제20회 한국정보처리학회 추계학술발표대회 논문집 제10권 제2호 (2003. 11)

트리 형태의 방송 시스템

이필환*,이원돈*
*충남대학교 컴퓨터 과학과
e-mail : indizarm@brain.cnu.ac.kr

Tree-Shaped Broadcasting System

Lee, Pil-hwan*, Won Don Lee*
*Dept. of Computer Science, ChungNam National University

요 약

Tree-shaped broadcasting system 은 기본적으로는 unicast server client model 에 기반을 둔다. 현재 multicast 에 대한 논의가 많이 이루어지고 있음에도 불구하고 실제적으로 적용을 하기에는 여러 가지 부담이 따르는 것이 사실이다. 따라서 기존의 조건에서 실시간 데이터 서비스를 하기 위한 방법을 제시한다

1. 서론

현재 인터넷 상의 서비스는 거의 대부분이 unicast 형태의 server - client 모델에 기반한다. Unicast 는 단일 송신자와 단일 수신자 간의 통신을 의미하며, 이것은 server 와 각각의 client 사이에 지속적이든 아니든간에 독립적인 연결을 가지고 있어야 한다는 것을 의미한다.

따라서 client 의 수가 늘어날수록 server 는 요구를 수용하기 위해서 더 많은 수의 연결을 생성, 관리해야 한다는 것을 의미하며, server 측의 비용이 상당히 증가할 수 밖에 없다.

이에 비해서 multicast 는 하나의 연결/ 전송만으로 다수의 client 에게 동일한 데이터를 전송할 수 있다는 장점이 있으나, Ipv4 에서는 지원하지 않으며, multicast 가 지원되는 장비들이 일반적이지 않다는 문제점이 있다.

그러므로 여기에서 제안하는 방법은 기존의 IP 네트워크의 틀에서 server 의 비용 부담을 최소화하면서도 client 에게 어느 정도 이상의 서비스를 제공하려는데 중점을 두었다.

Tree-shaped Broadcasting System 은 기본적으로 application layer 이상에서 고안되었고, server daemon/ program 과 client program 이 system 을 이루는 중요한 요소이다. 기본 아이디어는 client 의 server 역할 분담과 안정적인 서비스를 위해서 각 client 가 2 군데로부터 서비스를 받고 자신 역시 최대 6 군데로 서비스를 하며, 이 모양이 tree 에 가까운 graph 로 형태를 취므로, Tree-shaped Broadcasting System 이라고 표현했다.

이 system 의 특징은 다음 몇 가지로 요약할 수 있다.

server 는 자신과 직접 연결되어 있는 client 에만 서비스를 하며, client 들의 질문에 결정을 내리고 통보하는 역할을 한다.

client 는 다른 serve-client 모델과는 달리 조금 더 많은 부담을 떠안게 되는데, 자신보다 하위의 client 에 대한 서비스와 time delay 에 대한 간단한 식으로 표현되는 cost function 을 바탕으로 전체적인 system 의 성능 향상과 형태 유지를 위해서 분주하게 움직이게 된다.

그리고 다른 멀티 미디어 서비스들 (VOD 등)과는 달리 특정한 시간에는 오직 하나의 contents 만을 서비스하며 (1 contents/ server) client 는 자신이 원하는 content 를 서비스하는 server 를 고르는 것은 가능하나, 현재 서비스중인 content 에서의 탐색(seek) 이나 일시정지(pause)등은 허용되지 않는다. (마치 생방송 중인 TV 나 라디오와 같다.)

2. 본론

1) 시스템의 구성

시스템은 다음과 제목과 마찬가지로 트리 형태를 띄

본 논문은 2003 년도 충남대학교 발전기금재단의 지원에 의하여 연구되었음

고 있다.

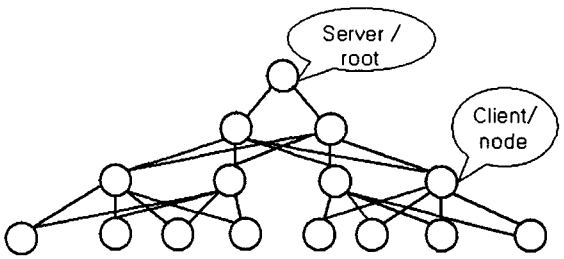

그림 1 System

root 에는 실제로 데이터를 서비스하는 server 가 위치하고 root 를 제외한 나머지 node 에는 일반적인 client 가 위치한다.

Client 들은 일반적인 server-client 모델에서의 client 와는 달리 어느 정도 server 와 비슷한 역할을 할 수 있다. 각 client 는 안정적인 서비스를 위해서 상위의 2 개 node 로부터 데이터를 전달 받고 또 자신 역시 하위의 node 들에게 전달하게 된다.

그런데 여기서 발생하는 문제는 각각의 node, 즉 client 들은 일반 사용자들이므로 server 가 서비스를 하는 동안 system 에 남아 있는다라고 확실히 말을 할 수 없다는 것이다. 다시 말하면 서비스를 받는 도중에 client 는 system 을 이탈할 수 있으며, 이는 system 의 전체적인 안정성에 큰 영향을 끼치게 된다.

