Voluntary SP) 기반의 SSPN의 구성을 위한 수퍼피어 선택 및 SPN 구성 방법이[10]에 제안되었다.

Ⅲ. 확장 수퍼피어 네트워크 모델

확장 수퍼피어 네트워크(Extended Super-Peer Network, XSPN)는 그림-2의 예에서 보여준 바와 같이, SP들이 임의 레벨의 트리 형태로 연결된 네트워크이다. 맨 하위의 노드들, 즉 터미널 노드들은 모두 GP들이며, 나머지 중간 노드들은 SP들이다. SSPN에서는 하나만 존재하므로 RP라고 정의하였지만, XSPN인 경우에는 다중 레벨에서 RP 역할의 SP들이 존재하므로 레벨과 SP를 결합하여 표현하기로 한다. 그리하여 각 레벨의 SP는 자신의 레벨과 더불어 정의가 된다. 예를 들면, $SP^1, SP^2 ..., SP^i$로 정의 하는데 이때 SP^i $(i \geq 1)$ 는 XSPN에서 i번째 레벨의 SP를 의미한다. 그러므로 $i \geq 2$ 인 모든 SP^i는 SSPN에서의 RP 역할을 하는 SP들이다. 또한 최하위 레벨을 제외한 각 레벨에는 하나 이상의 SP들이 존재할 수 있다. 레벨-i에 존재할 수 있는 최대 SP들의 집합을 L^i_{Max} 라고 하면,

$$L^i_{Max} = \left\{ SP^i_j \mid j = 1, 2, ..., C^i_L \right\} \ , \qquad (2)$$

$C^i_L = \sum_{j=1}^{C^{i+1}_L} c(S^{i+1}_j)$ 로 표현될 수 있다. 여기서, SP^i_j는 i번째 레벨의 j번째 SP를 의미하며, $c(S^{i+1}_j)$는 (i+1) 레벨의 j번째 SP의 capacity를 의미한다. 이때, C^i_L는 i번째 레벨의 *레벨 capacity*라고 하며, 그 레벨에 존재할 수 있는 최대 SP의 수이다. 이는 상위 레벨 (i+1)번째 레벨의 SP들의 capacity에 의해 결정된다. 그러므로 어떤 레벨에 존재하는 현재 SP의 총 수를 $|L^i_T|$라고 하면, 항상 $|L^i_T| \leq C^i_L = |L^i_{Max}|$ 이 만족되어야 한다. 여기서 XSPN의 레벨은 터미널 노드를 제외한 맨 아래 노드의 레벨을 1로 하여 상향 증가 형이다. 레벨-1의 SP에는 GP들로 구성된 클러스터들이 연관된다. 레벨-0를 제외한 나머지 트리가 하나의 XSPN이며, 레벨-0의 GP들을 포함하는 전체가 XSPN 기반의 P2P 시스템이다. P2P 시스템에 현재 참여하고 있지 않은 어떤 피어라도 해당 시스템에 GP 혹은 SP로 참여할 수 있고, 언제라도 현재 구성에서 빠질 수 있다. 참여를 원하는 노드가 GP가 될 것이냐 SP가 될 것이냐는, 시스템 구성에 참여하는 과정에서 demand-driven 기반으로 정해진다.

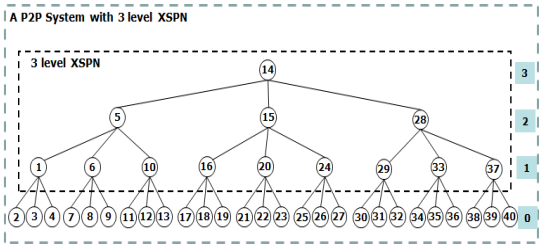

그림 2. 3-레벨 XSPN 기반의 P2P 시스템의 예
Fig. 2. A P2P system based on 3-level XSPN

그러므로 XSPN은 시간에 따라 레벨의 수와 각 레벨을 구성하는 SP들의 수가 변하는 동적 네트워크이다. 이는 고정된 레벨의 수를 가지는 SSPN에 비해 높은 확장성과 유연성을 가지게 된다. 3-레벨 XSPN의 레벨-1 노드에 3개의 GP들로 구성된 클러스터가 연결되어 구성된 4-레벨 P2P 네트워크의 예가 그림-2에 도시되어 있다.