2) Spare node

기본적으로 spare node 는 다른 형태의 client 가 아니다. 다른 client 와 동일한 형태의 client 이다

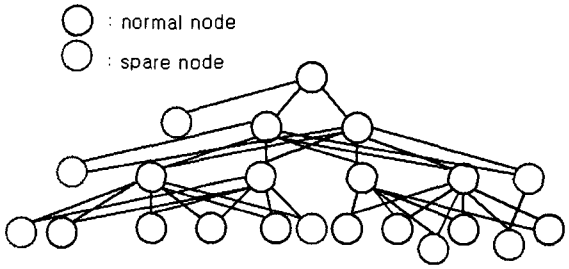

그림 2 System with spare nodes

단, 차이가 있다면 spare node 는 서비스를 받기만 할 뿐 자신보다 system 상에서 하위에 위치하는 node 들에게 데이터를 전달하지 않는다는 차이가 있다.

Spare node 의 기본적인 역할은 다른 client 들이 system 을 이탈했을 경우에 그 자리를 대신하는 역할을 한다. 물론 spare node 역시 다른 node(client)와 마찬가지로 예기치 않게 system 을 이탈할 수 있다. 그러나 spare node 가 system 을 이탈하지 않을 경우 다른 d 이탈한 node 의 위치를 대신하는 것은 일반적인 node 를 움직여서 처리하는 것보다 훨씬 수월하다.

예를 들어서 일반 node 가 다른 이탈한 node 의 위치를 대신하게 되면 하위 node 들을 재조정 해줘야지 그렇지 않은 경우에는 전체적으로 system 이 틀어지게 된다. (skewed)

3) ID

다른 tree 와 같이 이 system 에서도 동일한 형태로 ID 를 가진다. ID 는 각 node 를 구분하는 기준이며, root 가 1 그리고 나머지는 root 에서 가까운 순서대로 ID 를 갖게 된다.

Spare node 역시 ID 를 갖게 되는데 spare node 의 ID 는 자신의 sibling node 중 left node 의 ID 에 적당히 큰 수를 더한 값으로 한다.

그 이유는 spare node 를 다른 일반적인 node 와 구분하기 위한 방법인 동시에 spare node 의 위치를 표시하기 위한 방법이다.

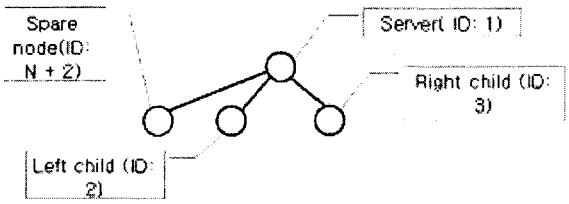

그림 3 ID

4) Block number

system 에서 서비스 하는 데이터는 기본적으로 일정한 format 의 완전한 file 이다.

모든 데이터를 한꺼번에 보내는 형식이 아니라 일정한 크기의 데이터 조각을 순차적으로 보내게 된다. 그리고 그 데이터 조각들을 구분하고 현재 서비스하거나 서비스 받고 있는 상태를 알기 위해서 block number 를 부여하게 되는데, 파일의 앞부분에서부터 부여되며, block 의 크기로 현재 파일을 읽고 있는 위치를 나눈 몫에 1 을 더한 값으로 한다.

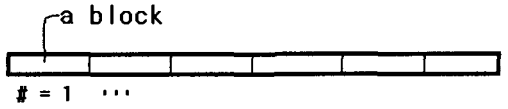

그림 4 Block Number

5) System 구성의 순서

어떤 의미에서 이 system 에서 root (server) 다음으로 중요한 것은 spare node 이다 따라서 system 을 구성할

때 root 는 spare node 를 우선 system 에 집어 넣고 left node 와 right node 순서대로 system 을 구성한다.

System 의 구성 도중에 각 node 들이 이탈을 하게 되면 처리하는 방법이 다르다. 일반적인 node 는 자신이 트리에서 leaf node 가 아닌 이상 하위의 node 를 가지게 된다. 따라서 그 node 의 부내는 하위 node 들에 대한 서비스가 지장을 받고 있다는 것을 의미한다.

그렇기 때문에 일반 node 의 이탈이 발생한 경우에는 자신의 sibling node 나 또는 하위 node 들에 의해서 처리되어야 한다.

이와는 반대로 spare node 는 자신의 이탈이나 다른 node 의 이탈을 처리하기 위해 다른 위치로 갔을 경우에 영향을 받는 node 들이 없다. 그렇기 때문에 spare node 가 이탈한 node 들을 대신해야 한다.

그런데 이 spare node 가 없을 경우에는 일반 node 가 다른 node 의 이탈을 처리해야 하므로 전체적인 조정이 불가피하므로 system 의 구성에서도 spare node 를 우선적으로 집어넣고, spare node 의 이탈이 보고되는 즉시 root 에서는 system 에 새로 들어오는 node, 다시 말하면 서비스를 받기 위해서 server (root)에 접속하는 client 를 spare node 로 집어 넣는다.