IV. 확장 수퍼피어 네트워크 구성 알고리즘

본 장에서는 XSPN 구성 개념과 생성을 위한 알고리즘이 기술된다. XSPN은 운영체제의 계층구조 파일 시스템의 구성과 유사한 특성을 지닌다. 하나의 디렉토리 내부에는 해당 디렉토리의 부모 디렉토리와 자식 노드에 관한 정보를 포함하고 있는 것처럼, XSPN의 각 SP 노드도 부모와 자식 노드에 관한 정보를 가지고 있다. 예를 들면, 레벨 번호, 노드의 유형, 자식노드의 수 및 리스트, 노드 생성 시간, 현재 노드의 uptime과 누적 uptime, 부모 노드 정보 등을 포함하고 있으며, 필요시 사용한다.

1. XSPN 노드의 상태 분류

XSPN의 각 노드는 SP이다. XSPN의 구성을 위하여 각 SP가 관리하는 주요 정보만 살펴보면 [그림-3]에 보여준 바와 같이, ① 자신의 자식 SP들의 id와 각 자식 SP^{id}에 대해 $|Cluster_{id}| = C^{id}$ 여부를 알려주는 FULL 정보를 보관하고 있는 ChildPeerList(CPL), ② 부모피어의 유무에 대한 정보, ParentPeer(PP), ③ 자신의 capacity, N과 레벨 값 k가 있다.

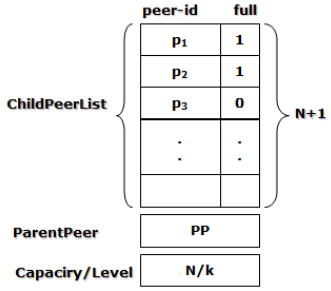

그림 3. 각 SP가 유지 관리하는 정보
Fig. 3. The information maintained by each SP

CPL은 자신에게 등록되어 있는 자식 SP들의 리스트이며, 그 크기는 capacity, N 보다 하나 큰, N+1 로 주어지는데, 이는 자기 자신이 자신의 CPL에 자식 피어로 등록되기 때문이다. 어떤 피어가 해당 P2P 시스템에 참여하기 위해서는 GP 혹은 SP로서 참여하게 된다. 그리고 GP로서 참여하느냐, 혹은 SP로서의 참여하느냐 문제는 GP로서의 등록 가능 여부와 SP로서의 자격부여 여부에 따라 결정되는데, 이는 각 노드의 상태, 즉 현재 등록되어 있는 자식 피어의 수와 부모 피어의 유무에 따라 [CASE-1]부터 [CASE-4]까지 4가지 경우로 분류할 수 있다.

여기서, p는 P2P 시스템에 참여하고자 등록을 요청하는 피어이며, R은 p의 등록 요청을 받는 SP이다. 본 논문의 XSPN은 *에이전트 SP(ASP)*라고 하는 특정 SP를 가지고 있는데, 모든 피어는 일단 ASP에게 등록 요청을 시도하면서 P2P 시스템에 참여하게 된다. 그리고 ASP는 레벨-1에 존재한다. 초기 값으로, $CPL(R) = \{R\}$ 로 주어지며, $PP(R) = \varnothing$ 이다. 즉, XSPN 초기에는 ASP 하나만 존재하고 있다는 것을 의미하고 있다. 그리고 $L(R)$은 R의 레벨이다. 그리고 p가 GP로서 등록이 가능하면 이를 우선으로 하는 것으로 하며(이를 GP-major 라고 한다), 또한 등록 요청을 받은 SP는 에이전트 SP가 아니라고 가정한다. 왜냐하면 에이전트 SP인 경우에는 간단한 과정을 거쳐 경과할 수 있으므로, ASP가 아닌 일반적인 경우로 가정한다. 그리하여 피어 p가 본 P2P에 참여하기 위해 R에게 등록을 요청할 경우, 다음 4가지 상태에 따라 피어의 참여 형태가 달라지면서 XSPN이 자생적으로 생성되게 되는데, 항상 트리 형태를 유지하면서 구성되도록 한다.