6) 1st parent & 2nd parent

각 node 는 상위 2 개의 node 로부터 서비스를 받는다. 이것은 안정적인 서비스를 위한 것인데, 2 군데서 서비스를 받는다는 사실이 전체적으로 혼란을 불러올 수 있다. 따라서 각 node 간의 관계를 좀더 명확히 하는 것이 필요한데, 기본적인 트리에서 parent node 가 1st parent 이며 이 1st parent 는 자신의 child node 에 대해서 1 차적인 책임을 지니며, 이와 상대되는 개념의 1st children 역시 자신의 parent node 의 이탈에 대해서 1 차적인 책임을 지닌다. (spare node 포함)

2nd parent 는 자신의 1st parent 의 sibling node 로 원활한 서비스를 위한 것이다.

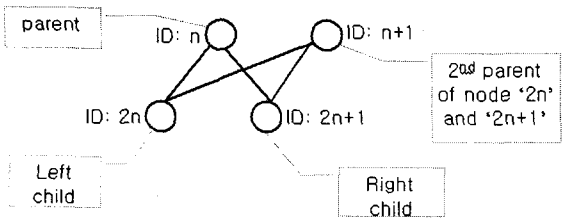

그림 5 1st parent and 2nd parent

7) 성능 향상을 위한 노력

system 이 최대한의 성능을 발휘하기 위해서는 상위 node 의 네트워크 상태와 자체 system (개인 PC system) 의 성능이 하위 node 에 비해서 좋아야 한다.

다시 말하면 root 에 가까운 node 일수록 좋은 성능을 보여야 하는데 이것은 상위 node 의 성능이 좋지 않을 경우, 그 node 의 하위 node 들은 그것의 영향을 받아서 전체적으로 system (트리 형태의 방송 시스템)의 성능 저하로 이어지게 된다.

따라서 좋은 성능을 보이는 node 는 현재의 위치와 상관없이 상위로 끌어올리는 노력을 할 필요가 있다. 그리고 판단하기 위한 기준으로 간단한 cost function 을 사용하는데 그것은 다음과 같다.

$$E_{system} = \sum delay\_child + \sum delay\_par + \sum_{par<cur} \alpha(play\_par - paly\_cur)^2 + \sum_{child<cur} \alpha(play\_child - play\_cur)^2$$

그림 6 Cost Function

전체 system 의 energy (cost)는 자신과 child node 사이의 delay, parent node 와의 delay 그리고 현재 자신이 play 하는 위치와 parent node 가 play 하는 위치의 차이 등을 합한 값이 된다. (sum 에서의 조건, 예를 들어서 parent node 의 play 위치가 자신의 현재 play 위치보다 앞서야 한다거나 하는 것은 절대적인 것은 아니다)

그러나 문제는 앞서 말한 바와 같이 각 node 들이 root 에 의해서 조절되는 것이 아니라 예기치 않게 system 을 이탈할 수 있다는 성질을 갖기 때문에 전체 system 의 energy 를 구한다는 것은 무의미하다.
(energy 를 구하는 동안 system 을 이탈하거나 새로 들어오는 node 가 있다면 그것은 무의미하게 된다)

따라서 local minima 에 빠질 염려는 있으나 부분적으로 각 node 끼리 미리 위치를 바꾸었을 경우의 cost (energy)를 계산해서 줄어드는 방향으로 자신들끼리 위치를 조정한다.

3. 결론

여기서 제안하는 system 은 일반적인 ftp 나 vod 와는 다른 형태를 취한다.

데이터를 온전하게 서비스하는 것도 아니고, 사용자들이 원하는 시각에 원하는 내용을 서비스 받으면서 임의로 서비스 받고 싶은 부분을 고를 수 있는 것도 아니다.

그러나 상대적으로 적은 computing power 로 많은 사용자들에게 서비스를 할 수 있다는 점과 데이터의 불법적인 copy 가 직접적으로는 힘들다라는 점이 불완전하지만 나름대로의 장점이 아닐까 생각한다.

아직 이 시스템에 관한 아이디어는 좀 더 많은 개선의 여지가 있으며, 지금도 개선을 위해서 노력하고 있는 중이다.

참고문헌

[1] Horowitz, Ellis/ Sahni, Sartaj/ Anderson-Freed, S "Fundamentals of Data Structures in C"
[2] Rick Boivie/ Nancy Feldman/ Christopher Metz "Small Group multicast : A new solution for multicasting on the internet" IEEE INTERNET COMPUTING May, June 2000
[3] Masataka Otha/ Jon Crowcroft "Static Internet multicast"
[4] Ituso Takanami/ Katsushi Inoue/ Takahiro Watanabe "Reconfigurable Fault Tolerant Binary Tree" IEEE 1994
[5] Siu-Cheong Chau "On implementing large fault-tolerant binary tree architecture in WSI" IEEE 1993