[CASE-1] : (|CPL(R)|<N+1) ∩ (PP(R)= ∅) :

R의 자식노드들의 수가 R의 capacity 보다 작고, 부모 피어가 존재하지 않는 경우이다. 그리하여 R에 p가 자식 피어로 등록이 가능하나, GP로 등록할 것이냐 SP로 등록할 것이냐가 결정되어야 한다. 만약 R이 ASP라면, GP로서 등록되지만, 그렇지 않은 경우, 즉 $L(R) \geq 2$ 인 경우에는, R의 자식피어 또한 SP이므로, GP-major 규칙에 따라, 이들 자식 SP의 연결 상태를 조사하여 FULL이 아닌 SP를 R로 대체하여 동일한 과정을 반복한다. 그리하여 모든 자식 SP의 연결 상태가 FULL이거나 혹은 R의 자식 레벨이 최하위 레벨인 레벨-1에 도달할 때까지 깊이 우선(Depth-First) 방식으로 작업을 반복한다. GP-major를 기반으로 하는 깊이 우선 방식으로 피어들이 GP 혹은 SP로의 할당이 이루어지면 하위 레벨부터 각 SP의 capacity를 채우면서 XSPN이 구성되므로 CPL 상의 자식 SP들의 현재 연결 상태가 FULL 상태이면 그 이하의 레벨도 모두 FULL 상태이므로 더 이상 상태 조사가 필요 없어지는 장점을 가지게 된다.

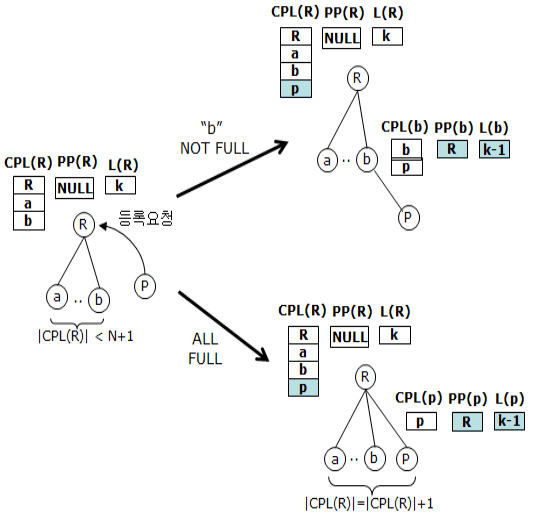

그림 4. (|CPL(R)|<N+1) ∩ (PP(R)= ∅)인 경우 최종 상태에 따른 p의 등록 과정

Fig. 4. The registration process for p according to the final status when (|CPL(R)|<N+1) ∩ (PP(R)= ∅)

그리하여 어느 레벨에서 모든 자식 SP들의 연결 상태가 capacity를 채운 FULL이라면, R의 자식 SP로 등록된다. 이때 SP 자격여부를 판단하는 과정을 통해 SP 자격여부가 결정되며, 자격부여에 실패하면 참여가 보류되고, 자격부여가 성공하면 SP로서 R의 자식피어가 된다. 반복 과정을 통해 R의 자식 레벨이 레벨-1에 도달한 경우에는 FULL 상태가 아닌 SP의 GP로서 등록시킨다. 최종 단계에서의 과정을 보여주고 있는 것이 그림-4이다.

[CASE-2] : (|CPL(R)|=N+1) ∩ (PP(R)= ∅) :

등록 요청을 받은 R의 자식 노드들의 수가 자신의 capacity에 도달하여, 당장 자식 피어로서 등록이 불가능하며, 또한 R의 부모피어도 존재하지 않는 경우이다. 그러므로 등록을 요청한 피어 p가 등록할 SP가 존재하지 않으므로 새로운 SP의 생성이 필요한 시점이며, demand-driven SP 생성 과정과 성능 추정을 통해 p의 capacity를 정하고 SP 자격 여부를 판단하여 자격이 불가능하면 참여를 기다리게 하며, 자격이 가능하면 R로 하여금 p를 자신의 부모피어로 등록시키도록 하고, 동시에 R을 p의 자식피어로 등록한다.

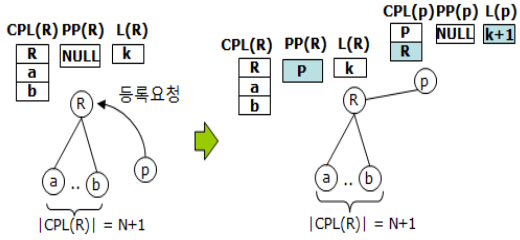

그림 5. (|CPL(R)|=N+1) ∩ (PP(R)= ∅ 인 경우 SP 자격을 얻은 경우 R의 부모피어가 되는 과정

Fig. 5. When (|CPL(R)|=N+1) ∩ (PP(R)= ∅ , the process of becoming a parent peer of R with getting SP qualification

즉 참여를 요청한 p가 새로운 SP로서 레벨의 증가를 초래하면서 XSPN의 확장이 이루어진다. 이때 $L(R) = k$ 인 경우 $L(p) = k+1$ 이 된다. 즉 R의 현재 연관 상태가 FULL인 경우에는 그렇지 않은 경우에 비해 등록 과정이 단순함을 알 수 있다. 이 과정을 보여주고 있는 것이 그림-5이다.

[CASE-3] : (|CPL(R)|<N+1) ∩ (PP(R) ≠ ∅) :

가장 일반적인 경우에 해당되며, [CASE-1]의 경우와 유사하며 단지 부모 노드가 존재한다는 것만 다른 경우이다. 피어 p가 등록을 요청한 R의 연결 상태에 여유가

있어 R에 등록이 가능하며, R의 부모피어가 존재하는 경우이다. [그림-6]에서와 같이, R에 연결할 수 있는 여유가 있으면서 Q와 같은 부모 피어가 존재한다는 것은 과거 언젠가 [CASE-2]가 발생하였다는 것, 즉 |CPL(R)| = N+1 이었다는 것을 의미한다. 그럼에도 불구하고 현재 |CPL(R)| < N+1 이라는 것은, 기존에 등록했던 자식피어가 XSPN에서 탈퇴하여, 참여 공간이 생긴 경우이므로, R에 자식피어로 등록될 수 있으며, 자신의 레벨은 k-1이 된다.

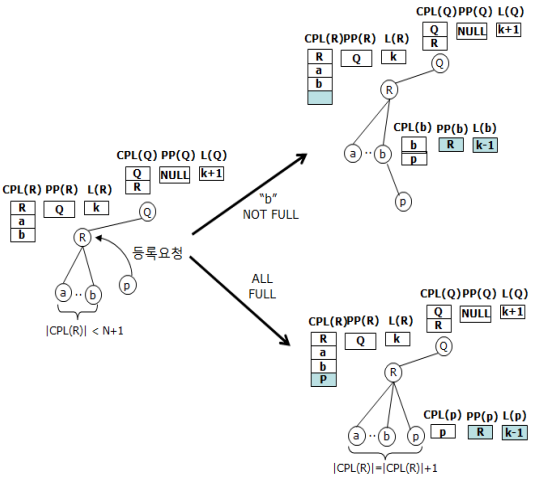

그림 6. (|CPL(R)| < N+1) ∩ (PP(R) ≠ ∅) 경우의 과정
Fig. 6. The case: (|CPL(R)| < N+1) ∩ (PP(R) ≠ ∅)

그러므로 이 경우는 상황이 [CASE-1]과 동일하다고 볼 수 있으므로 [CASE-1]의 경우처럼 GP-major 규칙을 따르면서 GP 혹은 SP로서 등록 과정을 따르도록 한다. 이 과정을 보여주고 있는 것이 그림-6이다.

[CASE-4] : (|CPL(R)|=N+1) ∩ (PP(R) ≠ ∅) :

수퍼피어 R의 자식피어 수가 최대이므로, 현재는 R의 하위 레벨의 모든 노드 어디에도 GP로서든 SP로서든 등록이 불가능하다는 것을 의미하고 있으나, R의 부모피어가 존재하는 경우이다. 그러므로 등록을 요청한 피어 p는 등록이 불가능한 R 대신에, R의 부모피어인 Q에게 다시 등록을 요청한다. 즉, R을 Q로 대체하여 동일한 과정을 반복하는 것이다. 이는 수퍼피어 Q와 피어 p에 대해 상기 4 가지 경우를 다시 적용하는 것과 등가적이며, 그 과정을 보여주고 있는 것이 그림-7이다.

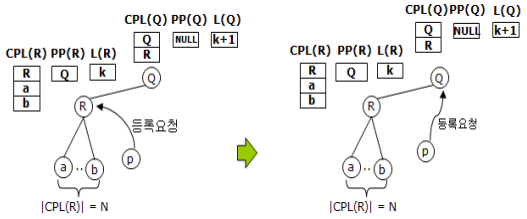

그림 7. (|CPL(R)|=N+1) ∩ (PP(R) ≠ ∅ 인 경우의 과정
Fig. 7. The case: (|CPL(R)|=N+1) ∩ (PP(R) ≠ ∅

2. XSPN 구성 알고리즘

[CASE-1]부터 [CASE=4]까지의 4가지 경우를 기반으로 XSPN을 구성하는 알고리즘을 기술하면 그림-8과 같다. 어떤 피어 p가 P2P 네트워크에 참여를 위해 R에게 등록 요청을 할 경우 다음을 따른다.

void **RegisterPeer**(generic_peer p, super_peer R) {
1. R의 현재 연결상태인 |CPL(R)|과 PP(R)을 조사하여 ConState에 저장
2. **switch(ConState)**

case CASE-1 : //((|CPL(R)|<N+1) ∩ (PP(R)= ∅))
if ((L(R) == 1) && NOT_FULL) {
CPL(R) = CPL(R)+p ; // GP로서 R에 등록
register = TRUE ;
}
return ;
else { // L(R) > 1
for $\forall e \in CPL(R)$ {
RegisterPeer(p, e) ;
if (!register)
if (QualifySP(p)) {
PP(R) = p ;
CPL(p)= {p, R};
PP(p)= ∅ ;
L(p) = L(R)+1 ;
}
return ;
}
case CASE-2 : // ((|CPL(R)|= N+1) ∩ (PP(R)= ∅))
if (QualifySP(p)) {
PP(R) = p ;
CPL(p)= {p, R};
PP(p)= ∅ ;

```
    L(p) = L(R)+1 ;
  }
  case CASE-3 : // ((|CPL(R)|< N+1) ∩ (PP(R) ≠ ∅ ))
   // [CASE-1]과 동일
  case CASE-4 : // ((|CPL(R)|= N+1) ∩ (PP(R) ≠ ∅ ))
   RegisterPeer(p, PP(R)) ; // 부모 피어에게 등록 요청
   return ;
  }
```

그림 8. XSPN 구성 알고리즘
Fig. 8. Algorithm for constructing an XSPN

3. XSPN 구성 예

XSPN 구성 알고리즘을 기반으로 레벨 3까지의 XSPN을 구성하는 과정을 보여주고 있는 것이 그림-9이다. 여기서 각 노드의 capacity는 모두 3으로 가정하였다. 그림-9에서 보아 알 수 있듯이 레벨-2의 구성 시작 점, 즉 S_0^2 이 생성된 순간이 SSPN과 동일함을 알 수 있다.

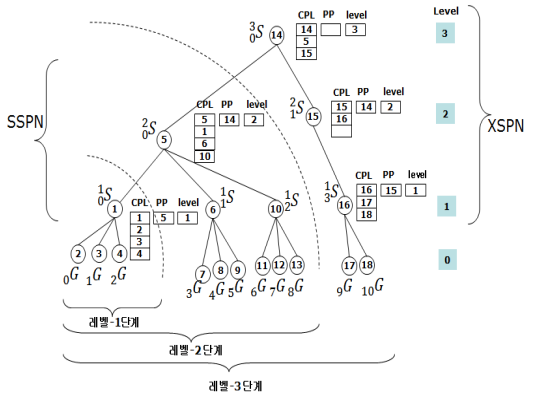

그림 9. 그림 8의 알고리즘을 기반으로 생성된 3-레벨 XSPN의 예
Fig. 9. An example of constructing 3-level XSPN based the algorithm shown in Fig. 8

V. 결론 및 향후연구

SP는 어떤 GP들의 집합에 대해 서버처럼 동작하는 특수한 피어이다. SP들의 네트워크를 구성하는 문제는 SPN에 기반을 둔 P2P 시스템에 있어서 중요한 문제 중의 하나이다. 이러한 SPN은 랜덤 그래프의 형태를 가지고 있는 것이 일반적이며, 대규모 피어들을 수용하기 위해서는 SPN 또한 규모성을 가지면서 확장되어야 한다. 본 논문에서는 이러한 대규모 P2P 시스템을 위한 트리 기반의 SPN의 계층적 구성 방법을 통한 확장 기법이 제안되었다. SSPN을 기반으로 하여, 그것을 일반화시키면서 SPN을 확장하는 방법을 제시하였다. 제안된 알고리즘에 의해 생성된 P2P 시스템은 높은 확장성 및 유연성을 가지고 있으며, 컴퓨팅 자원을 효율적으로 공유하고 관리, 전송하기 위한, Gnutella-2와 유사한 대규모 분산형 P2P 네트워크의 구성을 위해 적당하다고 할 수 있다. XSPN을 기반으로 원격 모니터링을 위한 네트워크를 구성하여, 라이브 혹은 저장된 멀티미디어 데이터 파일을 노드들 간에 멀티캐스트 스트리밍 방식으로 전송하면서 동시에 재생 하는 원격 모니터링 시스템을 구축하여 다양한 분야로 활용할 수 있는 방안을 연구 중에 있다.

References

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

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

[3] Gnutella2, http://www.gnutella2.com

[4] KaZaA, http://www.kazaa.com

[5] R. A. Schollmeier, "Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architecture and Applications," Proceedings of the 1st Int'l Conf.onPeer-to-PeerComputing (P2P'01), (2001), 101-102,
DOI: https://doi.org/10.1109/p2p.2001.990434

[6] E. K. Lua, *et al*, "A Survey and Comparison of Peer-to-Peer Overlay Network Schemes," IEEE Communications Survey & Tutorial, Vol. 7, No. 2, March 2005, pp.72-93,
DOI: https://doi.org/10.1009/comst.2005.1610546

[7] I. Clarke et al., "Freenet: A Distributed Anonymous Information Storage and Retrieval System," Proc. of the ICSI Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, 2000, LNCS 2001,
DOI: https://doi.org/10.1109/3-540-44702-4_4

[8] B. Yang, and H. Garcia-Molina, " Designing a Super-Peer Network," Proc. of 19th Int'l Conf. on Data Engineering, Bangalore, India, (2003)

DOI: https://doi.org/10.1009/icde.2003.1260781

[9] M. Liu, M. E. Harjula, and M. Ylianttila, "An Efficient Selection Algorithm for Building a Super-Peer Overlay," Journal of Internet Services and Applications, 4(4), (2013)
DOI: https://doi.org/10.1186/1869-0238-4-4

[10] W.-H. Chung, "A Super-Peer Selection Strategy for Peer-to-Peer Systems," Advanced Science & Technology Letters, 125, Jeju island, (2016), 25-29
DOI: https://doi.org/10.14257/astl.2016.125.05

[11] GP. Jesi, A. Montresor,, O. Babaoglu, "Proximity-aware Superpeer Overlay Topologies." Self-Managed Networks, Systems, and Services, 3996, LNCS, (2006), 43 - 57
DOI: https://doi.org/10.1007/11767886_4

[12] J. Yu, and M. Li, "CBT: A Proximity-Aware Peer Clustering System in Large Scale BitTorrent-like Peer-to-Peer Networks," Computer Communications 31(3), (2008), 591 - 602
DOI: https://doi.org/10.1016/j.comcom.2007.08.020

[13] S.-H. Min, J. Holiday, and D.-S. Cho, "Optimal Super-Peer Selection for Large-Scale P2P System," Proc. of 2006 Int'l Conf. on Hybrid Information Technology (ICHIT'06), (2006)
DOI: https://doi.org/10.1109/ichit.2006.253666

[14] P. Garbacki, D, H. J. Epema, and M. Steen, "The Design and Evaluation of a Self-Organizing Super-Peer Networks," IEEE Trans. on Computers, 59(3), (2010), 317-331
DOI: https://doi.org/10.1109/tc.2009.157

[15] W. -J. Kang, "A Method for Semantic Access Control using Hierarchy Tree," The Journal of The Institute of Internet, Broadcasting and Communication(JiiBC), Vol. 11, No. 6, pp.223-234

[16] M. Jelasity, W. Kowalczyk and M. van Steen, Newscast Computing, Internal Report IR-CS-006, Vrije Universiteit Amsterdam, (2003)

[17] J. Ledlie, J. M. Taylor, L. Serban, and M. Seltzer, "Self-Organization in Peer-to-Peer Systems," Proc. of the 10th Workshop on ACMSIGOPS European Workshop, (2002), 125-132
DOI: https://doi.org/10.1145/1133373.1133397

[18] T. E. Ng, and H. Zhang, "Predicting Internet Network Distance with Coordinates-based Approach," Proc. of INFOCOM, (2002)
DOI: https://doi.org/10.1109/infcom.2002.1019258

저자 소개

정 원 호(정회원)

- 2010년 3월 ~ 현재 : 덕성여자대학교 디지털미디어학과 교수
- 1992년 3월 ~ 1992년 7월 : IBM TJ Watson연구소 방문연구원
- 1989년 3월 ~ 2010년 2월 : 덕성여자대학교 컴퓨터공학부 교수
- 1984년 3월 ~ 1984년 8월 : KAIST 위촉 연구원
- 1978년 11 ~ 1984년 2 : ㈜대한전선, ㈜대우전자

※ 본 연구는 덕성여자대학교 2015년 연구비 지원으로 이루어졌습니다.